Decentralized Multichannel Medium Access Control: Viewing Desynchronization as a Convex Optimization Method

Nikos Deligiannis[*]
Vrije Universiteit Brussel - iMinds
University College London
ndeligia@etro.vub.ac.be

João F. C. Mota
University College London
j.mota@ucl.ac.uk

George Smart
University College London
g.smart@ee.ucl.ac.uk

Yiannis Andreopoulos
University College London
i.andreopoulos@ucl.ac.uk

ABSTRACT

Desynchronization algorithms are essential in the design of collision-free medium access control (MAC) mechanisms for wireless sensor networks. DESYNC is a well-known desynchronization algorithm that operates under limited listening. In this paper, we view DESYNC as a gradient descent method solving a convex optimization problem. This enables the design of a novel decentralized, collision-free, multichannel medium access control (MAC) algorithm. Moreover, by using Nesterov's fast gradient method, we obtain a new algorithm that converges to the steady network state much faster. Simulations and experimental results on an IEEE 802.15.4-based wireless sensor network deployment show that our algorithms achieve significantly faster convergence to steady network state and substantially higher throughput compared to the recently standardized IEEE 802.15.4e-2012 time synchronized channel hopping (TSCH) scheme. In addition, our mechanism has a comparable power dissipation with respect to TSCH and does not need a coordinator node or coordination channel.

Categories and Subject Descriptors

C.2.1 [**Computer- Communication Networks**]: Network Architecture and Design, Distributed Networks;
G.1.6 [**Numerical Analysis**]: Optimization, convex programming.

[*]Corresponding author. The authors acknowledge the support from the U.K. EPSRC, projects EP/K033166/1 and EP/M00113X/1.

General Terms

Theory, medium access control, distributed systems

Keywords

Decentralized multichannel coordination, medium access control, synchronization, desynchronization, gradient methods.

1. INTRODUCTION

Data-intensive wireless sensor network (WSN) applications, such as wireless visual sensor networks [2, 16], drone robots [45], or wireless capsule endoscopy [17], extend the human ability to visualize areas, detect events or explore hard-to-reach environments. Such applications impose stringent communication requirements on the network. In particular, to transmit high amounts of sensed data, they require energy-efficient medium access control (MAC) protocols. In addition, self-inflicted packet collisions need to be avoided, as they lead to unwanted energy consumption. In this context, in order to achieve fair TDMA scheduling, it is key to develop MAC protocols that perform synchronization and/or desynchronization at the MAC layer [1, 7, 11, 12, 34, 44, 47, 49]. Moreover, to extend TDMA scheduling to large-scale networks, it is important to develop protocols that achieve desynchronization across multiple channels [42, 44, 47].

1.1 Related Work

Conventional multichannel MAC schemes for WSNs make use of a global coordinator node or clock (e.g., GPS system) [44, 49] and a coordination channel. Under this category, which we refer to as *centralized protocols for WSNs*, the state-of-the-art is the time-synchronized channel hopping (TSCH) [44] protocol, which has been included in the IEEE 802.15.4e-2012 standard [21, 51]. According to the concept of channel hopping, nodes can hop between the 16 channels of the 2.4 GHz band so that transmitters and receivers are synchronized and evenly spread across channels. In this way, nodes are not constantly using a channel with excessive interference.

According to TSCH, wireless sensor nodes reserve timeslots within the predefined slotframe interval and within the

16 channels of IEEE 802.15.4. The TSCH timeslot reservation procedure, however, follows a rather complex advertising request-and-acknowledgment (RQ/ACK) process on a coordination channel. The use of this coordination channel renders TSCH a centralized MAC protocol and leads to several drawbacks. First, this channel is prone to interference and occasional self-inflicted collisions when data-intensive communications are required. This is because, under high data rates, the nodes are set to advertise slot reservations very aggressively. Second, in case of interference, TSCH-based networks suffer from slow convergence to steady state and low connectivity, as many of the advertisement RQ/ACK messages on the coordination channel are not heard by the targeted nodes. Third, if slot advertising is not aggressive and nodes leave the network, their slots may remain unoccupied for long periods until another advertisement RQ/ACK process reassigns them to other nodes, thereby limiting the bandwidth usage per channel.

In the category of *decentralized WSN MAC-layer protocols* several synchronization or desynchronization mechanisms have been proposed recently [3, 6, 10–12, 19, 23–25, 29, 34, 35, 40, 41, 44]. These mechanisms are based on distributed synchronization and desynchronization primitives that achieve and maintain distributed coordination between local clocks of networked agents via the exchange of limited information. These algorithms are inspired by decentralized coordination phenomena observed in natural phenomena (e.g., fireflies beaconing, pacemaker cells of the heart, superconducting Josephson junctions, etc.) [23, 27, 34, 37], and are expressed mathematically through the pulse-coupled oscillator (PCO) model by Mirollo and Strogatz [27] and Peskin [36].

Based on the PCO model, several distributed synchronization and desynchronization algorithms with various important properties have been proposed for WSNs. In particular, the algorithms in [12, 33, 37] support limited listening; the works in [6, 11, 12, 29] provide solutions amenable to multi-hop network topologies and the hidden nodes' problem; and the works in [23–25] present mechanisms that enable a convergence speed-up.

The link between PCO-based synchronization methods and consensus algorithms for networked oscillators is established in [32]. However, only limited work has been focused on extending distributed desynchronization to the multichannel case [7].

1.2 Contribution

Inspired by the link between consensus algorithms and PCOs [32], we show that DESYNC [35], a well-known desynchronization primitive with limited listening, can be seen as a gradient descent method. This enables for the problem of decentralized multichannel medium access control to be casted as an optimization problem. In particular, we propose a new joint synchronization and desynchronization primitive that yields decentralized multichannel MAC coordination with limited listening. Using optimization theory tools we prove that our algorithm converges to perfect TDMA scheduling in a multichannel setting. In addition, by using Nesterov's accelerated version of the gradient method [30, 31], we propose an algorithm that provides for faster convergence to the steady network state.

Finally, via simulations and experiments using a WSN deployment abiding by the IEEE 802.15.4 standard, we show that the proposed mechanism and its fast counterpart lead to decentralized multichannel MAC-layer coordination, achieving faster convergence to steady state and higher network throughput than the state-of-the-art TSCH protocol, while incurring comparable power consumption.

1.3 Outline

The remainder of the paper is organized as follows: Section 2 presents the background on pulse-coupled oscillator methods, while Section 3 presents our new fast version of the DESYNC algorithm. Section 4 presents our novel formulation of multichannel coordination. Experiments on the proposed algorithms are given in Section 5. Finally, Section 6 draws the conclusions of the work.

2. BACKGROUND ON PCO PRIMITIVES

Consider a *fully-meshed* WSN comprising n nodes. Each node acts as a pulse-coupled oscillator (PCO) [27]. When a node does not interact with others (when, for example, listening is switched-off), it broadcasts a *fire message* or *pulse* periodically. This is modeled by assigning to node i a *phase* $\theta_i(t)$, whose value at time t is given by [12, 33]

$$\theta_i(t) = \frac{t}{T} + \phi_i \mod 1\,, \tag{1}$$

where $\phi_i \in [0, 1]$ is the *phase offset* of node i and mod 1 denotes the modulo operation with respect to the unity. Fig. 2.1 illustrates equation (1) graphically: the phase $\theta_i(t)$ of node i can be seen as a bead moving clockwise on a circle, whose origin coincides both with 0 and 1 [13, 27, 33, 34]. From Fig. 2.1(a) to Fig. 2.1(b), all phase beads moved clockwise, keeping their relative distance, i.e., they moved at the same speed. In Fig. 2.1(b), the phase of node $i-1$ reaches 1, at a time we designate by t_{i-1}, that is $\theta_{i-1}(t_{i-1}) = 1$. When this happens, the respective node broadcasts a fire message and resets its phase to 0, i.e., $\theta_{i-1}(t_{i-1}) \leftarrow 0$. A firing of a node can trigger a phase update on other nodes, as described next.

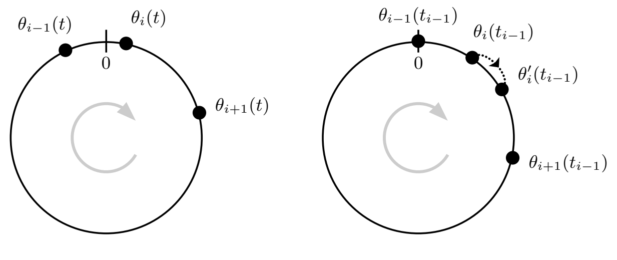
2.1 Mirollo-Strogatz and Peskin Model

When the nodes interact, e.g., by listening to each others' messages, they update their phases, according to a local state function $f(\theta_i(t))$, $f : [0, 1] \to [0, 1]$ that expresses the PCO dynamics. When node i receives a fire message from another node, it promptly updates its local state by a coupling parameter δ. Namely, node i modifies its phase as

$$\theta_i'(t) = f^{-1}\left(f(\theta_i(t)) + \delta\right) \mod 1, \tag{2}$$

where $f^{-1}(.)$ is the inverse PCO dynamics function.

Mirollo and Strogatz [27] proved that if f in (2) is monotonically increasing and concave then, for any $\delta > 0$, the firings of the oscillators will converge to synchrony. The same is achieved if f is convex, but the coupling parameter is negative ($\delta < 0$). Alternatively, when f is concave and $\delta < 0$, or when f is convex and $\delta > 0$, the nodes' phases will spread out, leading to desynchrony. Next, we mention existing PCO-based algorithms that perform (de)synchronization at the MAC layer of WSNs under limited listening. To express the rate of convergence, we introduce the notion of *firing round*, which is completed when each node in the network has fired exactly once.



(a) $t_{i+1} < t < t_{i-1}$ (b) $t = t_{i-1}$

Figure 1: Illustration of the phase update of node i according to the Desync algorithm: (a) at time instance t node $i+1$ and node i have fired (at time instances t_{i+1} and t_{i+1}, respectively) and node $i = 1$ is about to fire; (b) when node $i-1$ fires at time instance t_{i-1}, node i updates its phase from $\theta_i(t_{i-1})$ to $\theta'_i(t_{i-1})$, towards the average of the phases of nodes $i-1$ and $i+1$, its phase neighbors.

2.2 PCO-based Algorithms

Hong *et al.* [19] proposed a scalable, low-complex protocol that performs synchronization in large scale WSNs and has applications in cooperative reach-back communications. Their synchronization algorithm is based on the PCO dynamics model from Peskin [36]:

$$f(\theta) = C\,(1 - \exp(-\gamma T\theta))\,,$$

where $C - 1/\,(1 - \exp(-\gamma T))$ and γ is a leakage factor.

Pagliari *et al.* [33] proposed a PCO-based desynchronization algorithm with updates

$$\theta'_i(t_j) = (1-\alpha)\theta_i(t_j) \mod 1, \tag{3}$$

obtained by applying the convex function $f(\theta(t)) = -\log(\theta(t))$ and the coupling parameter $\delta = -\log(1-\alpha)$ in (2). The phase update in (3) is performed when another node, say node j, fires at time instant t_j. The *jump-phase parameter* $\alpha \in (0,1)$ controls the phase increment. The network reaches the *state of desynchrony* at time $\bar{t}$, after which the interval between consecutive firings is $1/n$ up to a small threshold ϵ. The update (3) was shown to achieve desynchrony even when each node carries out *limited listening*, i.e., listening for message broadcasts only within the $[T - \frac{T}{n}, T]$ interval of its own pulse cycle [33].

In the Desync algorithm [12, 35], the nodes are ordered according to their initial phases: $0 \le \theta_1(0) < \theta_2(0) < \cdots < \theta_n(0) < 1$. Assuming failure-free and noiseless beacon transmission and reception, the order of the firings in Desync is always the same [12, 35], i.e., node $i+1$ fires always after node i, for $i = 1,\ldots,n-1$, and node 1 fires always after node n. The phase θ_i of node i is updated based on the phases θ_{i-1} and θ_{i+1} of its phase neighbors[1], nodes $i-1$ and $i+1$, respectively. This is illustrated in Fig. 2.1(b): immediately after node $i-1$ transmits a fire message, node i modifies its phase towards the middle of the interval between the phases of nodes $i-1$ and $i+1$. The update formula for the phase of node i in Desync is given by [35]

$$\theta'_i(t_{i-1}) = (1-\alpha)\theta_i(t_{i-1}) + \alpha\frac{\theta_{i-1}(t_{i-1}) + \theta_{i+1}(t_{i-1})}{2}\,, \tag{4}$$

where t_{i-1} is the time instant in which node $i-1$ fires, i.e., $\theta_{i-1}(t_{i-1}) = 1$, and $i = 1,2,\ldots,n$, with periodic extension at the boundaries. When node i updates its phase, it has a *stale* knowledge of the phase of node $i+1$. By *stale* knowledge we mean that it only knows the previous value of θ_{i+1} and not the current one. This is because node $i+1$ modified its phase when node i fired, but the value of the new phase has not been "announced" yet [35]. In Desync, each node: *(i)* updates its phase once in each firing round; *(ii)* does not require knowledge of the total number of nodes, n, in the network; *(iii)* allows for *limited listening*, as only the messages from the two phase neighbors are required. These features make Desync quite popular [12,35]. Regarding the convergence speed of Desync, an operational estimate of the number of firing rounds required to reach convergence was established in [8,9].

3. DESYNC AS A GRADIENT METHOD

Driven by the connections between synchronization and consensus algorithms [32], we now describe the Desync primitive [12,35] as a gradient descent method applied to an unconstrained optimization problem.

Fig. 2 shows the first five phase configurations of a network with four nodes. The staleness of Desync makes each node i update its phase at iteration k using $\theta_{i-1}^{(k-1)}$ in place of $\theta_{i-1}^{(k)}$. For example, in Fig. 2(c), node 2 updates its phase at the firing of node 1. According to (4), this update requires the value of θ_1 and θ_3. Since node 2 is unaware of the current value of θ_3, i.e., the updated phase $\theta_3^{(1)}$, it will use $\theta_3^{(0)}$ instead; in other words, it uses a stale version of θ_3. Similarly, in step (e), node 4 updates its phase using a stale version of θ_1, $\theta_1^{(0)}$, and a non-stale version of θ_3, $\theta_3^{(1)}$. For the purpose of the interpretation, however, we assume that node 4, which was the first node to fire (cf. Fig. 2(a)), uses $\theta_3^{(0)}$ (in gray) and not $\theta_3^{(1)}$. We call this the *extra staleness assumption*. Notice that this assumption only applies to the node that first fired. We remark that the implementation of Desync using this assumption has a performance almost indistinguishable from the original one in [12,35].

The first step is to write (4) in terms of phase offsets. To do that, consider a node $2 \le i \le n-1$ and replace (1) into (4) at iteration k, using $\phi_i^{(k)}$ to denote the phase offset of node i at iteration k:

$$\begin{aligned}
\theta'_i(t_{i-1}) &= \frac{t_{i-1}}{T} + \phi_i^{(k)} \\
&= (1-\alpha)\Big[\frac{t_{i-1}}{T} + \phi_i^{(k-1)}\Big] \\
&\quad + \frac{\alpha}{2}\Big[\frac{t_{i-1}}{T} + \phi_{i-1}^{(k-1)} + \frac{t_{i-1}}{T} + \phi_{i+1}^{(k-1)}\Big] \\
&= \frac{t_{i-1}}{T} + (1-\alpha)\phi_i^{(k-1)} + \alpha\frac{\phi_{i-1}^{(k-1)} + \phi_{i+1}^{(k-1)}}{2}\,.
\end{aligned}$$

Subtracting t_{i-1}/T from both sides, we get

$$\phi_i^{(k)} = (1-\alpha)\phi_i^{(k-1)} + \alpha\frac{\phi_{i-1}^{(k-1)} + \phi_{i+1}^{(k-1)}}{2}\,. \tag{5}$$

Since θ wraps around 1, the expression for nodes 1 and n requires a correcting term: for $i = 1$, $\phi_n^{(k-1)}$ in (5) should

[1]We use the term "phase neighbors" to refer to nodes that fire consecutively. This concept of *neighbors* has nothing to do with neighbors in the communication network. We a assume a fully or densely connected communication network, that is, if their listening is switched-on, each node can communicate with any node in the network.

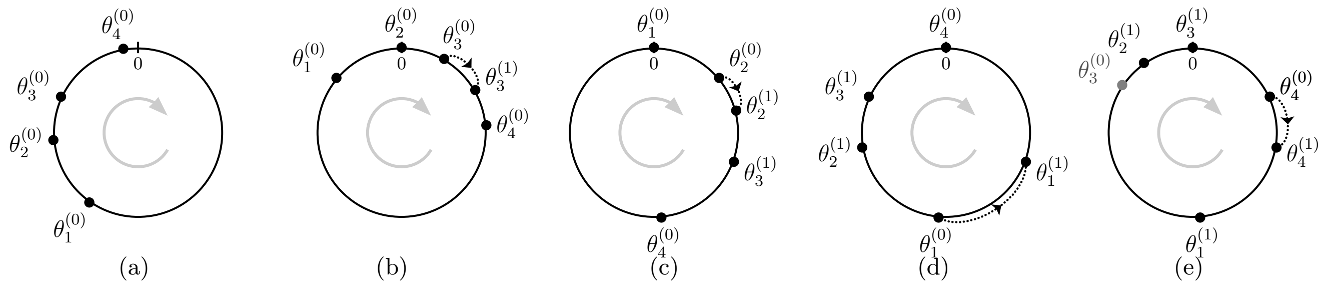


Figure 2: First five phase configurations of 4 nodes running Desync: (a) initial phases, before any firing; (b) first phase update: node 3 updates $\theta_3^{(0)}$ to $\theta_3^{(1)}$ at the firing of node 2. Note that between (a) and (b) nodes 4 and 3 fired, but did not cause any update. Remaining figures: node i fires and node j updates its phase, where (i,j) is $(1,2)$ in (c), $(4,1)$ in (d), and $(3,4)$ in (e). Due to the staleness of Desync and our extra staleness assumption for node 4, all updates use the initial values $\theta_1^{(0)}$, $\theta_2^{(0)}$,$\theta_3^{(0)}$, or $\theta_4^{(0)}$, and not the updated ones.

be replaced with $\phi_n^{(k-1)}-1$; for $i=n$, $\phi_1^{(k-1)}$ in (5) should be replaced with $\phi_1^{(k-1)}+1$. While the natural staleness of Desync allows writing (5) for $i=1,\ldots,n-1$ (with the correcting term for node 1), it is our extra staleness assumption that makes (5) also applicable to node n (with the proper correcting term). Using $\boldsymbol{\phi}^{(k)}=(\phi_1^{(k)},\phi_2^{(k)},\ldots,\phi_n^{(k)})\in\mathbb{R}^n$ to denote the phases of all nodes at iteration k, and $\boldsymbol{d}:=(1,0,\ldots,0,-1)\in\mathbb{R}^n$, we can now write (5) for all nodes in vector form:

$$\boldsymbol{\phi}^{(k)}=\begin{bmatrix}1-\alpha & \frac{\alpha}{2} & 0 & \cdots & 0 & \frac{\alpha}{2}\\ \frac{\alpha}{2} & 1-\alpha & \frac{\alpha}{2} & \cdots & 0 & 0\\ \vdots & & \ddots & & \vdots & \vdots\\ \frac{\alpha}{2} & 0 & 0 & \cdots & \frac{\alpha}{2} & 1-\alpha\end{bmatrix}\boldsymbol{\phi}^{(k-1)} - \frac{\alpha}{2}\boldsymbol{d}. \quad (6)$$

Equation (6) can be seen as a particular instantiation of a discrete-time consensus problem [14, 32, 39, 50] with an exogenous input (that is, the vector $\boldsymbol{d}$). In the control literature, consensus with exogenous inputs is usually referred to as multi-agent consensus with informed leaders [38]. We may resort to tools coming from linear system analysis to characterize the convergence properties of (6) and its equilibria subspace. However, we view (6) as an algorithm solving an optimization problem, since this can lead to a new accelerated version of desynchronization. We establish the following:

Proposition 1. *The* Desync *primitive in* (6) *is the gradient descent method,*

$$\boldsymbol{\phi}^{(k)}=\boldsymbol{\phi}^{(k-1)}-\beta\,\nabla g(\boldsymbol{\phi}^{(k-1)})\,, \quad (7)$$

with stepsize $\beta=\alpha/2$ applied to

$$\underset{\boldsymbol{\phi}}{\text{minimize}}\quad g(\boldsymbol{\phi}):=\frac{1}{2}\left\|\boldsymbol{D}\boldsymbol{\phi}-\frac{1}{n}\mathbf{1}_n+\boldsymbol{e}_n\right\|_2^2, \quad (8)$$

where $\mathbf{1}_n=(1,\ldots,1)\in\mathbb{R}^n$, $\boldsymbol{e}_n=(0,\ldots,0,1)\in\mathbb{R}^n$, and

$$\boldsymbol{D}=\begin{bmatrix}-1 & 1 & 0 & 0 & \ldots & 0\\ 0 & -1 & 1 & 0 & \ldots & 0\\ \vdots & \ddots & & \ddots & & \vdots\\ 0 & \ldots & 0 & 0 & -1 & 1\\ 1 & \ldots & 0 & 0 & 0 & -1\end{bmatrix}\in\mathbb{R}^{n\times n}. \quad (9)$$

Proof. We first compute the gradient of g:

$$\nabla g\left(\boldsymbol{\phi}\right)=\boldsymbol{D}^T(\boldsymbol{D}\boldsymbol{\phi}-\frac{1}{n}\mathbf{1}_n+\boldsymbol{e}_n)=\boldsymbol{D}^T\boldsymbol{D}\boldsymbol{\phi}+\boldsymbol{d}\,, \quad (10)$$

where $\boldsymbol{d}=\boldsymbol{D}^T\boldsymbol{e}_n$ is the vector in (6). Note that we used the fact that $\boldsymbol{D}^T\mathbf{1}_n=\mathbf{0}_n$. Hence, (7) becomes

$$\begin{aligned}\boldsymbol{\phi}^{(k)}&=\boldsymbol{\phi}^{(k-1)}-\beta\boldsymbol{D}^T\boldsymbol{D}\boldsymbol{\phi}^{(k-1)}-\beta\boldsymbol{d}\\ &=(\boldsymbol{I}_n-\beta\boldsymbol{D}^T\boldsymbol{D})\boldsymbol{\phi}^{(k-1)}-\beta\boldsymbol{d}\,,\end{aligned} \quad (11)$$

where $\boldsymbol{I}_n\in\mathbb{R}^{n\times n}$ is the identity matrix. Setting $\beta=\alpha/2$, and noticing that

$$\boldsymbol{D}^T\boldsymbol{D}=\begin{bmatrix}2 & -1 & 0 & 0 & \cdots & 0 & -1\\ -1 & 2 & -1 & 0 & \cdots & 0 & 0\\ 0 & -1 & 2 & -1 & \cdots & 0 & 0\\ \vdots & & \ddots & & & & \vdots\\ -1 & 0 & 0 & 0 & & -1 & 2\end{bmatrix}, \quad (12)$$

(11) is exactly (6). $\square$

Notice that, the optimality conditions for (8) implicitly impose the differences between consecutive phases to be $1/n$. To see that, let $\boldsymbol{\phi}^\star$ denote a solution of (8), i.e., $\nabla g(\boldsymbol{\phi}^\star)=0$ or $\boldsymbol{D}^T\boldsymbol{D}\boldsymbol{\phi}^\star=-\boldsymbol{d}$. From (12), we have

$$\phi_i^\star=\frac{\phi_{i-1}^\star+\phi_{i+1}^\star}{2}\,,\qquad i=2,\ldots,n-1\,, \quad (13)$$

$$\phi_n^\star=\frac{\phi_{n-1}^\star+(\phi_1^\star+1)}{2}\,, \quad (14)$$

where we omitted one equation, because $\boldsymbol{D}^T\boldsymbol{D}$ is rank deficient. Equations (13) impose each $\phi_i^\star$ to be the average of $\phi_{i-1}^\star$ and $\phi_{i+1}^\star$, for $i=2,\ldots,n-1$, and equation (14) imposes $\phi_n^\star$ to be the average of $\phi_{n-1}^\star$ and $\phi_1^\star+1$. This is shown in Fig. 3 for 5 nodes. The only possibility is to have $\phi_{i+1}^\star-\phi_i^\star=1/n$, for all $i=1,2,\ldots,n$. From the interpretation in (8), we can also see that if $\boldsymbol{\phi}^\star$ is a solution of (8), any $\boldsymbol{\phi}^\star+c\mathbf{1}_n$, for any $c\in\mathbb{R}$, is also a solution of (8), because $\mathbf{1}_n$ belongs to the nullspace of the matrix $\boldsymbol{D}$.

It is known that every limit point of the gradient descent method (7) minimizes $g(\boldsymbol{\phi})$ whenever ∇g is Lipschitz continuous, i.e., there is an $L>0$ such that $\|\nabla g(\boldsymbol{y})-\nabla g(\boldsymbol{x})\|_2\leq L\|\boldsymbol{y}-\boldsymbol{x}\|_2$ for all $\boldsymbol{x}$, $\boldsymbol{y}\in\mathbb{R}^n$, and $\beta\in(0,2/L)$ [4, Prop.1.2.3]. In our case, g is twice differentiable with an Hessian matrix

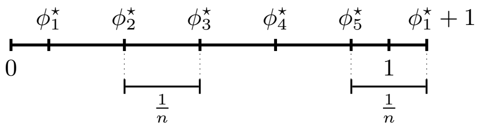


Figure 3: Illustration of the optimality conditions in (13)-(14) **for problem** (8).

equal to $\boldsymbol{D}^T\boldsymbol{D}$. Therefore, any $L \geq \lambda_{\max}(\boldsymbol{D}^T\boldsymbol{D})$ is a valid Lipschitz constant, where $\lambda_{\max}(\cdot)$ is the maximum eigenvalue of a matrix. From (12), we see that $\boldsymbol{D}^T\boldsymbol{D}$ coincides with the Laplacian matrix of the ring graph and, thus, its eigenvalues are given by $2 - 2\cos(2\pi k/n)$, $k = 1, \ldots, n$ [43, Lemma 2.4.4]. The maximum magnitude of these eigenvalues is equal to 4, when n is even, and is smaller than 4, when n is odd. Hence, we can set $L = 4$. It follows that every limit point of (6) is a solution of (8) whenever $\beta \in (0, 1/2)$, that is, $\alpha \in (0, 1)$.

Fast-Desync: An important advantage of viewing Desync as an optimization algorithm is that we can derive faster primitives based on more efficient optimization algorithms. For example, Nesterov's fast gradient algorithm [30,31] solves the same set of problems as the gradient method does, but exhibits a much faster convergence rate. It consists of (we use the version in [46]):

$$\boldsymbol{\phi}^{(k)} = \boldsymbol{\mu}^{(k-1)} - \beta\nabla g(\boldsymbol{\mu}^{(k-1)}) \tag{15a}$$

$$\boldsymbol{\mu}^{(k)} = \boldsymbol{\phi}^{(k)} + \frac{k-1}{k+2}\big(\boldsymbol{\phi}^{(k)} - \boldsymbol{\phi}^{(k-1)}\big), \tag{15b}$$

where $\boldsymbol{\mu} \in \mathbb{R}^n$ is an auxiliary vector. It is known the sequences $\{\boldsymbol{\phi}^{(k)}\}$ and $\{\boldsymbol{\mu}^{(k)}\}$ produced by (15) converge to the same solution of (8), whenever ∇g is Lipschitz continuous with constant L and $\beta \in (0, 1/L]$. Notice that the range of β is smaller than in the gradient method. However, at the expense of small extra computation and memory, Nesterov's method (15) takes $O(1/\sqrt{\epsilon})$ iterations to produce a point $\overline{\boldsymbol{\phi}}$ satisfying $g(\overline{\boldsymbol{\phi}}) - g(\boldsymbol{\phi}^\star) \leq \epsilon$, where $\boldsymbol{\phi}^\star$ minimizes g, whereas the gradient method (7) takes $O(1/\epsilon)$ iterations [46]. We will use this faster version of Desync, which we call Fast-Desync, in our implementation of the multichannel protocol presented in the next section.

4. MULTICHANNEL COORDINATION

We now propose a new *joint synchronization-desynchronization* (Sync-Desync) algorithm for decentralized multichannel MAC. Prior to describing the proposed update rules, we provide an introduction to the basic concept.

4.1 Proposed Multichannel MAC-layer Coordination

Let a WSN comprise n nodes that are initially randomly distributed in C channels [see Fig. 4(a)]—for example, the $C = 16$ channels of the IEEE 802.15.4 standard [26, 48]. The maximum achievable throughput per node is obtained when the nodes are uniformly distributed across the available channels and a perfect TDMA scheduling is reached at each channel [see Fig. 4(b)]. Note that if the total number of nodes in the network, n, is divisible by C then the protocol will lead to $n_c = \frac{n}{C}$ nodes being present in each channel, alternatively, $n_c = \left\{\left\lfloor\frac{n}{C}\right\rfloor, \left\lceil\frac{n}{C}\right\rceil\right\}$ nodes will be present in each channel, as shown in Fig. 4(b). By considering that each node acts as a pulse-coupled oscillator with a period of T sec-

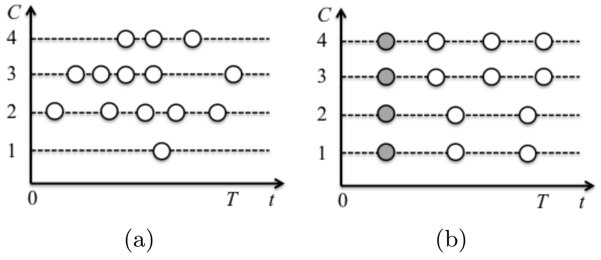


Figure 4: (left) Initial random distribution of 14 nodes in 4 available channels; (right) converged state of the protocol with 3 nodes in channels 1 and 2 and 4 nodes in channels 1 and 2. The figures shows the intra-channel desynchronization and inter-channel synchronization between Desync (white) and Sync (grey) nodes, respectively. The horizontal position of a node indicates the firing moment.

onds, we propose a novel PCO-based mechanism that leads to decentralized multichannel round-robin scheduling without requiring a coordination node or coordination channel. An illustration of the proposed algorithm is given in Fig. 5. To this end, the nodes in each channel are divided in two classes, each with distinguished roles (see Fig. 5). Specifically, all but one node (denoted as "Desync") in each channel apply desynchronization with the aim to achieve TDMA within the channel. Desync nodes (i.e., white nodes in Fig. 5) operate only within their channel, firing and listening to messages from the other nodes in their channel. In addition, one "Sync" node per channel performs cross-channel synchronization. The Sync node (i.e., grey nodes in Fig. 5) of each channel c, $c \in [1, C]$, listens for the Sync fire message in the next channel[2], i.e., channel $c + 1$. A node can be designated as the Sync node in a channel based on a pre-established rule, e.g., the node with the smallest node ID, or the node with the highest battery level (all nodes can be made to report their node ID and battery status in their beacon message broadcasts).

To balance the number of nodes per channel, a balancing scheme also takes place during convergence. In particular, a Sync node lying in channel c may jump to the next channel (with cyclic extension at the border), if it detects that less nodes are present there. Detection is possible by integrating the number of nodes lying in a channel in the fire messages transmitted by the nodes present in the channel. To avoid a race condition, where nodes continuously jump channels, the following conditions are defined for channel jumping:

$$\begin{cases} n_c - n_{c+1} \geq 1, \text{if } c \in [1, C) \\ n_c - n_{c+1} \geq 2, \text{if } c = C \end{cases}$$

where n_c denotes the number of nodes present in channel c, with $n = \sum_{c=1}^{C} n_c$. Note that when a Sync node jumps from one channel to another both channels get into election mode to elect their Sync nodes anew. Based on the described protocol, the network can reach a converged multichannel fair

[2]We consider a cyclic behaviour between channels 1 and 16 of IEEE 802.15.4 [26,48]. Namely, the Sync node at channel 16 listens for the fire message from the Sync node in channel 1.

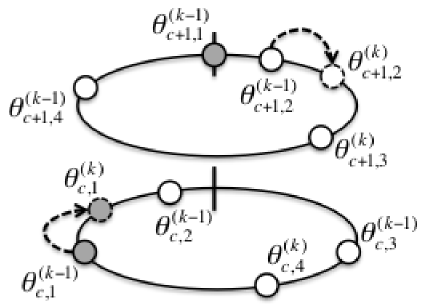


Figure 5: Example of the phase updates performed by the proposed MUCH-SYNC-DESYNC algorithm. Grey/White nodes represent the phase of SYNC/DESYNC nodes. In channel c, the phase of the SYNC node is updated due to the firing of the SYNC node in channel $c+1$. In channel $c+1$ the firing of the SYNC node also triggers a phase update of the DESYNC node 2.

TDMA scheduling, where *(i)* the same number of nodes is present at neighboring channels—if n is divisible by C then $n_c = \frac{n}{C}$ nodes are present in each channel, alternatively, the $n_c = \left\{\left\lfloor\frac{n}{C}\right\rfloor, \left\lceil\frac{n}{C}\right\rceil\right\}$ nodes are present per channel—, *(ii)* the nodes in each channel have converged to a TDMA scheduling and *(iii)* the nodes in adjacent channels (with the same number of nodes) have a parallel TDMA scheduling, where nodes allocated with the same time-slot order transmit synchronously [see Fig. 4(b)].

Cross-channel synchronization in the converged state, which is provided by the SYNC nodes, enables for a swapping mechanism to kick in. Specifically, nodes (both of SYNC and DESYNC type) in adjacent channels can swap channels and time-slots in pairs using a simple RQ/ACK scheme[3]. Channel swapping allows for communication between nodes initially present in different channels.

If nodes join or leave the network, all remaining nodes adjust their beacon packet timings spontaneously, in order to converge to multichannel TDMA anew. As such, the proposed mechanism leads to decentralized multichannel coordination of the WSN nodes from a random initial state, without the need for a coordination node or channel. Once convergence is achieved, the only overhead in the proposed framework is due to the fire messages, which, however, are very short packets compared to the payload data.

4.2 Proposed Joint Sync-Desync Primitive

To describe the joint SYNC-DESYNC primitive that we propose, denote the phase of node $i = 1, \ldots, n_c$ in channel $c = 1, \ldots, C$ with $\theta_{c,i}$ and the corresponding phase offset with $\phi_{c,i}$. Without loss of generality, we will assume that the SYNC node in each channel is node 1; all the remaining nodes are DESYNC nodes. Our goal is to create a primitive that converges to a DESYNC state among the nodes in the same channel and to a SYNC state among the first nodes of each channel. Inspired by Proposition 1, we formulate this problem as

$$\underset{\boldsymbol{\phi_1},\ldots,\boldsymbol{\phi_C}}{\text{minimize}}\ h(\boldsymbol{\phi_1},\ldots,\boldsymbol{\phi_C}) := \frac{1}{2}\sum_{c=1}^{C}\left\|\boldsymbol{D_c}\boldsymbol{\phi_c} - \frac{1}{n_c}\mathbf{1}_{n_c} + \boldsymbol{e_c}\right\|_2^2 + \frac{1}{2}\sum_{c=1}^{C}\left(\boldsymbol{w_{c+1}}^T\boldsymbol{\phi}_{c+1} - \boldsymbol{w_c}^T\boldsymbol{\phi}_c\right)^2, \quad (16)$$

where $\boldsymbol{\phi_c} = (\phi_{c,1}, \phi_{c,2}, \ldots, \phi_{c,n_c}) \in \mathbb{R}^{n_c}$ contains the phase offsets of all nodes of channel c, $\boldsymbol{D_c}$ is the same matrix as (9), but with dimensions $n_c \times n_c$, $\boldsymbol{e_c} = (0, 0, \ldots, 1) \in \mathbb{R}^{n_c}$, and $\boldsymbol{w_c} = (1, 0, \ldots, 0) \in \mathbb{R}^{n_c}$. The objective of (16) has two terms: the first one imposes desynchronization among all the nodes of the same channel; the second one imposes synchronization among the first nodes of consecutive channels. The second term is inspired by optimization-based approaches to the average consensus problem [18, 28].

Although we use FAST-DESYNC in our implementation, for simplicity, we derive here the multichannel primitive using the DESYNC version in (6). Given that the partial gradient of h with respect to $\boldsymbol{\phi_c}$ is

$$\nabla_{\boldsymbol{\phi_c}} h(\boldsymbol{\phi_1},\ldots,\boldsymbol{\phi_C}) = \boldsymbol{D_c}^T\boldsymbol{D_c}\boldsymbol{\phi_c} + \boldsymbol{d_c} + \left(2\boldsymbol{w_c}^T\boldsymbol{\phi_c} - \boldsymbol{w_{c-1}}^T\boldsymbol{\phi_{c-1}} - \boldsymbol{w_{c+1}}^T\boldsymbol{\phi_{c+1}}\right)\boldsymbol{w_c}, \quad (17)$$

where $\boldsymbol{d_c} := (1, 0, \ldots, 0, -1) \in \mathbb{R}^{n_c}$, it can be easily shown that the gradient descent algorithm applied to (16) with step size $\beta = \alpha/2$ yields

$$\phi_{c,i}^{(k)} = (1-2\alpha)\phi_{c,i}^{(k-1)} + \frac{\alpha}{2}(\phi_{c,i-1}^{(k-1)} + \phi_{c,i+1}^{(k-1)} + \phi_{c-1,i}^{(k-1)} + \phi_{c+1,i}^{(k-1)}), \quad (18)$$

for $i = 1$, and

$$\phi_{c,i}^{(k)} = (1-\alpha)\phi_{c,i}^{(k-1)} + \frac{\alpha}{2}(\phi_{c,i-1}^{(k-1)} + \phi_{c,i+1}^{(k-1)} + (\boldsymbol{d_c})_i), \quad (19)$$

for $i \neq 1$. Note that (19) is similar to the phase update in (5) for DESYNC. Although the application of the gradient descent to (16) leads to a very natural algorithm, the updates for the SYNC nodes in (18) fail to satisfy the coupling rules we described in Section 4.1. Namely, a direct implementation of (18) in a wireless transceiver requires each SYNC node to listen for fire messages at the channel where it resides, as well as in the previous and subsequent channels. This makes the implementation of (18)-(19) impractical if we use conventional half-duplex transceiver hardware in IEEE 802.15.4-based WSNs. This issue is caused by the symmetry of the Hessian matrix of h, and cannot be solved, at least in a straightforward way, with optimization-based techniques. We tackle this problem by modifying the matrix associated with the iterations (18)-(19), exploring the fact that there is one degree of freedom in each channel.

Multichannel Sync-Desync (MuCh-Sync-Desync): Without loss of generality, assume each channel has $n := n_1 = n_2 = \cdots = n_C$ nodes. We propose iterating

$$\begin{bmatrix} \boldsymbol{\phi_1}^{(k)} \\ \boldsymbol{\phi_2}^{(k)} \\ \vdots \\ \boldsymbol{\phi_C}^{(k)} \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{Q_1} & \boldsymbol{Q_2} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{Q_1} & \boldsymbol{Q_2} & \cdots & \mathbf{0} \\ \vdots & & \ddots & & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{Q_1} & \boldsymbol{Q_2} \\ \boldsymbol{Q_2} & \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{Q_1} \end{bmatrix}}_{=:\boldsymbol{M}} \begin{bmatrix} \boldsymbol{\phi_1}^{(k-1)} \\ \boldsymbol{\phi_2}^{(k-1)} \\ \vdots \\ \boldsymbol{\phi_C}^{(k-1)} \end{bmatrix}$$

[3]Swap-channel RQ or ACK packets can be transmitted at another channel during a short interval after and/or before a fire message transmission at a node's channel.

$$+\beta \underbrace{\begin{bmatrix} \boldsymbol{e}_n \\ \boldsymbol{e}_n \\ \vdots \\ \boldsymbol{e}_n \end{bmatrix}}_{=:\boldsymbol{b}}, \quad (20)$$

where $\mathbf{0}$ is the $n \times n$ zero matrix, $\boldsymbol{e}_n := (0, 0 \ldots, 0, 1) \in \mathbb{R}^n$, $\boldsymbol{Q_2} := \mathrm{Diag}(\gamma, 0, \ldots, 0) \in \mathbb{R}^{n \times n}$, $0 < \gamma < 1$, and $\boldsymbol{Q_1}$ is the $n \times n$ matrix

$$\boldsymbol{Q_1} := \begin{bmatrix} 1-\gamma & 0 & 0 & 0 & \cdots & 0 & 0 \\ \beta & 1-2\beta & \beta & 0 & \cdots & 0 & 0 \\ 0 & \beta & 1-2\beta & \beta & \cdots & 0 & 0 \\ \vdots & & & \ddots & & \vdots & \vdots \\ \beta & 0 & 0 & 0 & \cdots & \beta & 1-2\beta \end{bmatrix}.$$

According to (20), all but the first node in channel c, i.e., $i \neq 1$, perform (19), whereas node 1 performs

$$\phi_{c,1}^{(k)} = (1-\gamma)\phi_{c,1}^{(k-1)} + \gamma\, \phi_{c+1,1}^{(k-1)}, \quad (21)$$

which, in contrast with (18), is amenable to a practical implementation. Recall that the SYNC node in channel c updates its phase at the firing of the SYNC node in channel $c+1$, i.e., at $t_{c+1,i}$, where $\theta_{c+1,i}(t_{c+1,i}) = 1$. It can be shown that (21) corresponds to

$$\theta'_{c,1}(t_{c+1,1}) = (1-\gamma)\theta_{c,1}(t_{c+1,1}) + \gamma \mod 1, \quad (22)$$

in terms of phase updates. Since $0 \leq \theta_{c,1}(t) \leq 1$ and $0 < \gamma < 1$, (22) exhibits inhibitory coupling [19] between the SYNC nodes in subsequent channels, thereby leading to synchronization of their phases. Note that the iteration in (20) can be straightforwardly extended to the case where the channels have different number of nodes; in this case, the sizes of vectors $\boldsymbol{\phi}$, $\boldsymbol{e}$, and matrices $\boldsymbol{Q_1}$ and $\boldsymbol{Q_2}$ vary per channel $c = 1, \ldots, C$ but their format is the same. Still, the update equations, described in (19) and (21) remain the same independently of the number of nodes present in each channel.

PROPOSITION 2. *The sequence produced by* (20) *converges to a solution of* (16) *whenever* $0 < \gamma < 1$ *and* $0 < \beta < 1/2$.

PROOF SKETCH. The proof is divided into two steps. The first step is the simplest one: using reasoning analogous to the one in Fig. 3, it can be shown that any fixed point of (20) is a solution of problem (16). The second step, which is more complex, consists of showing that the sequence produced by (20) converges (to a fixed point). What makes the second step hard is that the matrix $\boldsymbol{M}$ has a spectral radius $\rho(\boldsymbol{M})$ equal to 1. In fact, it can be checked by inspection that the vector of ones, $\mathbf{1}_{nC}$, is a right eigenvector of $\boldsymbol{M}$ associated to the eigenvalue 1, and $\boldsymbol{u} := (\boldsymbol{e}_1, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_1) \in (\mathbb{R}^n)^C$ is a left eigenvector of $\boldsymbol{M}$ also associated to the eigenvalue 1. If $\rho(\boldsymbol{M})$ were strictly less than 1, than it would follow from [22, §1.2] that (20) converged to $(\boldsymbol{I} - \boldsymbol{M})^{-1}\boldsymbol{b}$. Also note that Perron-Frobenius theory is not applicable, because $\boldsymbol{M}$ is not irreducible. Yet, it is possible to compute the eigenvalues of $\boldsymbol{M}$ in closed-form, as explained in the next paragraph, and conclude that $\rho(\boldsymbol{M}) = 1$ has algebraic (and thus geometric) multiplicity 1. This implies that we can write (20) as

$$\boldsymbol{\phi}^{(k)} = \overline{\boldsymbol{M}}\boldsymbol{\phi}^{(k-1)} + \mathbf{1}_{nC}\boldsymbol{u}^T\boldsymbol{\phi}^{(k-1)} + \boldsymbol{b}, \quad (23)$$

where $\overline{\boldsymbol{M}} := \boldsymbol{M} - \mathbf{1}_{nC}\boldsymbol{u}^T$ has spectral radius strictly smaller than 1, i.e., $\rho(\overline{\boldsymbol{M}}) < 1$ [20, Lemma 8.2.7]. Then, using the fact that $\boldsymbol{u}^T\boldsymbol{\phi}^{(k)} = \boldsymbol{u}^T\boldsymbol{\phi}^{(k-1)} = \cdots = \boldsymbol{u}^T\boldsymbol{\phi}^{(1)} = \boldsymbol{u}^T\boldsymbol{\phi}^{(0)}$, we can rewrite (23) as

$$\boldsymbol{\phi}^{(k)} = \overline{\boldsymbol{M}}\boldsymbol{\phi}^{(k-1)} + \overline{\boldsymbol{b}}, \quad (24)$$

where $\overline{\boldsymbol{b}} = \boldsymbol{b} + \mathbf{1}_{nC}\boldsymbol{u}^T\boldsymbol{\phi}^{(0)}$. Since $\rho(\overline{\boldsymbol{M}}) \leq 1$, it follows from [22, §1.2] that (24) converges to $(\boldsymbol{I} - \overline{\boldsymbol{M}})^{-1}\overline{\boldsymbol{b}} = (\boldsymbol{I} - \overline{\boldsymbol{M}})^{-1}\boldsymbol{b} + (\boldsymbol{I} - \overline{\boldsymbol{M}})^{-1}\mathbf{1}_{nC}\boldsymbol{u}^T\boldsymbol{\phi}^{(0)}$, a well-defined and unique fixed point of (20).

To compute the eigenvalues of $\boldsymbol{M}$, we first find a permutation matrix $\boldsymbol{P}$ that reorders the nodes such that the first nodes of each channel correspond to the last coordinates of the new coordinate system, i.e., $\boldsymbol{\phi}$ is mapped onto

$$(\phi_{1,2}, \phi_{1,3}, \ldots, \phi_{1,n}, \phi_{2,2}, \phi_{2,3}, \ldots, \phi_{2,n}, \ldots, \\ \phi_{1,1}, \phi_{2,1}, \ldots, \phi_{C,1}).$$

The matrices $\boldsymbol{M}$ and $\boldsymbol{P}^T\boldsymbol{M}\boldsymbol{P}$ have the same eigenvalues, but $\boldsymbol{P}^T\boldsymbol{M}\boldsymbol{P}$ is block-upper triangular. Therefore, the eigenvalues of $\boldsymbol{P}^T\boldsymbol{M}\boldsymbol{P}$ are the eigenvalues of the blocks in its diagonal (counting multiplicity), which can be computed in closed-form. In particular, the first $n-1$ diagonal blocks of $\boldsymbol{P}^T\boldsymbol{M}\boldsymbol{P}$ are tridiagonal Toeplitz matrices and the last block is a circulant matrix. $\square$

As mentioned before, we can modify (20) to make the nodes in each channel perform FAST-DESYNC. This yields a primitive that we call FAST-MUCH-SYNC-DESYNC and whose performance is assessed in the next section.

5. EXPERIMENTAL RESULTS

5.1 Simulation Results

Simulation results on the proposed MUCH-SYNC-DESYNC algorithm and its fast version (denoted as FAST-MUCH-SYNC-DESYNC) were obtained using MATLAB. We implemented our algorithms using the event-driven simulator of Degesys *et al.* [12] as basis for our software. We use two different convergence thresholds, namely, $\epsilon = 10^{-3}$ and $\epsilon = 10^{-4}$. Convergence to multichannel TDMA scheduling is reported at the firing round after which successive updates never perturb each node's phase above the predetermined threshold ϵ. We note that our algorithms' updates are performed on the node's phases θ_i. We set $\gamma = 0.6$ and evaluate the convergence speed of the algorithm for various α values, ranging for $\alpha = 0.05$ to $\alpha = 0.95$, with a stepsize of 0.05. All simulations were repeated 400 times and average results are reported.

The results are given in Fig. 6(a)–(d) for $n_c = 4$ nodes per channel in $C = 4, 6, 10,$ and 16 channels, respectively. The results show that for various values of α both proposed algorithms (namely, the simple and the fast version) converge. The convergence of both algorithms is also independent of the number on available channels in the network.

It is worth noticing that the proposed FAST-MUCH-SYNC-DESYNC algorithm offers a notable convergence speed-up (i.e., 6.01%–42.54%) with respect to the simple MUCH-SYNC-DESYNC algorithm, irrespective of the number of available channels. Furthermore, the convergence speed-up increases when a strict threshold ($\epsilon = 10^{-4}$) is used, which is important to guarantee strict convergence to cross-channel synchronization as well as perfect TDMA scheduling in each channel. The improvement is more significant at low and

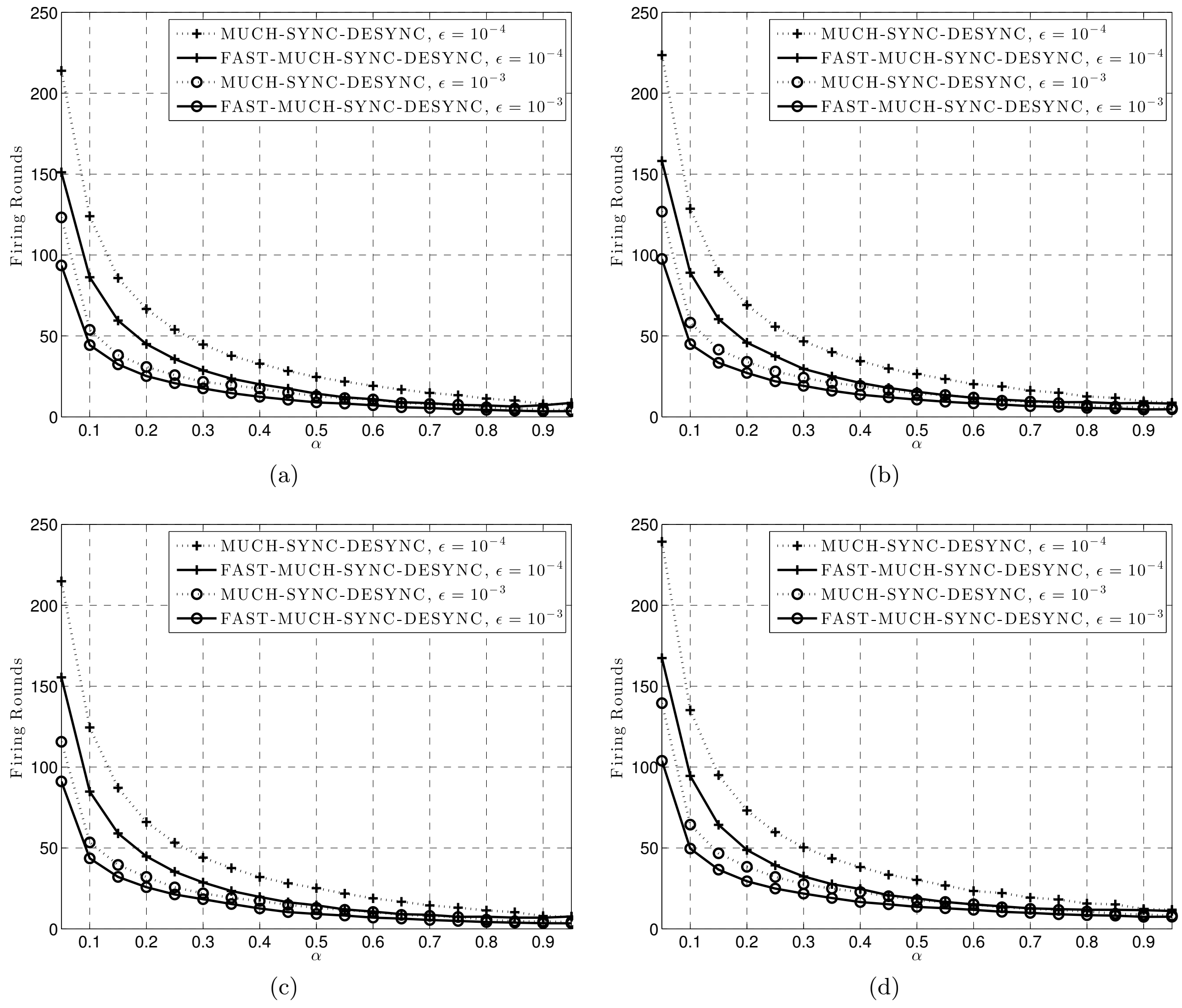


Figure 6: Average number of firing rounds for convergence to decentralized multichannel TDMA scheduling for the proposed MuCh-Sync-Desync algorithm and its fast counterpart; $n_c = 4$ nodes per channel are considered with: (a) $C = 4$, (b) $C = 6$, (c) $C = 10$ and (d) $C = 16$ channels.

medium values of α, which are typically used in practice to attenuate noise in fire message transmissions.

5.2 Experiments with TelosB Motes

5.2.1 Experimental setup

We implemented the proposed MuCh-Sync-Desync and its Fast version, described in Section 4.1, as applications in the Contiki 2.7 operating system running on TelosB motes. By utilizing the `NullMAC` and `NullRDC` network stack options in Contiki, we control all node interactions at the MAC layer via our code. Moreover, by utilizing the TelosB high-resolution timer (`rtimer` library), we can achieve the scheduling of transmission and listening events with sub-millisecond accuracy. Our implementation follows the description of Section 4.2 and performs the iterations of (20) with T = 100 ms and $\alpha = \gamma = 0.6$. Initially, all nodes listen constantly until convergence is achieved in their channel. When convergence in the steady network state is achieved, data transmission is initiated and nodes switch to limited listening so as to conserve energy. Therefore, the use of MuCh-Sync-Desync or its Fast version influence the speed of convergence and not the bandwidth of the network. Due to interference in the unlicensed 2.4 GHz band of IEEE 802.15.4 and timing uncertainties in the fire message broadcast and reception, we apply the following practical modifications. These modifications ensure that, once the network reaches the steady state, it remains there indefinitely until the entire network operation is suspended, or nodes join or leave the network.

Firstly, each node can transmit data in-between its own fire message and the subsequent fire message from another node, albeit allowing for a *guard time* of 6 ms before and after the anticipated beacon broadcast times. In this way, no collisions occur between data and fire message packets.

Secondly, in the steady state, each node turns its transceiver on solely for the 12 ms guard time corresponding to the beacon message they expect to receive. Moreover, all nodes switch to "sparse listening", that is, they listen for beacons only once every eight periods, unless high interference noise is detected. In the converged state, each node determines the interference noise floor in-between transmissions by reading the CC2420 RSSI register. If high interference is detected, the node switches to regular listening. Thus, the option of sparse listening does not affect the stability of MuCh-Sync-Desync.

Thirdly, to remain in sparse listening (that is, to avoid unnecessarily switching to prolonged listening, which is energy consuming) and avoid interrupting data transmission due to transient interference, all nodes are set to switch to full listening only if they do not receive $N_c = 10$ consecutive fire messages. Our choice of N_c provides stable operation under interference, albeit at the cost of slower convergence and reaction time.

As mentioned in Section 4.1, once all nodes are activated, they are first balanced across the available channels based on the TFDMA balancing algorithm of Buranapanichkit and Andreopoulos [7]. Namely, each node can switch to another channel if it detects less nodes are present therein by sending a switch message in its original channel.

5.2.2 Comparison against TSCH

We select TSCH as benchmark for our comparisons, since it is a state-of-the-art MAC protocol for densely-connected WSNs [47, 49]. Our implementation of TSCH follows the 6tisch simulator and TSCH standard [21,47,48]. Specifically, channel 11 of IEEE 802.15.4 was used for advertisements, the RQ/ACK ratio was set to $\frac{1}{9}$, the slotframe comprised 101 slots of 15 ms each, and one node (at the center of our deployment) was set to broadcast the slotframe beacon for global time synchronization. Finally, the network under TSCH is deemed as converged to the steady state when 5% or less of the timeslots changed within the last 10 slotframes.

We deployed $n = 64$ nodes in the $C = 16$ channels of IEEE 802.15.4. This leads to $n_c = 4$ nodes per channel after balancing and converging to the steady state. The 64 TelosB motes were placed in four neighboring rooms on the same floor of an office building, with each room containing 16 nodes. Our setup corresponds to scenarios involving dense network topologies and data-intensive communications (e.g., visual sensor network applications [15, 17]).

5.2.3 Power dissipation results

We assessed the average power dissipation of our scheme against TSCH by placing selected TelosB motes in series with a high-tolerance 1-Ohm resistor and by utilizing a high-frequency oscilloscope to capture the current flow through the resistor in real time. During this power dissipation experiment, no other devices (or interference signal generators) operating in the 2.4 GHz band were present in the area. Average results over 5 min of operation are reported. The average power dissipation of MuCh-Sync-Desync without transmitting or receiving data payload was measured to be 1.58 mW. The average power dissipation of a TSCH node under minimal payload (128 bytes per 4 s) was found to be 1.64 mW, which is very close to the value that has been independently reported by Vilajosana *et al.* [47]. Therefore, under the same setup, our proposal and TSCH were found to incur comparable power dissipation for their operation.

5.2.4 Convergence speed results

We investigate the convergence time of MuCh-Sync-Desync, Fast-MuCh-Sync-Desync and TSCH under varying interference levels. Rapid convergence to the steady state is very important when the WSN is initiated from a suspended state, or when sudden changes happen in the network (e.g., nodes join or leave).

We carried out 100 independent tests, with each room containing an interference generator for 25 tests. To generate interference, an RF signal generator was used to create an unmodulated carrier in the center of each WSN channel. The carrier amplitude was adjusted to alter the signal-to-noise-ratio (SNR) at each receiver [5]. The nodes were set to maximum transmit power (+0 dBm) in order to operate under the best SNR possible.

Fig 7 shows the time required for MuCh-Sync-Desync, Fast-MuCh-Sync-Desync and TSCH to converge under varying interfering signal power levels. The results obtained with our Contiki implementation corroborate that our proposal reduces the convergence time by an order of magnitude in comparison to TSCH and that the Nesterov-based modification offers 36.48%-41.07% increased convergence speed under a realistic setup. Moreover, the difference in convergence time between the proposed mechanism and TSCH increases with the interference level because TSCH nodes

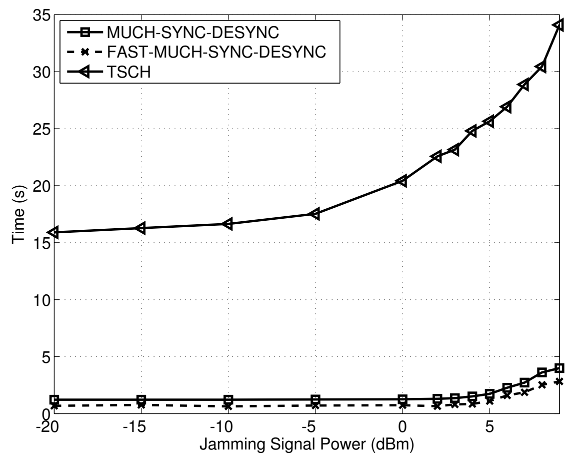


Figure 7: Average time required for MuCh-Sync-Desync, its Fast version, and TSCH to converge under various interference levels.

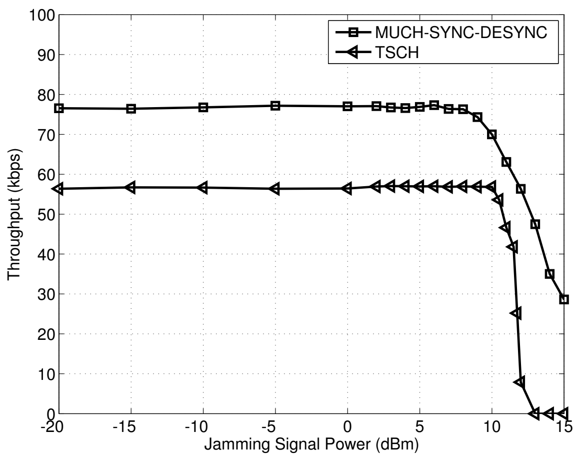


Figure 8: Total network throughput between MuCh-Sync-Desync and TSCH under varying signal power levels.

Table 1: Average Convergence Time (in seconds) Under Hidden Nodes. Numbers in Parenthesis Show the Convergence time of the Fast (Nesterov-based) version of our proposal.

	MuCh-Sync-Desync	TSCH
Without Hidden Nodes	1.1356 (0.7351)	15.5845
With Hidden Nodes	1.8514 (1.2896)	15.2957

miss most of the advertisement RQ/ACK messages in the advertisement (control) channel. This result demonstrates the key advantages of our decentralized MAC mechanism with respect to TSCH, namely: *(i)* it is fully decentralized and *(ii)* it does not depend on an advertisement and acknowledgement scheme.

5.2.5 Results under hidden nodes

We now investigate the robustness and convergence speed of our scheme when some nodes in the WSN are hidden from other nodes. We measure the time to achieve convergence to steady state when a random subset of 20 nodes in our WSN setup was programmed to ignore transmissions from 4 randomly chosen nodes. The results in Table 1 show that, irrespective of the presence of hidden nodes, the convergence of MuCh-Sync-Desync and its fast version is an order-of-magnitude faster than that of TSCH. When hidden nodes are present, the required convergence time of MuCh-Sync-Desync (resp. its Fast version) increases by 63.03% (resp. 75.43%), while that of TSCH is actually sightly decreased by 2.13%. This is to be expected, as TSCH nodes simply ignore RQ packets from hidden nodes. Conversely, due to the Desync (resp. Fast-Desync) process within each channel, applied by MuCh-Sync-Desync (resp. its Fast version), prolonged beaconing will take place until all hidden nodes are placed amongst non-hidden Desync phase neighbors. This spontaneous robustness of MuCh-Sync-Desync (and its Fast version) to hidden nodes is an interesting property that deserves further study. For instance, one can try to determine conditions that guarantee that no configuration of hidden nodes can lead to instability.

5.2.6 Bandwidth results

We measure the total network throughput (i.e., total payload bits transmitted by all nodes per sec) achieved with MuCh-Sync-Desync and TSCH under various interference levels. As stated in Section 5.2.1, both the proposed MuCh-Sync-Desync or its Fast version achieve the same network throughput, as data transmission is initiated only after convergence to the steady network state (that is, perfect TDMA scheduling per channel). The results in Fig 8 show that MuCh-Sync-Desync systematically achieves substantially higher network throughput (more than 40% increase w.r.t. TSCH), irrespective of the interference level. Both protocols suffer a significant throughput loss of under high interference (i.e., above 10 dBm), which is, however, substantially more severe for TSCH. In effect, when interference is above 12 dBm, the bandwidth obtained with TSCH drops to zero because of the inability to recover lost slots through advertising. Conversely, even under high interference levels, MuCh-Sync-Desync recuperates bandwidth utilization due to the elasticity of Sync and Desync mechanisms and the high value used for N_c.

6. CONCLUSION

In this paper, we see Desync, a well-known desynchronization method for the medium access control layer of wireless sensor networks (WSNs), as the gradient method, which is used in convex optimization. Specifically, we proposed MuCh-Sync-Desync, a novel completely decentralized and collision-free multichannel MAC algorithm for WSNs. Moreover, we proposed a new fast version of this algorithm, which is based on the Nesterov's modification of the gradient method. Fast-MuCh-Sync-Desync was shown to systematically improve the convergence speed of the simple algorithm, irrespective of the number of available channels in the network. In addition, using our WSN deployment, we compared our MuCh-Sync-Desync algorithm and its fast counterpart against time-synchronized channel hopping (TSCH), which is included in the IEEE 802.15.4e-2012 standard. Our algorithms were experimentally proven to provide for an order-of-magnitude decrease in the convergence time to the network steady state and more than 40% increase in the total network throughput. Finally, the proposed algorithms offer significantly-increased robustness to interference and hidden nodes in the network, while requiring comparable power dissipation and without requiring a coordination node or channel.

7. ACKNOWLEDGMENTS

We wish to thank Prof. Andrea Gasparri for his comments on the final version of our manuscript.

8. REFERENCES

[1] M. S. A. Mutazono and M. Murata. Energy efficient self-organizing control for wireless sensor networks inspired by calling behavior of frogs. *Computer Communications*, 2011.

[2] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury. A survey on wireless multimedia sensor networks. *Computer networks*, 51(4):921–960, 2007.

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

[4] D. P. Bertsekas. Nonlinear programming. 1999.

[5] C. A. Boano, T. Voigt, C. Noda, K. Romer, and M. Zúñiga. Jamlab: Augmenting sensornet testbeds with realistic and controlled interference generation. In *Int. Conf. on Information Processing in Sensor Networks (IPSN)*, pages 175–186, 2011.

[6] I. Bojic, V. Podobnik, I. Ljubi, G. Jezic, and M. Kusek. A self-optimizing mobile network: Auto-tuning the network with firefly-synchronized agents. *Information Sciences*, 182(1):77–92, 2012.

[7] D. Buranapanichkit and Y. Andreopoulos. Distributed time-frequency division multiple access protocol for wireless sensor networks. *IEEE Wirel. Comm. Lett.*, 1(5):440–443, Oct. 2012.

[8] D. Buranapanichkit, N. Deligiannis, and Y. Andreopoulos. On the stochastic modeling of desynchronization convergence in wireless sensor networks. In *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pages 5045–5049, 2014.

[9] D. Buranapanichkit, N. Deligiannis, and Y. Andreopoulos. Convergence of desynchronization primitives in wireless sensor networks: A stochastic modeling approach. *IEEE Trans. Signal Process.*, 63(1):221–233, 2015.

[10] S. Choochaisri, K. Apicharttrisorn, K. Korprasertthaworn, P. Taechalertpaisarn, and C. Intanagonwiwat. Desynchronization with an artificial force field for wireless networks. *ACM SIGCOMM Computer Communication Review*, 42(2):7–15, 2012.

[11] D. De Guglielmo, G. Anastasi, and M. Conti. A localized de-synchronization algorithm for periodic data reporting in ieee 802.15. 4 wsns. In *IEEE Symposium on Computers and Communications (ISCC)*, pages 605–610, 2012.

[12] J. Degesys and R. Nagpal. Towards desynchronization of multi-hop topologies. In *Proc. IEEE Int. Conf. Self-Adaptive and Self-Organizing Syst. (SASO)*, pages 129–138, 2008.

[13] J. Degesys, I. Rose, A. Patel, and R. Nagpal. Desync: self-organizing desynchronization and tdma on wireless sensor networks. In *Int. Conf. on Information Processing in Sensor Networks (IPSN)*, pages 11–20, 2007.

[14] M. DeGroot. Reaching a consensus. *J. American Statistical Association*, 69(345):118–121, 1974.

[15] N. Deligiannis, F. Verbist, J. Barbarien, J. Slowack, R. Van de Walle, P. Schelkens, and A. Munteanu. Distributed coding of endoscopic video. In *IEEE Int. Conf. Image Process. (ICIP)*, pages 1813–1816, 2011.

[16] N. Deligiannis, F. Verbist, A. C. Iossifides, J. Slowack, R. Van de Walle, P. Schelkens, and A. Munteanu. Wyner-ziv video coding for wireless lightweight multimedia applications. *EURASIP J. Wireless Commun. Netw.*, 2012(1):1–20, 2012.

[17] N. Deligiannis, F. Verbist, J. Slowack, R. v. d. Walle, P. Schelkens, and A. Munteanu. Progressively refined wyner-ziv video coding for visual sensors. *ACM Trans. Sensor Netw.*, 10(2):21, 2014.

[18] T. Erseghe, D. Zennaro, E. Dall'Anese, and L. Vangelista. Fast consensus by the alternating direction multipliers method. *IEEE Trans. Signal Process.*, 59(11):5523–5537, 2011.

[19] Y.-W. Hong and A. Scaglione. A scalable synchronization protocol for large scale sensor networks and its applications. *IEEE J. Sel. Areas Commun.*, 23(5):1085–1099, 2005.

[20] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge university press, 2012.

[21] IEEE 802.15.4e-2012. IEEE Standard for Local and Metropolitan Area Networks. Part 15.4: Low-Rate Wireless Personal Area Networks (LRWPANs) Amendment 1: MAC Sublayer. *IEEE Std.*, Apr. 2012.

[22] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, Philadelphia, 1995.

[23] J. Klinglmayr and C. Bettstetter. Self-organizing synchronization with inhibitory-couples oscillaotrs: convergence and robustness. *ACM Trans. on Autonomous and Adaptive Systems*, 7(3), Sept. 2012.

[24] R. Leidenfrost and W. Elmenreich. Firefly clock synchronization in an 802.15.4 wireless network. *EURASIP J. Embed. Syst.*, 2009.

[25] C.-M. Lien, S.-H. Chang, C.-S. Chang, and D.-S. Lee. Anchored desynchronization. In *Proc. IEEE INFOCOM'12*, pages 2966–2970, 2012.

[26] G. Lu, B. Krishnamachari, and C. Raghavendra. Performance evaluation of the IEEE 802.15. 4 MAC for low-rate low-power wireless networks. In *IEEE Internat. Conf. on Perf., Comput., and Comm.*, pages 701–706, 2004.

[27] R. E. Mirollo and S. H. Strogatz. Synchronization of pulse-coupled biological oscillators. *SIAM Journal on Applied Mathematics*, 50(6):1645–1662, 1990.

[28] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Puschel. D-ADMM: A communication-efficient distributed algorithm for separable optimization. *IEEE Trans. Signal Process.*, 61(10):2718–2723, 2013.

[29] A. Motskin, T. Roughgarden, P. Skraba, and L. Guibas. Lightweight coloring and desynchronization for networks. In *IEEE INFOCOM'09*, pages 2383–2391, 2009.

[30] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[31] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2004.

[32] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[33] R. Pagliari, Y.-W. P. Hong, and A. Scaglione. Bio-inspired algorithms for decentralized round-robin and proportional fair scheduling. *IEEE J. on Select. Areas in Commun.*, 28(4):564–575, May 2010.

[34] R. Pagliari and A. Scaglione. Scalable network synchronization with pulse-coupled oscillators. *IEEE Trans. Mobile Comput.*, 10(3):392–405, 2011.

[35] A. Patel, J. Degesys, and R. Nagpal. Desynchronization: The theory of self-organizing algorithms for round-robin scheduling. *Proc. IEEE Int. Conf. Self-Adaptive and Self-Organizing Syst. (SASO)*, july 2007.

[36] C. S. Peskin. *Mathematical aspects of heart physiology.* Courant Institute of Mathematical Sciences, New York University, 1975.

[37] Y.-W. Peter Hong, A. Scaglione, and R. Pagliari. Pulse coupled oscillators' primitive for low complexity scheduling. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pages 2753–2756, 2009.

[38] W. Ren and R. W. Beard. *Distributed consensus in multi-vehicle cooperative control.* Springer, 2008.

[39] A. Sarlette and R. Sepulchre. Synchronization on the circle. http://arxiv.org/abs/0901.2408, 2009.

[40] O. Simeone, U. Spagnolini, Y. Bar-Ness, and S. H. Strogatz. Distributed synchronization in wireless networks. *IEEE Signal Process. Mag.*, 25(5):81–97, Sep. 2008.

[41] G. Smart, N. Deligiannis, Y. Andreopoulos, R. Surace, V. Loscri, and G. Fortino. Decentralized time-synchronized channel swapping for wireless sensor networks. In *11th European Conference on Wireless Sensor Networks (EWSN'14), poster presentation*, 2014.

[42] J. Song, S. Han, A. K. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt. WirelessHART: Applying wireless technology in real-time industrial process control. In *IEEE Real-Time and Embed. Tech. and Appl. Symp., (RTAS)*, pages 377–386, 2008.

[43] D. Spielman. The Laplacian, 2009. Lecture notes, Spectral Graph Theory, Yale.

[44] A. Tinka, T. Watteyne, and K. Pister. A decentralized scheduling algorithm for time synchronized channel hopping. In *Ad Hoc Netw.*, pages 201–216. 2010.

[45] Y.-C. Tseng, Y.-C. Wang, K.-Y. Cheng, and Y.-Y. Hsieh. iMouse: an integrated mobile surveillance and wireless sensor system. *IEEE Computer*, 40(6):60–66, 2007.

[46] L. Vandenberghe. Gradient method, Spring 2008-09. Lecture Notes, Optimization Methods for Large-Scale Systems (EE-236C), UCLA.

[47] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne, T. Chang, and K. Pister. A realistic energy consumption model for TSCH networks. *IEEE Sensors J.*, 2013.

[48] Q. Wang, X. Vilajosana, and T. Watteyne. 6TSCH operation sublayer (6top). *Internet-Draft, IETF Std., Rev. draft-wang-6tisch-6top-sublayer-00*, Apr. 2014.

[49] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister. Openwsn: a standards-based low-power wireless development environment. *Transactions on Emerging Telecommunications Technologies*, 23(5):480–493, 2012.

[50] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, 53:65–78, 2004.

[51] P. Zand, S. Chatterjea, K. Das, and P. Havinga. Wireless industrial monitoring and control networks: The journey so far and the road ahead. *J. Sens. Actuator Netw.*, 1(2):123–152, 2012.