IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 25, 2018, accepted December 8, 2018, date of publication December 18, 2018,
date of current version January 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2888551

Energy-Efficient Distributed Leader Selection Algorithm for Energy-Constrained Wireless Sensor Networks

SANDER ULP, YANNICK LE MOULLEC, AND MUHAMMAD MAHTAB ALAM
Thomas Johann Seebeck Department of Electronics, Tallinn University of Technology, 19086 Tallinn, Estonia

Corresponding author: Sander Ulp (sander.ulp@taltech.ee)

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 668995 and partly supported by European Union Regional Development Fund in the framework of the Tallinn University of Technology Development Program 2016-2022. This material reflects only the authors view and the EC Research Executive Agency is not responsible for any use that may be made of the information it contains. This research was supported by the Information Technology Foundation for Education.

ABSTRACT In the context of green communication and energy-efficiency in wireless communication, this paper investigates distributed estimation algorithms in an energy-constrained wireless sensor network and proposes an energy-efficient distributed leader selection algorithm. The existing state-of-the-art diffusion algorithm and the recently introduced distributed leader selection algorithm are investigated. To evaluate the energy consumption of the algorithms, their respective algorithmic complexity, and number of operations and information exchanges are derived and compared. The obtained values are used as a basis to estimate the execution time and energy consumption of the algorithms. We propose and introduce the energy-efficient distributed leader selection algorithm which retains the performance of the existing leader selection algorithm while reducing the complexity and energy consumption. For the simulations, the algorithms are mapped to widely used wireless sensor network hardware architectures (MSP430 and RSL10). The numerical results show that the proposed algorithm is able to decrease the energy consumption of the network by 32%–53% and can extend the network lifetime by 14%–46% as compared with the diffusion and the distributed leader selection algorithms.

INDEX TERMS Energy-efficiency, distributed estimation, wireless sensor networks, distributed leader selection, diffusion.

I. INTRODUCTION

Recently, data traffic has had an exponential growth due to the rapid increase of the usage of wireless devices. With new applications on the horizon in the field of Internet-of-Things (IoT), the amount of energy consumed by wireless applications and systems is on the rise. This has put a lot of focus on energy-efficiency (EE) and green communication (GC) systems to reduce the amount of energy consumed [1], [2]. EE in wireless sensor networks (WSN) is an important topic that has received a lot attention due to the numerous applications that WSNs have in IoT [1]–[4]. Many recent works have aimed at reducing the amount of energy consumed by optimizing protocols and clustering on the physical and the network levels [5]–[7]. Similarly, hardware such as microcontrollers (MCUs) and radio modules are being optimized for low-power systems and applications [8]. The above approaches are especially important when the WSN is energy-constrained and the nodes have limited battery capacity to work with. In this work we focus on distributed estimation as one of the applications of WSNs and on the energy-efficiency of the underlying algorithms.

The idea behind distributed estimation is to solve a common task or to estimate some process in a network and by means of cooperation to enhance the performance of the network and of the nodes. Distributed estimation is a popular research topic in various interdisciplinary fields, such as agriculture, military, environmental studies and signal processing [9], [10]. We are interested in fully distributed solutions without any centralized processing unit. The benefits of fully distributed estimation include low energy consumption, low processing complexity and robustness [10].

Methods proposed for solving distributed estimation problems include incremental, consensus, diffusion and leader selection algorithms [10]–[12]. Incremental algorithms suffer

from a NP hard problem as a cyclic route through the network has to be defined for the algorithm to work [10]. Consensus algorithms are outperformed by diffusion algorithms and also have stability issues under infinitesimally small step-sizes [11]. Therefore, in this work we focus on the diffusion algorithm and the distributed leader selection algorithm. Diffusion algorithms have shown good performance as well as robustness [10], [13]. Distributed leader selection (DLS) has been proposed in our earlier work [12]. DLS, while not always able to outperform the diffusion algorithms, has the benefits of being simpler and more robust in terms of the secondary parameter estimation accuracy [14].

When considering a massive number of devices communicating in a distributed fashion in an energy-constrained situation, it is critical to have an extended lifetime for the network and thus, as a first step, we want to estimate the energy cost of the diffusion algorithm and of the DLS algorithm. The energy costs of the algorithms are obtained by using high-level estimation and are based on the number of operations required for the algorithms to run. Thus, we investigate the complexity of the aforementioned algorithms, as well as the required number of operations for both of the algorithms. It has been shown that by using high-level estimation it is possible to get accurate and meaningful estimations of the energy consumption of the algorithms [15]–[17]. The estimates are a good basis for knowing how the algorithms scale on real hardware architectures. We are also interested in further improving the energy-efficiency of the DLS algorithm. We analyse the energy cost of the DLS algorithm and propose a novel method to reduce its energy consumption. The resulting proposed energy-efficient distributed leader selection (EEDLS) algorithm is described and compared to the diffusion algorithm and the DLS algorithm. The results are verified by simulations. For the simulations, the algorithms are mapped onto widely used WSN hardware architectures (MSP430 and RSL10).

To summarize, the purpose of this work is to investigate and compare the diffusion algorithm and the DLS algorithm from the perspective of energy cost of the computations and radio communication, and propose a novel method to improve the energy-efficiency of the DLS algorithm.

The research contributions are detailed as follows:

- The analysis and comparison of the diffusion algorithm's and DLS algorithm's complexities and number of operations required. Previous works have outlined the number of operations required for the diffusion algorithm [11]; this work extends this by including the additional operations and complexity for the weight calculations and comparing it to the complexity and operations of the DLS algorithm. In addition, the impact of the size of the neighbourhood and of the LMS filter length on the number of operations required are explicitly considered.
- A novel method for reducing the energy consumption of the DLS algorithm by reducing the computations and radio communication. There exist works on reducing the energy consumption of the diffusion algorithm by modifying either the algorithm, the communication frequency, or the topology [18]–[21]. However, in these works additional complexity is introduced [18], [19] or the performance of the diffusion algorithm degrades [20], [21]. We propose a novel EE method for reducing the energy consumption of the DLS algorithm. In contrast to existing works, the proposed EEDLS algorithm is able to reduce the amount of required radio communications to the minimum and to reduce the complexity while retaining the performance of the DLS algorithm.
- A numerical simulation study of the computational and radio communication energy consumption of the diffusion, DLS, and the proposed EEDLS algorithms considering the MSP430 MCU and the RSL10 radio module. Previous works have focused on the diffusion algorithm's analytical side [22], [23] and, to the authors' best knowledge, there are no works that explore the physical energy requirements of the diffusion algorithms and that estimate the energy consumption on real hardware models. Simulation results of the EEDLS algorithm show that the network energy consumption is reduced by $32-53\%$ and the network lifetime is extended by $14-46\%$ as compared to the DLS algorithm and diffusion algorithm.

The remainder of the paper is organized as follows. In Section II we outline the diffusion algorithm and the DLS algorithm, as well as describe the problem setting. Section III gives an overview of the computational and radio communication energy consumption estimation and the network lifetime estimation. The complexity and number of operations required for the diffusion algorithm and DLS algorithm are given and compared. The computational impact of the neighbourhood and of the LMS adaptive filter length are described. The proposed EEDLS algorithm is introduced in Section IV. Section V presents the simulation setting and the numerical results for the diffusion, DLS, and EEDLS algorithms for different topologies. Section VI concludes the paper.

In this paper, italic letters are used for scalars (e, E) and lower case bold letters denote vectors $(\mathbf{x})$. All vectors are column vectors, except for the regression vector $\mathbf{u}_{k,i}$. The operator $E[\cdot]$ stands for mathematical expectation of the subject and (y^*) denotes the complex conjugate transpose of (y).

II. ENERGY-CONSTRAINED DISTRIBUTED ESTIMATION

In this section, we introduce the problem setting and outline the diffusion and DLS algorithm for the readers' better understanding of the analysis of the algorithms presented later on in Section III.

A. GENERAL MODEL

Assume that there are K nodes that are estimating a common parameter, signal source, or information about some target or object. Each of the nodes have access to $d_k(i)$ and

$\mathbf{u}_{k,i}$ at each time instant i:

$$d_k(i) = \mathbf{u}_{k,i}\mathbf{w}^o + v_k(i), \tag{1}$$

where $d_k(i)$ is the measurement at node k, $\mathbf{u}_{k,i}$ is the row regression vector, $\mathbf{w}^o$ is the unknown column vector and $v_k(i)$ is zero-mean white random noise with power $\sigma^2_{v,k}$.

$v_k(i)$ and $\mathbf{u}_{k,i}$ are assumed to be independent for all k values. The $\mathbf{u}_{k,i}$ has a positive-definite covariance matrix, $R_{u,k} = E\,\mathbf{u}^*_{k,i}\mathbf{u}_{k,i} > 0$. The nodes estimate $\mathbf{w}^o$ by minimizing the global cost function:

$$J_k(w) = E \mid d_k(i) - \mathbf{u}_{k,i}\mathbf{w} \mid^2 . \tag{2}$$

Each of the nodes employs the LMS algorithm for adaptation:

$$\mathbf{w}_{k,i} = \mathbf{w}_{k,i-1} + \mu_k \mathbf{u}^*_{k,i}[d_k(i) - \mathbf{u}_{k,i}\mathbf{w}_{k,i-1}], \tag{3}$$

where μ_k is a constant positive step-size such that, $0 < \mu_k < \frac{2}{\lambda_{max}(R_{u,k})}$.

In this manner the nodes in the network are working independently by employing a LMS filter locally to the data. The performance of the individual nodes and that of the network can be improved if the nodes cooperate and communicate in some manner [10]. As noted earlier, we are looking at fully distributed algorithms and we focus on the diffusion algorithm and the DLS algorithm as they have been shown to have good performance [14].

The K nodes form a WSN. Each of the nodes is able to communicate with a subset of nodes and with itself, which forms the node k-s neighbourhood $\mathbb{N}_k$. In addition, the nodes are energy-constrained which means that the energy available to the nodes is limited; the nodes have a certain amount of battery capacity C. During its lifetime, a node can carry out a certain amount of communications to its neighbours as well as a certain amount of computations, which is reflected by the amount of iterations the node can carry out before running out of energy and dying. The topology of the network is assumed to be strongly connected and the connections between the nodes are lossless and noiseless. The nodes are identical with respect to processing power and physical attributes. The nodes do not possess any *a priori* knowledge about other nodes and the network topology.

B. DIFFUSION

We outline the diffusion algorithm in this subsection. In the following two variants of the diffusion algorithm, each of the nodes calculate the estimates using an LMS filter. The estimates are shared with the node's neighbours from the neighbourhood $\mathbb{N}_k$. The nodes combine the estimates using weights $\alpha_{l,k}(i)$ which form the $\mathbf{A}$ matrix [24]. Since there are limitations to the energy available to the nodes, we do not consider exchanging the measurements between the nodes, i.e. the matrix for measurement weights is $\mathbf{C} = I$ and the measurements are only available to the node itself. Sharing the measurements can improve the performance of the diffusion algorithm, but will double the amount of information exchanges per iteration [24]. The combination step and adaptation step can be performed in different order, resulting in the ATC (adapt and then combine) and the CTA (combine and then adapt) variants of the diffusion algorithm which are given as Algorithms 1 and 2, respectively. The algorithms have identical computational complexity [11], but it has been shown that the ATC algorithm attains better performance and therefore, later on we consider the ATC algorithm [24].

Algorithm 1 ATC Diffusion

1: **for** each time instant $i > 0$:
2: each node $k = 1, 2, ..., K$ performs the update:
3: $\boldsymbol{\psi}_{k,i} = \mathbf{w}_{k,i-1} + \mu_k \mathbf{u}^*_{k,i}[d_k(i) - \mathbf{u}_{k,i}\mathbf{w}_{k,i-1}]$
4: $\mathbf{w}_{k,i} = \sum_{l\in\mathbb{N}_k} \alpha_{l,k}(i)\boldsymbol{\psi}_{l,i}$
5: **end**

Algorithm 2 CTA Diffusion

1: **for** each time instant $i > 0$:
2: each node $k = 1, 2, ..., K$ performs the update:
3: $\boldsymbol{\psi}_{k,i-1} = \sum_{l\in\mathbb{N}_k} \alpha_{l,k}(i)\mathbf{w}_{l,i-1}$
4: $\mathbf{w}_{k,i} = \boldsymbol{\psi}_{k,i-1} + \mu_k \mathbf{u}^*_{k,i}[d_k(i) - \mathbf{u}_{k,i}\boldsymbol{\psi}_{k,i-1}]$
5: **end**

There exist several methods for calculating the weights $\alpha_{l,k}(i)$ in the $\mathbf{A}$ matrix, such as Metropolis, uniform, relative degree etc. [24]–[26]. The network performance can be further improved by assigning weights based on the situation of the nodes, which is related to some secondary parameter [10]. The noise power $\sigma^2_{v,k}$ can be used as a secondary parameter to calculate the weights. In practice, the noise powers are not usually known and have to be estimated. Works on estimating noise powers at different nodes are available in e.g. [12], [27], [28]. In this work, we assume that the nodes have *a priori* knowledge of their noise powers as the calculations for the secondary parameter estimation would be identical for the diffusion algorithm and DLS algorithm. The relative variance rule and relative degree-variance rule weight calculations for the diffusion algorithms have shown good and similar performance and are given in (5) and (5) [27].

$$\alpha_{l,k}(i) = \begin{cases} \dfrac{\sigma^{-2}_{v,l}}{\sum\limits_{l\in\mathbb{N}_k} \sigma^{-2}_{v,l}}, & \text{if } l \in \mathbb{N}_k \\ 0, & \text{if } l \notin \mathbb{N}_k, \end{cases} \tag{4}$$

$$\alpha_{l,k}(i) = \begin{cases} \dfrac{n_l\sigma^{-2}_{v,l}}{\sum\limits_{l\in\mathbb{N}_k} n_l\sigma^{-2}_{v,l}}, & \text{if } l \in \mathbb{N}_k \\ 0, & \text{if } l \notin \mathbb{N}_k, \end{cases} \tag{5}$$

where $\alpha_{l,k}(i)$ is the weight calculated at node k for neighbouring node l, i is the current iteration, $\sigma^2_{v,k}$ is the noise power at node k, $\mathbb{N}_k$ is the neighbourhood of node k and n_k is the number of neighbours of the node plus the node itself.

The diffusion algorithm including the weight calculation based on the relative variance rule is given as Algorithm 3.

Algorithm 3 Diffusion Algorithm

1: **for** each time instant $i > 0$:
2: each node $k = 1, 2, ..., K$ performs the update:
3: $\boldsymbol{\psi}_{k,i} = \mathbf{w}_{k,i-1} + \mu_k \mathbf{u}_{k,i}^{*}[d_k(i) - \mathbf{u}_{k,i}\mathbf{w}_{k,i-1}]$
4: **for** each $l \in \mathbb{N}_k$
5: $\alpha_{l,k}(i) = \frac{\sigma_{v,l}^{-2}}{\sum_{l\in\mathbb{N}_k} \sigma_{v,l}^{-2}}$
6: **end**
7: $\mathbf{w}_{k,i} = \sum_{l\in\mathbb{N}_k} \alpha_{l,k}(i)\boldsymbol{\psi}_{l,i}$
8: **end**

C. DISTRIBUTED LEADER SELECTION

In contrast to the diffusion algorithms, the nodes in the DLS algorithm determine a leader node in the network. The DLS has been previously introduced in [12] and [14]. The nodes only have access to the information from the neighbours from the neighbourhood $\mathbb{N}_k$ and select the node which corresponds to the best performing neighbour c_k. The estimates from this node are used and distributed along with the corresponding weight (Algorithm 4). In this manner, the nodes converge to local leaders in their neighbourhood and then to a global leader. A secondary parameter is required to select the leader. Whether this secondary parameter is known or estimated, the nodes are able to use its information to determine which of the nodes is the best performing one and follow its lead. In this work, as for the diffusion algorithm, the secondary parameter is the noise power at a node and is assumed to be known by the node (see Subsection II-B).

The nodes exchange the estimate $\mathbf{e}_{k,i}$ between themselves, together with the corresponding weight $\alpha_k(i)$ that is assigned to the estimate at iteration i. Unlike the diffusion algorithm where the communication paths are weighted, the weights of the DLS algorithm correspond to the nodes themselves. The nodes compare their weight to that of their neighbouring nodes in $\mathbb{N}_k$ and if they have the highest weight, they become the leader node (Algorithm 4 line 4). Otherwise they become follower nodes, in which case they listen to their best neighbour c_k (Algorithm 4 line 8). $\mathbf{e}_{k,i}$ is the estimation result that is used by the node.

III. ENERGY CONSUMPTION ESTIMATION

In this section, the energy consumption models and estimation method are described. The complexity and the required number of operations for the diffusion algorithm and DLS algorithm are derived.

A. NETWORK ENERGY CONSUMPTION

We estimate the energy consumed by a sensor node by following the model proposed and used in [29] and [30]. The energy E consumed by each sensor is given as:

$$\frac{E}{V} = I_a t_a + I_l t_l + I_t t_t + I_r t_r + \sum I_c t_c, \tag{6}$$

where V is the voltage of the power supply. I_a and t_a are the current drawn and execution time of the MCU (Microprocessor Control Unit) in the active mode. I_l and t_l are the current drawn and execution time of the MCU in the low power mode. I_t, I_r, t_r and t_r are the currents drawn and execution times of the radio in transmit and receive modes, respectively. I_c and t_c are the currents drawn and execution time of other components.

Since we are interested in the comparison and the performance of the algorithms, we simplify the formula by focusing only on the active, transmit and receive modes. The energy consumption for the low power mode and for the other components are assumed to be the same for both algorithms. We also have to take into account that the computational times, transmit and receive times for each node are unique due to the topology and how many neighbours the node communicates with (see Subsection II-A). We note that if the amount of neighbours change during estimation, the value changes according to the iteration i. We then can write for node k at iteration i:

$$\frac{E_k(i)}{V} = I_a t_{a,k}(i) + I_t t_{t,k}(i) + I_r t_{r,k}(i) \tag{7}$$

and the energy consumption for node k at iteration i:

$$\begin{aligned} E_k(i) &= V(I_a t_{a,k}(i) + I_r t_{r,k}(i) + I_t t_{t,k}(i)) \\ &= E_{comp,k}(i) + E_{radio,k}(i). \end{aligned} \tag{8}$$

Node k uses $E_k(i)$ energy per iteration i. $E_{radio,k}(i)$ is the communication energy consumption of node k and the computational energy consumed by node k at iteration i is $E_{comp,k}(i)$. The energy consumed by the network in one iteration i is given as:

$$E_{network}(i) = \sum_{k=1}^{K} E_k(i). \tag{9}$$

The energy consumed by the network during the observation period i_{obs} is given as:

$$E_{obs} = \sum_{i=1}^{i_{obs}} E_{network}(i). \tag{10}$$

Algorithm 4 Distributed Leader Selection

1: **for** each time instant $i > 0$:
2: each node $k = 1, 2, ..., K$ performs the update:
3: $\mathbf{w}_{k,i} = \mathbf{w}_{k,i-1} + \mu_k \mathbf{u}_{k,i}^{*}[d_k(i) - \mathbf{u}_{k,i}\mathbf{w}_{k,i-1}]$
4: **if** $\alpha_k(i-1) \geq \max_{k\in\mathbb{N}_k}(\alpha_k(i-1))$
5: $\mathbf{e}_{k,i} = \mathbf{w}_{k,i}$
6: $\alpha_k(i) = (1-\mu_k)\alpha_k(i-1) + \mu_k \sigma_{v,k}^{-2}$
7: **else**
8: $c_k = \operatorname{argmax}_{k\in\mathbb{N}_k}(\alpha_k(i-1))$
9: $\mathbf{e}_{k,i} = \mathbf{e}_{c_k,i}$
10: $\alpha_k(i) = (1-\mu_k)\alpha_{c_k}(i-1) + \mu_k \sigma_{v,k}^{-2}$
11: **end**
12: **end**

B. COMPUTATIONAL ENERGY CONSUMPTION

The computational energy consumption of the algorithms can be calculated through the complexity and the number of operations. From these values we can deduct the execution time needed for the computations and the energy consumed. It has been shown that calculating the energy consumption based on the number of operations can be used as a good benchmark [31]. We also note that the current drawn by the MCU is considered constant during different operations [29]. Therefore, the computational energy consumption from node to node differs in the execution time, which in turn is dependent on the number of operations the node has to carry out.

The complexities for the diffusion algorithm (Subsection II-B) and DLS algorithm (Subsection II-C) per node are given in Tables 1, 2 and 3. The complexities of the LMS and diffusion step have been derived in earlier works, but the complexity for calculating the weights had not been included [11]. M is the size of the LMS filter and n_k is the size of the neighbourhood. To further reduce the computational complexity, the relative variance method is used since the relative degree-variance method would introduce additional complexity due to additional multiplications in the weight calculations (4), (5). For the DLS, the complexities of the leader node and of a follower node are given separately as the nodes have different complexities, see Tables 2 and 3. We can see that the complexity of the node employing the diffusion algorithm is higher than that of the node employing the DLS algorithm, which makes the latter less complex.

The computational energy required for one iteration i at node k can be calculated as:

$$E_{comp,k}(i) = I_c V t_{c,k}(i), \tag{11}$$

where $t_{c,k}(i)$ is the execution time for node k at iteration i in seconds, V is the supply voltage and I_c is the current consumed by the MCU in a second.

The execution time for the operations at node k at iteration i is given as:

$$t_{c,k}(i) = f(\#O_{m,k}(i)C_m + \#O_{d,k}(i)C_d + \#O_{a,k}(i)C_a + \#O_{c,k}(i)C_c), \tag{12}$$

where f is the frequency of the MCU, $\#O_{m,k}(i), \#O_{d,k}(i), \#O_{a,k}(i), \#O_{c,k}(i)$ are the number of multiplications, divisions, additions and comparisons, respectively, and C_m, C_d, C_a, C_c are the number of clock cycles required for multiplication, division, addition and comparisons, respectively.

The multiplications, divisions, additions, comparisons and vector exchanges for the diffusion algorithm and the DLS algorithm are given in Table 4. We can see that the number of operations required for the diffusion algorithm increase substantially in comparison to the DLS algorithm when the filter size M increases (Fig. 1a) or when the neighbourhood of the node increases (Fig. 1b). Therefore, for applications with larger filter lengths and more connected networks, the DLS algorithm would be preferred.

TABLE 1. Diffusion algorithm complexity.

Diffusion	Complexity
$\boldsymbol{\psi}_{k,i} = \mathbf{w}_{k,i-1} + \mu_k \mathbf{u}_{k,i}^*[d_k(i) - \mathbf{u}_{k,i}\mathbf{w}_{k,i-1}]$	$\mathcal{O}(2M)$
for each $l \in \mathbb{N}_k$ $\alpha_{l,k}(i) = \frac{\sigma_{v,l}^{-2}}{\sum_{l\in\mathbb{N}_k}\sigma_{v,l}^{-2}}$ **end**	$\mathcal{O}(n_k)$
$\mathbf{w}_{k,i} = \sum_{l\in\mathbb{N}_k} \alpha_{l,k}(i)\boldsymbol{\psi}_{l,i}$	$\mathcal{O}(n_k M)$
	$\mathcal{O}(2M + n_k + n_k M)$

TABLE 2. DLS (leader node) algorithm complexity.

DLS (leader node)	Complexity
$\mathbf{w}_{k,i} = \mathbf{w}_{k,i-1} + \mu_k \mathbf{u}_{k,i}^*[d_k(i) - \mathbf{u}_{k,i}\mathbf{w}_{k,i-1}]$	$\mathcal{O}(2M)$
if $\alpha_k(i-1) \geq \max_{k\in\mathbb{N}_k}(\alpha_k(i-1))$	$\mathcal{O}(n_k)$
$\mathbf{e}_{k,i} = \mathbf{w}_{k,i}$	$\mathcal{O}(1)$
$\alpha_k(i) = (1-\mu_k)\alpha_k(i-1) + \mu_k\sigma_{v,k}^{-2}$	$\mathcal{O}(2)$
	$\mathcal{O}(2M + n_k + 3)$

TABLE 3. DLS (follower node) algorithm complexity.

DLS (follower node)	Complexity
$\mathbf{w}_{k,i} = \mathbf{w}_{k,i-1} + \mu_k \mathbf{u}_{k,i}^*[d_k(i) - \mathbf{u}_{k,i}\mathbf{w}_{k,i-1}]$	$\mathcal{O}(2M)$
if $\alpha_k(i-1) \geq \max_{k\in\mathbb{N}_k}(\alpha_k(i-1))$	$\mathcal{O}(n_k)$
$c_k = \operatorname{argmax}_{k\in\mathbb{N}_k}(\alpha_k(i-1))$	$\mathcal{O}(n_k)$
$\mathbf{e}_{k,i} = \mathbf{e}_{c_k,i}$	$\mathcal{O}(1)$
$\alpha_k(i) = (1-\mu_k)\alpha_{c_k}(i-1) + \mu_k\sigma_{v,k}^{-2}$	$\mathcal{O}(2)$
	$\mathcal{O}(2M + 2n_k + 3)$

TABLE 4. Operations for diffusion and DLS.

Operation	Diffusion	DLS	
		Follower	Leader
Multiplications	$2M + Mn_k$	$2M+2$	$2M+2$
Divisions	n_k	–	–
Additions	$Mn_k + M + n_k$	$M+2$	$M+2$
Comparisons	–	$2n_k$	n_k
Vector exchanges	n_k	n_k	n_k

To further illustrate this point, the amount of operations required for one node in the diffusion network and the DLS network are given in an example (Fig. 1c). While the filter length M is the same for all the nodes, the number of neighbours n_k is different from node to node and nodes with larger numbers of neighbours require more operations. We can see that the number of operations for the diffusion algorithm's nodes depend more on the topology and will vary to a larger degree than for the DLS algorithm nodes.

C. RADIO COMMUNICATION ENERGY CONSUMPTION

The radio communication energy consumed $E_{radio,k}(i)$ by node k at iteration i is given as:

$$E_{radio,k}(i) = V(I_r t_{r,k}(i) + I_t t_{t,k}(i)), \tag{13}$$

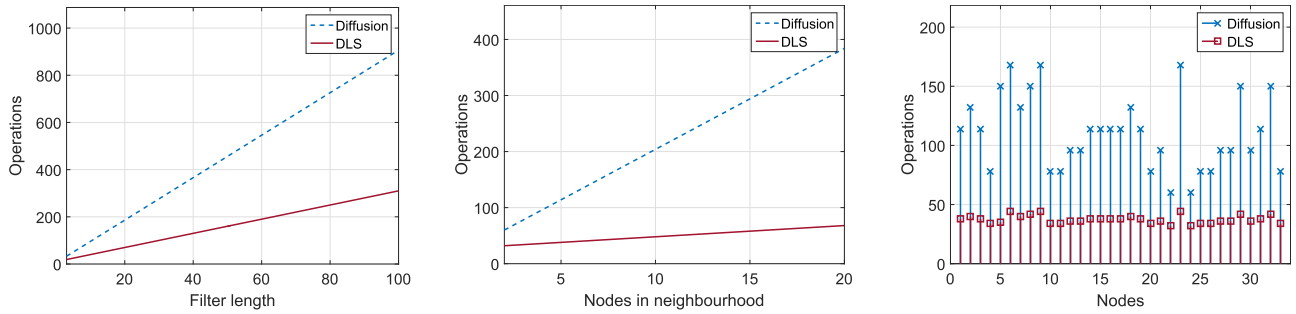


FIGURE 1. (a) Number of operations based on the length of the LMS filter. (b) Number of operations based on the number of nodes in the neighbourhood. (c) Number of operations per node of the diffusion algorithm and the DLS algorithm.

where I_r and I_t are the currents drawn by the receiver and transmitter, respectively, and $t_{r,k}(i)$ and $t_{t,k}(i)$ are the communication periods for the receiver and transmitter at node k at iteration i, respectively.

Each of the nodes communicates with each of its neighbours once per iteration. Nodes with more connections consume more energy. The vector exchanges per iteration are given as n_k in Table 4. It is important to note that the value n_k includes the self-loop i.e. the vector exchange with the node itself. Since the node does not consume energy communicating with itself, the amount of radio communications required is $n_k - 1$. We can write the energy consumed by each communication path as $V(I_r t_r + I_t t_t)$ and write the communication energy consumed at node k at iteration i as:

$$E_{radio,k}(i) = (n_k(i) - 1)V(I_r t_r + I_t t_t), \tag{14}$$

where $n_k(i) - 1$ is the number of neighbours for node k.

We can further write that the network radio consumption can be given as:

$$\begin{aligned} E_{radio}(i) &= \left(\sum_{k=1}^{K}(n_k(i) - 1)\right) V(I_r t_r + I_t t_t) \\ &= (N(i) - K)V(I_r t_r + I_t t_t), \end{aligned} \tag{15}$$

where $N(i) - K$ is the amount of connections without the self-loops in the network at iteration i.

D. NETWORK LIFETIME

The network lifetime can be computed by taking into account the computational and the communication energy consumptions. We refer to the method proposed in [32]. Each node is assumed to have the same battery capacity C. The lifetime for a sensor node k can be computed as:

$$Lt_k = \frac{C}{\overline{I}_k}, \tag{16}$$

where C is the battery capacity in mAh, Lt_k is the lifetime of the node k, $\overline{I}_k$ is average current consumption.

Since we are also interested in the amount of iterations the nodes are able to carry out before running out of battery energy and dying, we modify the formula as:

$$LI_k = \frac{C}{\overline{E}_k} \tag{17}$$

where LI_k is the lifetime of the node k in number of iterations and $\overline{E}_k$ is the average energy consumed by node k.

IV. ENERGY-EFFICIENT DISTRIBUTED LEADER SELECTION

In this section, the proposed EEDLS algorithm is introduced. The algorithm is based on the DLS algorithm introduced in Subsection II-C.

A. PROPOSED ALGORITHM

The goal of the proposed EEDLS is to reduce the energy consumed by the nodes in the DLS algorithm. Under the assumption that the energy saving mode duration has been selected appropriately so that during this duration the leader node does not change, we can reduce the amount of connections and computations. In this manner we are able to use the energy more efficiently and extend the lifetime of the network. The basis of this energy reduction is the vector $\mathbf{c}$ which contains the neighbours of all the nodes that they follow, see Algorithm 4. Based on this vector we are able to reduce the amount of connections and the traffic required in the network and modify the topology to reduce the communication energy consumption. Since the nodes only use the information from the best performing neighbour, communication to the other nodes is redundant and connections only remain between the nodes which are used to spread the information across the network.

Working under the assumption that during the energy saving mode the leader node does not change, we can determine the leader in the network and then discontinue its selection. Based on this, we are able to reduce the number of computations. In addition, since the computations from the follower nodes are not required and only computations from the leader node are required, we can reduce the computational complexity of the algorithm as well. These reductions do not impact the MSD (mean squared deviation) performance of the EEDLS algorithm, which can be seen in Fig. 2a. Therefore, the MSD performance of the EEDLS algorithm and DLS

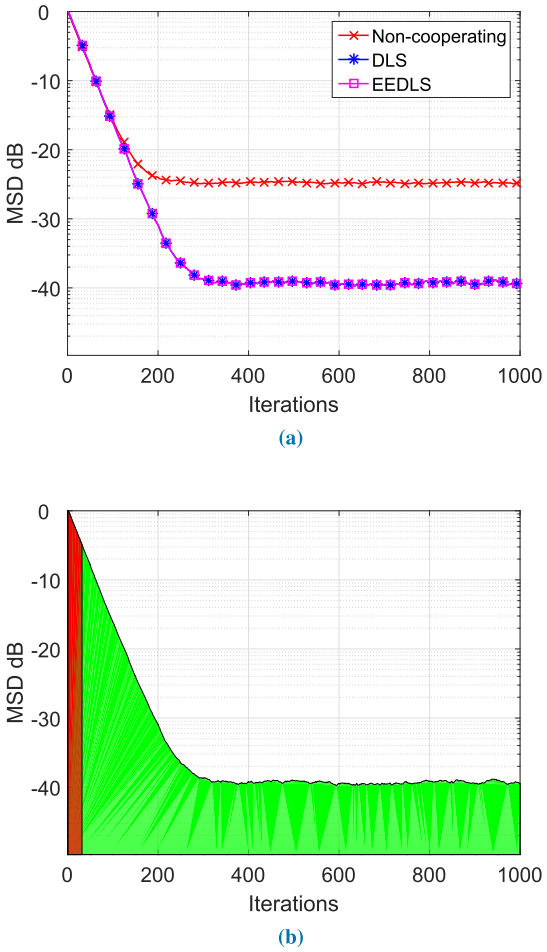


FIGURE 2. (a) MSD network performance of the non-cooperating algorithm, DLS algorithm and the EEDLS algorithm. (b) The EEDLS algorithms leader selection mode (red) and energy saving mode (green).

algorithm are identical; for reference, the performance of the DLS algorithms has been shown previously in [12] and [14].

In order to obtain the vector $\mathbf{c}$, the network has to run and select the leader, which can be accomplished after a certain number of iterations. The amount of iterations required for this depends on the topology of the network and the amount of nodes. The maximum amount of iterations for the leader selection is $i_{ls} = K - 1$ which is the longest path in the network with K nodes if all the nodes in the network are in serial connections. This is the maximum amount of iterations required for the leader node to be selected for the worst case scenario network. In other cases, the leader would be selected in lesser amount of iterations. For example, in the star topology the leader node would be selected in one iteration.

The information about the amount of nodes in the network might not be available beforehand, but during the leader selection time this can be communicated or estimated between the nodes. In this work, we assume that the nodes are able to communicate it across the network.

During the leader selection mode, the algorithm performs as the DLS algorithm (Fig. 2b (red)). After the leader has been determined, the network enters the energy saving mode where redundant connections are disabled and the computational complexity is decreased (Fig. 2b (green)). The leader node continues the adaptation step (Algorithm 5) and the follower nodes follow the leader nodes estimations. The follower nodes relay the information from the leader across the network (Algorithm 6) and the leader continues its estimation until the observation period ends.

Algorithm 5 EEDLS (Leader Node)

for each time instant $i > i_{ls}$:
  $\mathbf{w}_{k,i} = \mathbf{w}_{k,i-1} + \mu_k \mathbf{u}^*_{k,i}[d_k(i) - \mathbf{u}_{k,i}\mathbf{w}_{k,i-1}]$
  $\mathbf{e}_{k,i} = \mathbf{w}_{k,i}$
end

Algorithm 6 EEDLS (Follower Node)

for each time instant $i > i_{ls}$:
  $\mathbf{e}_{k,i} = \mathbf{e}_{c_k,i}$
end

B. COMPUTATIONAL ENERGY REDUCTION

The complexity of the computations in the nodes is reduced after i_{ls} iterations and is given in Table 5. Before $(i_{ls} + 1)$ iterations, the complexity of the algorithm is that of the DLS algorithm as given in Tables 2 and 3. The leader node resumes the LMS adaptation step and the follower nodes only receive and send information to other nodes. The number of operations required is given in Table 6. The size of the neighbourhood n_k does no longer affect the amount operations required for one iteration and the length of the filter M only impacts the amount of operations required for the leader node.

TABLE 5. The EEDLS algorithm's leader and follower node complexities in the energy saving mode.

EEDLS (leader)	Complexity
$\mathbf{w}_{k,i} = \mathbf{w}_{k,i-1} + \mu_k \mathbf{u}^*_{k,i}[d_k(i) - \mathbf{u}_{k,i}\mathbf{w}_{k,i-1}]$	$\mathcal{O}(2M)$
$\mathbf{e}_{k,i} = \mathbf{w}_{k,i}$	$\mathcal{O}(1)$
EEDLS (follower)	**Complexity**
$\mathbf{e}_{k,i} = \mathbf{e}_{c_k,i}$	$\mathcal{O}(1)$

TABLE 6. The EEDLS algorithm's operations for the leader and follower node in the energy saving mode.

Operation	EEDLS leader	EEDLS follower
Multiplications	$2M + 1$	–
Divisions	–	–
Additions	2	–
Comparators	–	–
Vector exchanges	$[2, n_k]$	$[2, n_k]$

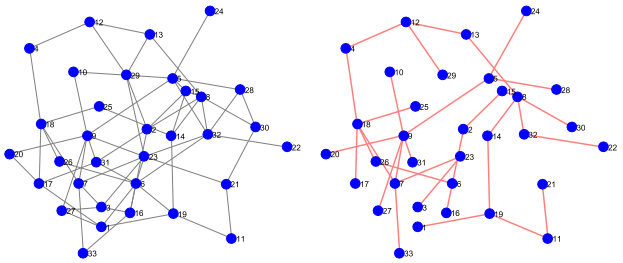
C. RADIO COMMUNICATION ENERGY REDUCTION

We switch off the communication that is redundant for the DLS algorithm (Algorithm 5, 6). The selection of the communication paths is determined by the vector **c** which holds the connections that are required to maintain the performance of the algorithm.

The resulting amount of connections might not decrease for every node; the amount of vector exchanges for one node is given in Table 6. The minimum number of vector exchanges is 2 when the node is communicating to one node and to itself. The minimum number of radio communications required is 1 as the node does not need radio communication for the vector exchange with itself. If all the connections of a node are needed for the network connectivity or to maintain the performance of the algorithm, then the resulting amount of vector exchanges is n_k and radio communications $n_k - 1$. Therefore, the radio communications for one node depend on the topology of the network, as can be seen from the example in Fig. 3. Overall, the amount of radio communications in the network is reduced to the minimum amount of communications possible to retain the connectivity which is $N_{EEDLS} = K - 1$.

$$N_{EEDLS} = \sum_{k=1}^{K} (n_k(i_{ls} + 1) - 1) = K - 1, \tag{18}$$

where $n_k(i_{ls} + 1) - 1$ is the amount of neighbours for node k in the energy saving mode.



(a) (b)

FIGURE 3. (a) Topology of the network. (b) The EEDLS algorithm topology with reduced connections.

The communication energy consumed by the network after i_{ls} iterations can be computed as:

$$E_{radio}(i_{ls} + 1) = (K - 1)V(I_r t_r + I_t t_t). \tag{19}$$

The network radio communication energy savings during the energy saving mode of the reduced topology can be expressed through (15) as:

$$\begin{aligned} E_{\Delta radio} &= ((N(i) - K) - (K - 1))V(I_r t_r + I_t t_t) \\ &= (N(i) - 2K + 1)V(I_r t_r + I_t t_t). \end{aligned} \tag{20}$$

V. SIMULATION RESULTS

In this section we present the simulation setup and the numerical results obtained from the simulations for the diffusion, DLS, and EEDLS algorithms.

A. SIMULATION SETUP

For the simulations different sensor networks were randomly generated with different number of nodes and connections, which are given in Table 9. All of the generated networks are strongly connected. We take a closer look at the network with $K = 33$ nodes and $N = 93$ connections which can be seen in Fig. 3a. The networks employ the diffusion algorithm, the DLS algorithm and the EEDLS algorithm (which were introduced in Subsections II-B, II-C and IV-A respectively). The EEDLS topology can be seen from Fig. 3b where the redundant connections have been disconnected. Each of the nodes in the networks employ a TI MSP430 family microprocessor [33]. The TI MSP430 has been selected as it is a widely used low-power MCU in WSNs as can be seen from recent examples in the literature [34]–[36]. The ON Semiconductor RSL10 Ultra-Low-Power Multi-protocol Bluetooth Radio SOC (system on a chip) [37] has been selected as the radio module.

TABLE 7. Simulation parameters.

$C = 3000mAh$	$K = 33$
$I_{Rx} = 3mA$	$T_{Rx} = 2.5ms$
$I_{Tx} = 4,6mA$	$T_{Tx} = 2.5ms$
$I_c = 3.7mA$	$N = 93$
$U = 3V$	$\mu = 0.01$
$f = 16Mhz$	$M = 8$

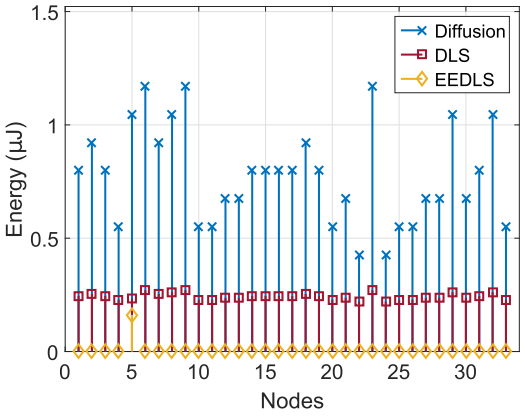
The parameters for the simulations are given in Table 7. The radio communication current values are used as I_{Rx} and I_{Tx} at supply voltage $U = 3V$ [37]. Communication times for sending and receiving are given as T_{Rx} and T_{Tx} [38]. For the computational current, the value I_c is used [38]. The battery capacity for each of the nodes is C. The clock frequency of the MCU is given as f [33]. The LMS filter length for the simulations is selected as $M = 8$ and the step size is selected for all nodes as $\mu = 0.01$.

B. COMPUTATIONAL ENERGY CONSUMPTION

The required clock cycles based on the MSP430 architecture for each of the operations are given in Table 8 [33], [39]. We see that the largest number of clock cycles are required for the division operation [39], which makes the diffusion algorithm computationally heavy as the weight calculations for the algorithm require n_k division for each iteration (Table 4).

TABLE 8. MSP430 clock cycles for different operations.

Operation	Clock cycles
Multiplications	13
Divisions	21
Additions	6
Comparisons	6



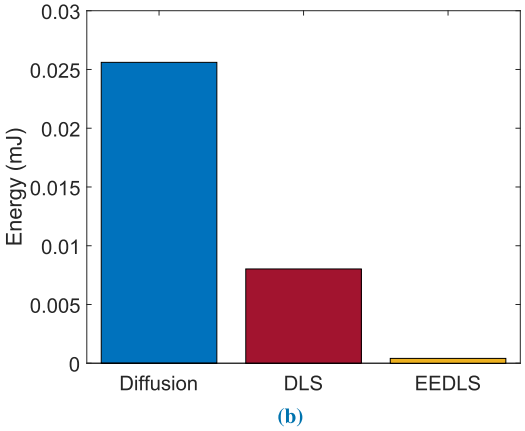
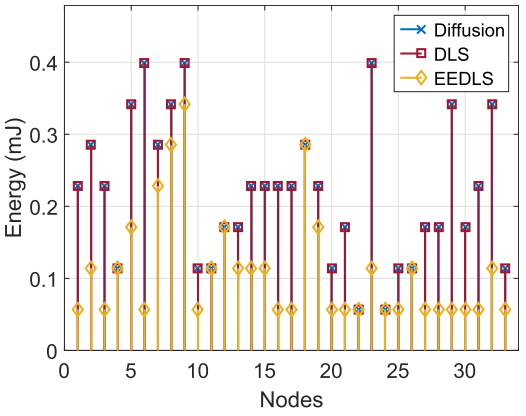


FIGURE 4. (a) Computational energy consumption at different nodes for the diffusion algorithm, DLS algorithm and EEDLS algorithm. (b) Network computational energy consumption for the diffusion algorithm, DLS algorithm and EEDLS algorithm.



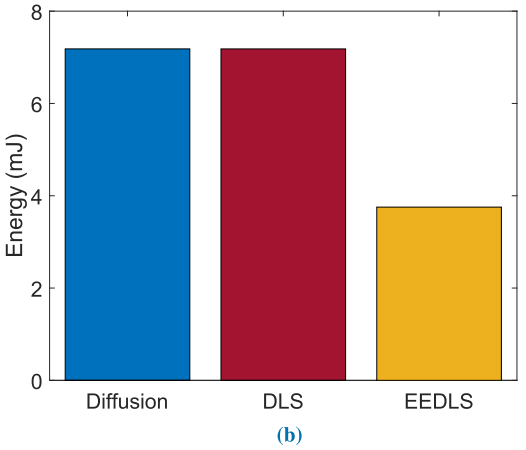


FIGURE 5. (a) Radio communication energy consumption at different nodes for the diffusion algorithm, DLS algorithm and EEDLS algorithm. (b) Network radio communication energy consumption for the diffusion algorithm, DLS algorithm and EEDLS algorithm.

We can see from Fig. 4a that the computational energy consumption for the diffusion algorithm varies from node to node as the computational amount is impacted by the amount of neighbours n_k of node k. The DLS computations vary to a lesser degree from node to node. The EEDLS algorithm in the energy saving mode attains the lowest energy consumption, requiring only the leader node to carry out operations on the MCU. From Fig. 4b we can see that the DLS algorithm consumes 68% less computational energy than the diffusion algorithm on the network level for this example. The EEDLS algorithm is further able to reduce the network computational energy by 98% as compared to the diffusion algorithm.

C. RADIO COMMUNICATION ENERGY CONSUMPTION

The radio energy consumption for the DLS algorithm and the diffusion algorithm are identical as the amounts of communication and vector exchanges are identical. From Fig. 5a we can see that the EEDLS algorithm is able to reduce the radio communication energy for each node compared to the DLS algorithm and the diffusion algorithm. The amount of radio communication energy reduced by the EEDLS algorithm depends on the topology as the amount of radio communications at node k can remain unchanged if the connections are required for the other nodes to retain connectivity to the network (Table 6). The overall network radio communication energy consumption is reduced by 47% (Fig. 5b) as the number of connections in the network has been reduced.

D. OVERALL REDUCED ENERGY CONSUMPTION

In addition to the earlier example ($K = 33$ nodes and $N = 93$ connections) we analyse the results for the two larger randomly generated sensor networks with $K = 206$ and $K = 510$, which can be seen in Table 9. We compare the different topologies employing different algorithms based on the values given in the table. The averaged values have been calculated by averaging the results over 1000 iterations. The network energy has been calculated by (9). The network lifetime in number of iterations has been calculated by considering the first node that runs out of energy and dies based on (17). The average lifetime in iterations is calculated over all the nodes in the network.

TABLE 9. Results for different topologies.

	Diffusion	DLS	EEDLS	Diffusion	DLS	EEDLS	Diffusion	DLS	EEDLS
Nodes	33	33	33	206	206	206	510	510	510
Connections	93	93	93	613	613	613	1530	1530	1530
Average Computational Energy (μJ)	**0.7759**	**0.2432**	**0.0125**	**1.0408**	**0.2618**	**0.0543**	**1.0401**	**0.2617**	**0.1334**
Average Radio Communication Energy (mJ)	0.2176	0.2176	**0.1138**	0.3392	0.3392	**0.1595**	0.3389	0.3389	**0.2281**
Average Energy per Iteration (mJ)	0.2184	0.2179	**0.1138**	0.3403	0.3395	**0.1596**	0.3399	0.3391	**0.2283**
Network Energy (mJ)	7.2076	7.1900	**3.7542**	70.0964	69.9359	**32.8694**	173.3544	172.9575	**116.4101**
Network Lifetime (Iterations)	22490	22541	**26310**	9264	9284	**17475**	9843	9864	**15483**
Average Node Lifetime (Iterations)	53318	534650	**111514**	32847	32929	**113014**	34003	34088	**109368**

We see that the diffusion algorithm's average computational energy consumption is the highest and that the DLS algorithm consumes 68% less energy in the worst case and 74% less in the best case as compared to the diffusion algorithm. The EEDLS algorithm is improved on top of this and is able to further reduce the amount of computational energy required. Compared to the diffusion algorithm the reduction is 98% in the best case and 87% in the worst case. From the average radio communication energy consumption, we see that the DLS algorithm and the diffusion algorithm consume the same amount of energy, as expected. The EEDLS algorithm is able to reduce the radio communication energy consumption by 32% in the worst case and 52% in the best case.

The average energy consumption numbers are quite similar to the radio communication energy consumption values as the impact of the computational energy consumption is marginal in the overall energy consumption. The DLS algorithm consumes less energy than the diffusion algorithm, but only by a slight margin and it can be said that there is no advantage between the two algorithms if we consider both the computational energy consumption and the radio communication energy consumption. Given the above, the EEDLS algorithm is able to reduce the average energy consumption per iteration by 32% in the worst case and by 53% in the best case. The network energy consumption values show the same improvements for the EEDLS algorithm compared to the other algorithms.

The average node lifetime and the network lifetime difference between the DLS and the diffusion algorithm are similar with marginal improvements for the DLS algorithm. The EEDLS algorithm is able to improve the network lifetime in the worst case by 14% and in the best case by 46%. As noted before, if the topology includes nodes that have multiple connections which cannot be disconnected and are needed to retain the performance or the connectivity of some the nodes in the network, the lifetime of these nodes is not improved as much as the other nodes' lifetime in the network. This is evident from the average node lifetime as it improves under the EEDLS algorithm in the worst case by 52% and in the best case by 70%.

Overall, the EEDLS algorithm notably improves the energy-efficiency of the network in every aspect and in some areas by quite large margins. Comparing different network sizes illustrates that the EEDLS algorithm is able to improve the energy-efficiency of all the networks and for larger networks with more connections the improvements are greater.

VI. CONCLUSION

In this work, we started by investigating the computational complexity of the diffusion algorithm and the DLS algorithm. We found that the DLS algorithm is less complex in terms of computations. Furthermore, the DLS algorithm requires less operations and is preferred in applications where the network is more densely connected or longer adaptive filter lengths are required. In addition, we analysed the energy consumption for both of the algorithms taking into account energy-constrained conditions. Whereas the computational efficiency is better for the DLS algorithm in comparison to the diffusion algorithm, the radio communication energy consumption for both algorithms makes the computational energy savings negligible. To further reduce the overall energy consumption we proposed EEDLS, a new energy-efficient distributed leader selection algorithm, which reduces the amount of computations and radio communication in the network while retaining the performance of the DLS algorithm. In the simulation section we demonstrated the energy consumption and illustrated the energy savings of the proposed EEDLS algorithm on the TI MSP430 microcontroller family architecture and on the On Semiconductor RSL10 Bluetooth radio module. We are able to reduce the network energy consumption compared to the diffusion algorithm and the DLS algorithm by 32% in the worst case and 53% in the best case. We are able to extend the network lifetime by 14% in the worst case and 46% in the best case compared to the diffusion and the DLS algorithm. The increase of the average lifetime of a node is 52% in the worst case and 70% in the best case.

REFERENCES

[1] K. Wang, Y. Wang, Y. Sun, S. Guo, and J. Wu, "Green industrial Internet of Things architecture: An energy-efficient perspective," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 48–54, Dec. 2016.

[2] Z. Sheng, C. Mahapatra, C. Zhu, and V. C. M. Leung, "Recent advances in industrial wireless sensor networks toward efficient management in IoT," *IEEE Access*, vol. 3, pp. 622–637, 2015.

[3] R. Mahapatra, Y. Nijsure, G. Kaddoum, N. U. Hassan, and C. Yuen, "Energy efficiency tradeoff mechanism towards wireless green communication: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 686–705, May 2016.

[4] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Comput. Netw.*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008.

[5] Y. Yao, Q. Cao, and A. V. Vasilakos, "EDAL: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for heterogeneous wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 810–823, Jun. 2015.

[6] J. S. Leu, T. H. Chiang, M. C. Yu, and K. W. Su, "Energy efficient clustering scheme for prolonging the lifetime of wireless sensor network with isolated nodes," *IEEE Commun. Lett.*, vol. 19, no. 2, pp. 259–262, Feb. 2015.

[7] S. Rani, R. Talwar, J. Malhotra, S. H. Ahmed, M. Sarkar, and H. Song, "A novel scheme for an energy efficient Internet of Things based on wireless sensor networks," *Sensors*, vol. 15, no. 11, pp. 28603–28626, 2015.

[8] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Comput. Netw.*, vol. 67, pp. 104–122, Jul. 2014.

[9] A. H. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z. J. Towfic, "Diffusion strategies for adaptation and learning over networks: An examination of distributed strategies and network behavior," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 155–171, May 2013.

[10] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Found. Trends Mach. Learn.*, vol. 7, nos. 4–5, pp. 311–801, 2014.

[11] S.-Y. Tu and A. H. Sayed, "Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks," *IEEE Trans. Signal Process.*, vol. 60, no. 12, pp. 6217–6234, Dec. 2012.

[12] S. Ulp and T. Trump, "Leader selection in cooperative network based on MDL subspace algorithm for cognitive radio," in *Proc. 50th Asilomar Conf. Signals, Syst. Comput.*, Nov. 2016, pp. 704–708.

[13] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3122–3136, Jul. 2008.

[14] S. Ulp, Y. Le Moullec, and M. M. Alam, "LMS-based leader selection for distributed estimation," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT)*, Dec. 2017, pp. 211–215.

[15] P. Heinrich, H. Bergler, and D. Eilers, "Energy consumption estimation of software components based on program flowcharts," in *Proc. IEEE Int. Conf. High Perform. Comput. Commun., IEEE 6th Int. Symp. Cyberspace Saf. Secur., IEEE 11th Int. Conf. Embedded Softw. Syst. (HPCC,CSS,ICESS)*, Aug. 2014, pp. 542–545.

[16] V. Konstantakos, A. Chatzigeorgiou, S. Nikolaidis, and T. Laopoulos, "Energy consumption estimation in embedded systems," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 4, pp. 797–804, Apr. 2008.

[17] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step towards software power minimization," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 2, no. 4, pp. 437–445, Dec. 1994.

[18] W. Hu and W. P. Tay, "Multi-hop diffusion LMS for energy-constrained distributed estimation," *IEEE Trans. Signal Process.*, vol. 63, no. 15, pp. 4022–4036, Aug. 2015.

[19] M. O. Sayin and S. S. Kozat, "Compressive diffusion strategies over distributed networks for reduced communication load," *IEEE Trans. Signal Process.*, vol. 62, no. 20, pp. 5308–5323, Oct. 2014.

[20] J. Fernandez-Bes, R. Arroyo-Valles, J. Arenas-García, and J. Cid-Sueiro, "Censoring diffusion for harvesting WSNs," in *Proc. IEEE 6th Int. Workshop Comput. Adv. Multi-Sensor Adapt. Process. (CAMSAP)*, Dec. 2015, pp. 237–240.

[21] Y. Wang, W. P. Tay, and W. Hu, "An energy-efficient diffusion strategy over adaptive networks," in *Proc. 10th Int. Conf. Inf., Commun. Signal Process. (ICICS)*, Dec. 2015, pp. 1–5.

[22] C. G. Lopes and A. H. Sayed, "Diffusion adaptive networks with changing topologies," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar./Apr. 2008, pp. 3285–3288.

[23] C. G. Lopes and A. H. Sayed, "Steady-state performance of adaptive diffusion least-mean squares," in *Proc. IEEE/SP 14th Workshop Stat. Signal Process.*, Aug. 2007, pp. 136–140.

[24] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1035–1048, Mar. 2010.

[25] X. Zhao and A. H. Sayed, "Performance limits for distributed estimation over LMS adaptive networks," *IEEE Trans. Signal Process.*, vol. 60, no. 10, pp. 5107–5124, Oct. 2012.

[26] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. 4th Int. Symp. Inf. Process. Sensor Netw.*, Apr. 2005, pp. 63–70.

[27] S.-Y. Tu and A. H. Sayed, "Optimal combination rules for adaptation and learning over networks," in *Proc. IEEE 4th Int. Workshop Comput. Adv. Multi-Sensor Adapt. Process. (CAMSAP)*, Dec. 2011, pp. 317–320.

[28] N. Takahashi, I. Yamada, and A. H. Sayed, "Diffusion least-mean squares with adaptive combiners: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 58, no. 9, pp. 4795–4810, Sep. 2010.

[29] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based online energy estimation for sensor nodes," in *Proc. ACM 4th Workshop Embedded Netw. Sensors*, 2007, pp. 28–32.

[30] W. Rukpakavong, L. Guan, and I. Phillips, "Dynamic node lifetime estimation for wireless sensor networks," *IEEE Sensors J.*, vol. 14, no. 5, pp. 1370–1379, May 2014.

[31] P. Ruberg, K. Lass, E. Liiv, and P. Ellervee, "Performance estimation of embedded applications on microcontrollers," in *Proc. IEEE Nordic Circuits Syst. Conf. (NORCAS), NORCHIP Int. Symp. Syst.-Chip (SoC)*, Oct. 2017, pp. 1–6.

[32] B. Selvig, "Measuring power consumption with CC2430 & Z-stack," Texas Instrum., Dallas, TX, USA, Appl. Note AN053, 2007, vol. 5.

[33] *MSP430x5xx MSP430x6xx Family User's Guide Rev. Q*, Texas Instrum., Dallas, TX, USA, Mar. 2018.

[34] S. Sonavane, B. Patil, and V. Kumar, "Experimentation for packet loss on MSP430 and nRF24L01 based wireless sensor network," *Int. J. Adv. Netw. Appl.*, vol. 8, no. 5, p. 25, 2017.

[35] S. Kurt, H. U. Yildiz, M. Yigit, B. Tavli, and V. C. Gungor, "Packet size optimization in wireless sensor networks for smart grid applications," *IEEE Trans. Ind. Electron.*, vol. 64, no. 3, pp. 2392–2401, Mar. 2017.

[36] J. Bito, R. Bahr, J. G. Hester, S. A. Nauroze, A. Georgiadis, and M. M. Tentzeris, "A novel solar and electromagnetic energy harvesting system with a 3-D printed package for energy efficient Internet-of-Things wireless sensors," *IEEE Trans. Microw. Theory Techn.*, vol. 65, no. 5, pp. 1831–1842, May 2017.

[37] *Ultra-Low-Power Multi-Protocol Bluetooth 5 Certified Rev. 1*, ON Semicond., Phoenix, AZ, USA, Jan. 2018.

[38] M. M. Alam, O. Berder, D. Menard, T. Anger, and O. Sentieys, "A hybrid model for accurate energy analysis of WSN nodes," *EURASIP J. Embedded Syst.*, vol. 2011, 2011, Art. no. 307079.

[39] *Efficient Multiplication Division Using MSP430*, Texas Instrum., Dallas, TX, USA, 2006.

SANDER ULP received the B.S. and M.Sc. degrees *(cum laude)* in telecommunication from the Tallinn University of Technology (TTÜ), Tallinn, in 2011 and 2013, respectively, where he is currently pursuing the Ph.D. degree. From 2013 to 2016, he was a Junior Researcher with the Department of Radio and Communication Engineering, TTÜ. Since 2017, he has been with the Thomas Johann Seebeck Department of Electronics, TTÜ. His research interests are in distributed estimation, learning and adaptation over networks, wireless sensor networks, digital signal processing, and cognitive radio.

YANNICK LE MOULLEC received the M.Sc. degree in electrical engineering (EE) from the Université de Rennes I, France, in 1999, and the Ph.D. degree in EE from the Université de Bretagne Sud, France, in 2003. From 2003 to 2013, he was a Post-Doctoral Professor, an Assistant Professor, and an Associate Professor with Aalborg University, Denmark. He then joined the Tallinn University of Technology, Estonia, where he was a Senior Researcher, from 2013 to 2016 ,and is currently a Professor. He has supervised nine Ph.D. theses and more than 50 M.Sc. theses; he is also supervising five M.Sc. theses and five Ph.D. theses. His research interests span HW/SW co-design, embedded systems, reconfigurable systems, and IoT. He is a Co-Principal Investigator for the H2020 COEL ERA-Chair Project.

MUHAMMAD MAHTAB ALAM received the M.Sc. degree in electrical engineering from Aalborg University, Denmark, in 2007, and the Ph.D. degree from the INRIA Research Center, University of Rennes 1, in 2013. He did his Post-Doctoral Research in the Qatar Foundation funded project Critical and Rescue Operations Using Wearable Wireless Sensor Networks from the Qatar Mobility Innovations Center, from 2014 to 2016. In 2016, he was elected as a European Research Area Chair holder in Cognitive Electronics Project and an Associate Professor with the Thomas Johann Seebeck Department of Electronics, Tallinn University of Technology. In 2018, he received a tenure professorship to a chair Telia professorship under the cooperation framework between Telia and the Tallinn University of Technology. He has authored and co-authored over 55 research publications. His research interests include self-organized and self-adaptive wireless sensor and body area networks specific to energy efficient communication protocols and accurate energy modeling, the Internet-of-things, public safety and critical networks, embedded systems, digital signal processing, and software defined radio. He is a Principal Investigator of NATO-SPS-G5482 Grant and the Estonian Research Council PUT-Team Grant.