Deployment of Applications in Wireless Sensor Networks: a Gossip-based Lifetime Maximization Approach

Virginia Pilloni[a], *Member, IEEE,* Mauro Franceschelli[a], *Member, IEEE,* Luigi Atzori[a], *Senior, IEEE,* Alessandro Giua[b,a], *Senior, IEEE*

Abstract

In this paper a Wireless Sensor Network (WSN) where the nodes are equipped with a programmable middleware that allows the quick deployment of different applications is considered. We propose a distributed method to extend the network lifetime by managing how tasks are assigned to different nodes. A detailed model of energy consumption of the wireless sensor nodes is provided and a gossip-based algorithm which exploits only iterative and asynchronous optimizations involving local information between neighboring nodes to maximize the global network lifetime is proposed. Experimental results and simulations performed on embedded hardware show that our framework can provide a significant lifetime extension, particularly for networks with high node densities and non-uniform energy consumption or initial battery charge.

Published in: IEEE Transactions on Control Systems technology, 2016

[a] Department of Electrical and Electronic Engineering, University of Cagliari, Cagliari, Italy.

[b] Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296,13397, Marseille, France.

This research has been partially funded by the Artemis JU (GA number 333020) as part of the ACCUS project and grant SIR "Scientific Independence of young Researchers", project CoNetDomeSys, code RBSI14OF6H, funded by the Italian Ministry of Research and Education (MIUR).

Mauro Franceschelli is corresponding author.

I. Introduction

Thanks to their increasing processing and transmission power, nodes in Wireless Sensor Networks (WSNs) have become capable of taking decisions and acting upon the information gathered about the monitored environment. This programmable middleware allows the deployment of applications composed of several tasks running on top of it so as to follow the changing ambient needs. This introduces the issue of assigning tasks to the network nodes to distribute evenly the workload and to maximize the network lifetime.

WSNs lifetime improvement has been approached from different points of view. The authors of [1] introduce an energy aware opportunistic routing protocol aimed at keeping balance between Quality of Service (QoS) and energy efficiency. The work in [2] proposes to balance the transmission burden on mobile nodes by predicting their position and using a cluster routing strategy. In [3] the problem of maximizing energy saving in wireless ad-hoc networks was addressed by proposing a collision-free MAC protocol while in Boggia [4] power saving is achieved with a feedback based dynamic scheduler by trading off QoS and providing bounded delays.

These works do not analyze the energy consumption of a WSN with respect to the application assigned to it. The optimal task allocation to network nodes has been investigated in [5] and [6]. However, in these works the objective is to reduce the energy consumption at node or cluster level rather than maximizing the network lifetime.

In [7] a distributed approach to maximize a sensor network utility function for the whole lifetime of the network, which may be defined in terms of energy consumption, is proposed. It is designed to optimize the scheduling of the modes of operation of sensor nodes such as sensing, communication and sleeping. Differently, we aim to extend the network lifetime of the network by dynamically reassigning processing tasks to be performed on sensed data among nodes. In this way, the data rate of the sensed data flows travelling from sources to sink is adjusted according to a distributed approach.

In [8], a centralized method based on mixed integer programming (MIP) was proposed to deploy applications in a WSN and minimize its energy consumption. This paper exploits the detailed modeling of the energy consumption of sensor nodes presented in [8] and proposes a distributed and scalable method to extend the WSN lifetime for large networks based on iterative and asynchronous local optimizations of the task allocations between neighboring nodes, i.e., gossip-based.

There is a significant literature regarding gossip-based algorithms in several applications. In [9] a gossip-based distributed optimization scheme is exploited to solve a multi-vehicle routing problem with heterogeneous vehicles which have to serve requests distributed in space. In [10] the heterogeneous multi-robot routing problem was augmented with requests that are distributed in space but occur at specified time instants. In [11] the authors solve a distributed task assignment problem with asynchronous and gossip-like local state updates and show improved convergence speed with respect to the state of the art. In [12] gossip algorithms were presented in the context of distributed averaging exploiting pairwise averaging between random pairs of nodes. In [13] gossiping is exploited to solve the distributed averaging problem on digraphs with quantized state variables. In [14] the consensus on the average problem in directed graphs is addressed in the case of unreliable communications. Finally, in [15] a gossip

scheme for distributed averaging that exploits geographic information is proposed and shown to provide greater convergence performance.

Our contribution consists in a distributed and scalable approach to the distributed task allocation problem to maximize the lifetime of a large WSN coupled with a careful modeling and analysis of the energy consumption processes of sensor nodes. Furthermore, simulations and experimental results with embedded hardware on a realistic application scenario are presented.

First, we propose a distributed algorithm which estimates the network lifetime. Second, we propose a distributed algorithm which exploits the estimated lifetime to improve the global network lifetime without exploiting knowledge of network topology or global state information.

One advantage of the proposed distributed approach consists in the capability to deal with large sensor networks due to a computational complexity that does not grow with network size.

This paper is organized as follows. In Section II some preliminaries and the problem statement are introduced. In Section III the modeling of the power consumption in wireless nodes is discussed. In Section IV the proposed distributed approach to improve the network lifetime is presented. In Section V simulation results with experimental data obtained from embedded hardware are presented. Finally, in Section VI concluding remarks are discussed.

II. Problem formulation

We consider the scenario of a WSN where nodes can perform a given set of tasks, such as: data processing, temperature measurement, video monitoring, data transmission and data storage. The set of nodes is defined as $X = \{1, \ldots, N\}$. The considered network topology is represented by a rooted acyclic graph $\mathcal{G}_{\mathcal{X}} = \{X, E_X\}$, where $E_X \subseteq (X \times X)$ is the set of edges that represent point-to-point communication channels between the nodes. Edge (i, j) has its tail in node i and its head in node j, with its orientation representing the direction of the information flow. Let $\bar{D}$ be the length of the longest directed path in $\mathcal{G}_{\mathcal{X}}$. Let $\mathcal{N}_i^{out} = \{j \in X : (i, j) \in E_X\}$ be the set of nodes that receive information from node i and $\mathcal{N}_i^{in} = \{j \in X : (j, i) \in E_X\}$ the set of nodes that send information to node i. Let $\mathcal{N}_i = \mathcal{N}_i^{out} \bigcup \mathcal{N}_i^{in}$ be the neighborhood of node i. A path $\mathbf{p}_{ij}$ is an alternating sequence of consecutive vertices and edges $\mathbf{p}_{ij} = \{i, (i, r), r, (r, k), k, (k, j), j\}$ without repetitions starting from node i and ending in node j. A rooted graph contains a node, called root, which is reachable by every node in the graph by a directed path. Fig. 1 shows an example of network topology.

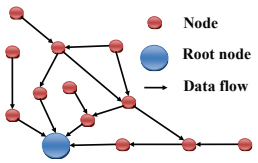


Fig. 1: Example of network topology

The considered application is composed by a set of tasks $T = \{1, \ldots, \Lambda\}$ divided into sensing tasks T^s and processing tasks T^p, with $T = T^s \bigcup T^p$. The relation among tasks is represented by the directed graph $\mathcal{G}_T =$

(T, E_T), where $E_T = (e_{uv})$ is the set of edges. Edge e_{uv} represents a unidirectional data transfer from task u to task v. A binary state vector $\boldsymbol{s}_i \in \{0,1\}^\Lambda$ is assigned to each node. It specifies the tasks currently assigned to the node: the state $s_{i\lambda}$ is equal to 1 if node i performs task λ. Sensing task assignments are given a priori and are not changed during the WSN operation. Since different nodes may be built with different hardware, they may consume different amounts of power for the same task. We consider this heterogeneity of devices in the modeling by considering different costs associated to the same task if it is executed by different nodes. Furthermore, nodes may have different battery or initial charge. To each node i is associated a binary vector $\boldsymbol{d}_i \in \{0,1\}^\Lambda$, which represents the tasks that node i is allowed to execute. To simplify the notation we denote the task assignment matrix that collects the states of all nodes as $\boldsymbol{S} = [\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_N]^T$ and the matrix that represents all constraints on task execution as $\boldsymbol{D} = [\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_N]^T$.

Problem statement

We consider the problem of maximizing the *lifetime of the network* τ, intended as the time it takes for at least one node to exhaust its energy reserve from the battery: when this happens the network topology may disconnected and the application can not be executed [16]. The energy reserve of node i at time t is denoted by $\gamma_i(t)$. Let $P_i(\boldsymbol{S})$ represent the power consumed by node i with the task assignment $\boldsymbol{S}$, as detailed in Section III. The lifetime $\tau_i(t)$ of node i corresponds to $\tau_i(t) = \gamma_i(t)/P_i(\boldsymbol{S})$, where it is assumed that the task assignment matrix $\boldsymbol{S}$ does not change with time. This lifetime is to be considered in absolute terms such as the time and date when at least a node in the network exhaust its energy reserve and disrupts the network operations. Thus, the global lifetime of the network is

$$\tau(t) = \min_{i \in X} \tau_i(t), \tag{1}$$

in the case in which all the transmission rates are constant and that the assignment defined by matrix $\boldsymbol{S}$ does not change. A solution to this problem can be found with a centralized approach by solving the next optimization problem

$$\begin{aligned} &\max_{\boldsymbol{S}} \min_{i \in X} \frac{\gamma_i(t_0)}{P_i(\boldsymbol{S})}, \\ &s.t. \\ &\boldsymbol{d}_i \geq \boldsymbol{s}_i \quad \forall i \in X \\ &\boldsymbol{S} \in \{0,1\}^{N \times \Lambda} \end{aligned} \tag{2}$$

The complexity of solving problem (2) grows exponentially with the number of integer variables and constraints, therefore it is unfeasible to find an optimal solution for networks of significant size. Furthermore, full knowledge of the state of the battery charge on the sensor nodes is needed and this requires a significant communication overhead when considering large scale networks. In the following, the elements of the power consumption modeling that contribute to the total consumed power $P_i(\boldsymbol{S})$ by the generic node i are discussed.

III. Analysis of traffic flows and power consumption

In our scenario we assume that sources of traffic in the network (the sensors) generate samples of k bits at a certain frequency f. The generic node i receives a number of incoming traffic flows represented by vector $\mathbf{\Phi}_i^{in} = (\phi_{i1}^{in}, ..., \phi_{ih_i'}^{in})$, where each element is a traffic flow (bit/sec) and $h = 1, \ldots, h_i'$ are the labels of the traffic flows incoming from its in-neighbors. The output traffic $\mathbf{\Phi}_i^{out} \in \mathbb{R}^{h_i'}$ of node i, can be modeled as

$$\mathbf{\Phi}_i^{out} = p_i(\mathbf{\Phi}_i^{in}, \boldsymbol{s}_i), \tag{3}$$

where $p_i(\cdot) : \mathbb{R}^{h_i'} \times \{0,1\}^{\Lambda} \rightarrow \mathbb{R}^{h_i'}$ is a look-up table that makes a correspondence between the incoming traffic flows, the processing performed by node i on these traffic flows and the output traffic. The number of elements in $\mathbf{\Phi}_i^{in}$ is equal to that of $\mathbf{\Phi}_i^{out}$. The elements of vector $\mathbf{\Phi}_i^{out} = (\phi_{i1}^{out}, ..., \phi_{ih_i'}^{out})$ correspond to traffic flows with samples of size k_{ih}^{out} bits transmitted at frequency f_{ih}^{out}. If node i is aware of the incoming traffic flows $\mathbf{\Phi}_j^{in}$, the processing assignment $\boldsymbol{s}_j$ and the look-up tables of its in-neighbors $j \in \mathcal{N}_i^{in}$, then it can compute its output traffic flows as

$$\mathbf{\Phi}_i^{out} = p_i \left(\sum_{j \in \mathcal{N}_i} p_j \left(\mathbf{\Phi}_j^{in}, \boldsymbol{s}_j \right), \boldsymbol{s_i} \right). \tag{4}$$

For each task, the look-up table of each node $p_i(\cdot)$ has to be experimentally identified. For our objective, this operator is needed to figure out the traffic flows that will be traversing the network for each deployment scenario.

Power consumption in WSNs is determined by three main components: sensing, processing and transmission. The power consumption of node i due to sensing is expressed as

$$P_i^{sens} = \sum_{\lambda \in T^s} f_i^{out} \times e_{i\lambda}^{sens} \times s_{i\lambda}, \tag{5}$$

with $e_{i\lambda}^{sens}$ representing the energy consumption to perform the sensing task λ. Variable $s_{i\lambda}$ is equal to 1 if the task is being executed, it is zero otherwise. f_i^{out} is the frequency of generated output data. The node power consumption due to processing is expressed as

$$P_i^{proc} = \sum_{\lambda \in T^p} \sum_{h=1}^{h_i'} f_{ih}^{out} \times e_{i\lambda}^{proc}(\boldsymbol{\phi}_{ih}^{in}) \times s_{i\lambda}, \tag{6}$$

where $e_{i\lambda}^{proc}$ is the processing energy consumption, which depends on the processing task λ that has to be executed and on the input traffic ϕ_{ih}^{in}. Since the processing cost depends on the number of instructions per second executed by the node, P_i^{proc} is proportional to the frequency f_{ih}^{out} of each of the h traffic flows, where h_i' is the size of $\mathbf{\Phi}_i^{out}$. Power consumption due to transmission of information is

$$P_i^{tx} = f_i \times \sum_{j \in \mathcal{N}_i^{out}} e_{ij}^{tx} \left(\mathbf{\Phi}_i^{out}, \delta_{ij} \right), \tag{7}$$

where f_i is the transmission frequency and e_{ij}^{tx}, defined in [17], is the transmission energy consumption that depends on: the data to be transmitted $\mathbf{\Phi}_i^{out}$; the characteristic parameters of node i; the characteristic parameters of all the nodes that receive data from node i and their distance δ_{ij} from node i. Given (5)-(7), the overall power consumption of node i due to the tasks assignment $\boldsymbol{S}$ can be modeled as

$$P_i(\boldsymbol{S}) = P_i^{sens} + P_i^{proc} + P_i^{tx}. \tag{8}$$

With a slight abuse of notation the power consumption of node i may also be denoted $P_i(\boldsymbol{s}_k : k \in \mathcal{N}_i^{in} \bigcup \{i\})$ to show that (assuming the input flows $\mathbf{\Phi}_j^{in}$ for $j \in \mathcal{N}_i^{in}$ are known) it depends only on the task assigned to node i and to its in-neighbors. A detailed analysis of power consumption in WSNs can also be found in [8].

IV. PROPOSED ALGORITHMS

We now propose two distributed algorithms which provide an approximate solution to problem (2). First, the Minimum Lifetime Estimation (MLE) algorithm, is used to spread information about the current minimum node lifetime in the network. It consists in a broadcast message that is updated every time it is retransmitted by the nodes until all the nodes in the network received at least one of these messages. By the execution of this algorithm each node estimates the minimum node lifetime in the set of nodes included in all the paths between itself and the sink. A similar idea has been exploited for time synchronization in sensor networks [18].

Second, the Decentralized Lifetime Maximization Algorithm (DLMA), executes an iterative local optimization between neighboring nodes exploiting information only locally available to locally reassign the processing tasks to the nodes. The basic idea to achieve the maximization on the network lifetime through a distributed algorithm is to use the information provided by Algorithm MLE to iteratively execute the local optimizations of Algorithm DLMA, while guaranteeing a monotonic increase in the network lifetime.

We now introduce some preliminaries needed to describe Algorithm MLE. Let $\mathcal{X}_i = \{j \in X : \exists \mathbf{p}_{iN}\}$ be the set of all the nodes contained in the path from node i to the root. To each node $i \in X$ a variable x_i is assigned to it to represent its current estimation of the minimum node lifetime of the nodes in the set $\mathcal{X}_i$. Consider a set of instants of time $t_1, t_2, \dots t_k$ which represents the instants of time in which local state updates are performed. The root node $i = N$ sends to all its in-neighbors $j \in \mathcal{N}_N^{in}$ the value of its lifetime τ_N. All the other nodes, at each instant of time, send their current estimation $x_j(t_k)$ to all their in-neighbors. Whenever a node i receives an estimation update by an out-neighbor, it updates its own minimum lifetime estimation according to $x_i(t_{k+1}) = \min\{\tau_i, x_j(t_k) : j \in \mathcal{N}_j^{in}\}$. In Algorithm MLE we state our lifetime estimation method.

Algorithm: Minimum Lifetime Estimation (MLE)

1 - Each node $i \in X$ initializes variable $x_i(t_0) = \tau_i$ its own lifetime and continuously executes the next operations.

2 - Every T units of time, the root node $i = N$ sends the value of $x_N(t_k)$ to the nodes in its in-neighborhood $\mathcal{N}_i^{in}$.

3 - Whenever a node i receives a message from its out-neighbors, it updates its local estimation according to $x_i(t^+) = \min\{\tau_i, x_j(t_k) : j \in \mathcal{N}_j^{out}\}$ and sends its new estimation $x_i(t)$ to its in-neighborhood $\mathcal{N}_j^{in}$.

Algorithm MLE can be seen as a wave of messages triggered by the root node that propagates from the root to the leaves of the network. As soon as the wave has completed its path, each node i becomes aware of the value of the minimum lifetime between the nodes in the set $\mathcal{X}_i$.

Proposition 4.1: Let graph $\mathcal{G}_\mathcal{X}$ be rooted and acyclic. If a message reception/transmission cycle is performed by each node in at most T units of time and $\bar{D}$ is the length of the longest directed path in $\mathcal{G}_\mathcal{X}$, then in at most $T \cdot \bar{D}$ units of time, each node i estimates a lower bound on the minimum lifetime of the nodes in set $\mathcal{X}_i$

$$\forall i \in X, \quad x_i(t) = \min_{j \in \mathcal{X}_i}\{x_j(t - T \cdot \bar{D})\} = \min_{j \in \mathcal{X}_i}\{\tau_j(t - T \cdot \bar{D})\}.$$

Proof: See Appendix A. ■

The root node is supposed to initialize Algorithm MLE periodically each T units of time. This is necessary to keep the information about the minimum network lifetime updated in each node. The execution of Algorithm MLE is independent from the execution of Algorithm DLMA. On the contrary, the execution of Algorithm DLMA requires the information about the network lifetime.

We now focus on the core of proposed technique, the Decentralized Life Maximization for WSNs Algorithm (DLMA). It consists in a local state update rule that nodes apply to their state if triggered by an incoming request message. In the following, let t_k be the instant of time in which the local state update rule at step 2 of Algorithm DLMA is triggered and let t_{k+1} be the instant of time of the next update. The execution of the proposed state update has an energy cost. Let us define $\hat{E}_i$ as the upper bound to the energy spent by solving the local optimization in node i. The node that is expected to execute the optimization, knows its lifetime and corresponding expected lifetime decrement if it executes DLMA:

$$\tau_i(t_{k+1}) = \frac{\gamma_i(t_k) - \hat{E}_i}{P_i}, \quad \text{and} \quad \Delta\tau_i = \frac{\hat{E}_i}{P_i}. \tag{9}$$

Clearly, at each iteration of the algorithm, the lifetime of each node may change. Such local state update rule is always triggered by a node toward its in-neighbor nodes. The dependencies of $\mathbf{\Phi}_i^{out}$ on its input parameters is implied.

Fig. 2 shows a flow diagram that explains the basic ideas behind the proposed method.

***Remark** 4.2:* Note that the conditions that trigger the execution of the optimization algorithms in the local state update rule at step 2 of Algorithm DLMA

$$\tau_i > \min_{j \in \mathcal{N}_i^{in} \bigcup \{i\}} \tau_j + \Delta\tau_i \tag{12a}$$

$$\tau_i > x_i + \Delta\tau_i \tag{12b}$$

ensure that the algorithm is not triggered whenever DLMA execution on a node is not expected to improve the network lifetime, either because its outcome cannot be better than the current one or its execution would be too demanding for that node, it is not triggered. ■

In Theorem 4.3 it is proven that at each iteration k of the Algorithm DLMA, the objective function of (2) is either improved or does not change.

***Theorem** 4.3:* During the execution of Algorithm DLMA, at each instant of time t_k in which tasks are reassigned, the global network lifetime may not decrease. It holds $\tau(t) \geq \tau(t_k), \quad \forall t > t_k$.

Proof: See Appendix B. □

Algorithm DLMA offers several advantages with respect to centralized algorithms. Both problems (10) and (11) are hard to solve. However, the computational complexity of the local optimization depends only on the number of nodes locally involved, thus despite being a MILP problem, its complexity does not grow with the size of the network and is small in absolute terms if the number of locally executed tasks is small. Additionally, since the allocation of processing is dynamic, the algorithm reacts to unexpected drops in battery charge by locally changing

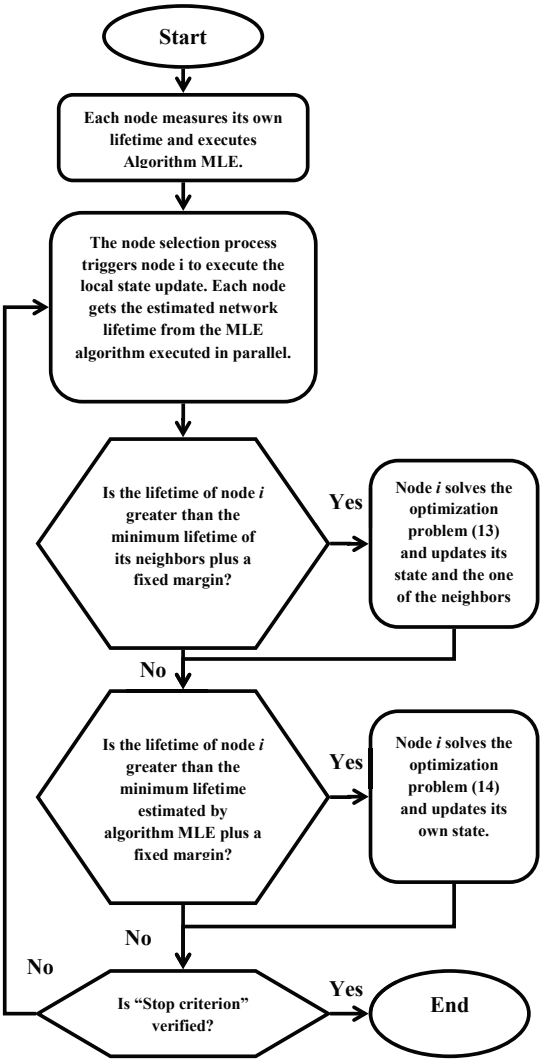


Fig. 2: Flow diagram of Algorithm DLMA.

the status of the nodes involved, a centralized approach would require a full scale optimization with intractable complexity to address this issue.

Remark on changes to network lifetime due to external causes. Theorem 4.3 states that the network lifetime is improved continuously by the execution of Algorithm DLMA. It implies that if the network lifetime changes abruptly, for instance due to a sudden drop in battery charge due to a fault, the proposed approach will continue to locally reallocate tasks improving the new network lifetime seamlessly since no global information is exploited in the optimization process. To optimize resource usage, the Algorithm DLMA exploits a stop criterion not to execute unnecessary iterations. However, if a fault occurs, it is detected by either surrounding or faulty nodes which then should trigger a restart procedure of the algorithm.

Node selection processes. We now discuss two node selection strategies to be implemented at step 1 of Algorithm DLMA to identify which node executes a local update rule. The first, a *deterministic* node selection process, is defined by function $\mathbf{v}(t) : \mathbb{R}^+ \to X \bigcup \{\varnothing\}$ that at each step k selects the out-neighbors of the nodes chosen at step $k-1$. The root node executes the local state update periodically with period T, this triggers an avalanche of local state updates involving first its neighbors and then all other nodes. The period T in which the root node triggers its neighbors controls the frequency of local optimizations in the network. Since the graph is acyclic, once leaf nodes are triggered the local optimizations stop.

The second, a *stochastic* node selection process, is defined by a random stochastic process $\mathbf{v}(t) : \mathbb{R}^+ \to X \bigcup \{\varnothing\}$, which assigns to each node a probability p_i (with $i = 1, \ldots, N$) of activating a local optimization at each instant of time t.

The advantages of the deterministic node selection process are that it terminates in finite time, gives complete control over frequency of local optimization to the root node and ensures that local optimizations are triggered evenly throughout the network. The advantages of the stochastic node selection process are that it is simpler to implement and does not require time or event synchronization in the network.

In Section V these different approaches to node selection are compared through simulations and experimental results.

Stop criterion. We now discuss a stop criterion for Algorithm DLMA.

***Definition** 4.4 (Stop Criterion):* Consider a deterministic node selection process such that in at most every DT units of time every node has executed the local update according to Algorithm DLMA. Then, for each node i selected to perform Algorithm 2, if $x_i(t) = x_i(t')$ for $t \in \left[t', t' + \frac{D(D+3)}{2} \cdot T\right]$, assign to node i $Stop\ criterion := True$ and stop the execution of Algorithm DLMA at node i. ■

Next, it is shown that when the stop criterion in Definition 4.4 is satisfied any further execution of Algorithm DLMA cannot improve the network lifetime.

***Proposition** 4.5:* During the execution of Algorithm DLMA and Algorithm MLE if a deterministic node selection process is implemented and each node verifies the conditions for the stop criterion in Definition 4.4 then Algorithm DLMA cannot further improve the network lifetime.

Proof: See Appendix C. □

Computational complexity and optimization cost. The local state update rule at step 2 of Algorithm DLMA involves the solution of a Mixed Integer Linear Programming (MILP) problem. It is known that to solve it is required a number of operations that grows exponentially with respect to the number of variables. The feasibility of solving the considered MILP problem in an embedded system such as a sensor node, depends on the number of processing tasks to be considered. In the worst case, the number of required operations is at most equal to the number of operations required to perform an exhaustive search between all the possible combinations of processing assignments. Let E^o denote the energy required to evaluate one of the objective functions of the local state update rule of DLMA for a given task assignment and verify that the constraints of the local state update rule are satisfied.

The number of possible processing assignments (pa) in the generic in-neighborhood $\mathcal{N}_i^{in}$ of node i, when it performs the local state update, is $N_{i,pa} = 2^{\mathbf{1}^T \sum_{j \in \mathcal{N}_i^{in} \cup \{i\}} \sum_{\lambda \in T^p} d_{j\lambda}}$. The energy cost to perform the state update on node i is at most

$$\hat{E}_i \leq E^o \times N_{i,pa} = E^o \times 2^{\mathbf{1}^T \sum_{j \in \mathcal{N}_i^{in} \cup \{i\}} \sum_{\lambda \in T^p} d_{j\lambda}}. \tag{13}$$

Remarks on packet loss, time delays and dynamic topology. Packet loss and time delays do not affect the algorithm performance since only asynchronous local optimizations which require only local information are exploited. The order in which optimizations are executed does not affect the end result of the algorithm. If a state update fails due to communication failures due to significant packet loss, then the state update is simply not executed and this does not affect the global algorithm performance as long as eventually, after a sufficiently long time, a state update involving the same nodes is successful. Therefore, one of the advantages of our approach is that it is robust with respect to packet loss and time delays. Furthermore, since the proposed method does not require knowledge of network topology, if it changes during execution it does not affect the algorithm performance. This holds true as long as the network is connected and topology changes do not occur with frequency greater than the convergence time of Algorithms MLE which is in the order of seconds as shown in Table IV in the proposed experimental results in Section V.

V. Simulations and Experimental results

In this section, the scenario of an urban environment, where nodes have been positioned along the streets as shown in Fig. 3, is considered. Solid circles represent Arduino UNO [19] nodes equipped with sensors for speed measurement of the vehicles passing through; speed measurements are sent every 2 minutes and are 64-bit long. Empty circles are smart Arduino DUE [20] nodes not equipped with sensors but able to execute more processing operations. We considered an application that provides to drivers information on which the best path from the "Start" to the "Destination" point is, in terms of minimum travel time, based on the speed information collected by sensor nodes. Nodes communicate using the XBee-PRO DigiMesh 2.4 [21] radio interfaces on the 2.4 GHz ISM

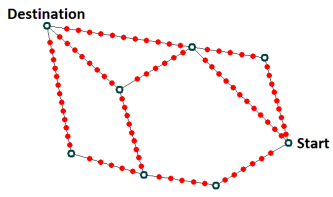


Fig. 3: Network topology. Solid circles represent sensor nodes, empty circles represent smart nodes.

frequency band. The characteristic parameters of the considered hardware can be found in [19], [20], [21]. We tested the resulting scenario implementing the proposed algorithms on the Arduino nodes and compared the performance of the DLMA algorithm against three alternative methods denoted as C, CH and CO in terms of network lifetime improvement:

- method C: All processing tasks are assigned to a Coordinator (the sink) (no intermediate processing);
- method CH: Processing tasks are given a priori to Cluster Heads found in the path towards the Coordinator (some intermediate processing);
- method CO: Centralized optimization proposed in [8].

The considered application has been divided into 11 sensing tasks and 37 processing tasks. A *speed sensing* task related to a particular stretch is assigned to a sensor node only if the sensing node is placed in that stretch. Furthermore, since knowledge of the mean travelling time for each stretch is needed in order to compare them to each other, 11 *mean speed computing* processing tasks and 11 *mean travelling time computing* processing tasks have been considered, one for each stretch of street. To find the best path, we sum the mean travelling times of all the combination of stretches that can be driven one after the other, and compare them. Therefore, the remaining processing tasks are: 12 different *summation of mean travelling times for different stretches*, and 3 different *choice of the best path*. With reference to Fig. 3, solid markers represent nodes that perform the *speed sensing* and *mean speed computing* for the stretch of the street where they are placed. Empty markers are more capable nodes (with an initial battery charge three times higher than the others) which can execute all the processing tasks.

The algorithm has been preliminarily tested on a small scale network using 5 Arduino UNO and 2 Arduino DUE boards. To implement the DLMA algorithm, the GLPK (GNU Linear Programming Kit) library [22] has been used. The GLPK is a set of routines, written in ANSI C, intended for solving MILP problems. In particular, the routine to solve MIP problems based on the *branch-and-bound* method has been used (*glp_intopt*, *solve_mip* and *preprocess_and_solve_mip* functions). Starting from it, the code has been tailored to the specific DLMA problem that a node is required to execute, taking into account only those processing tasks that it is able to perform. The obtained performance has been used to profile the boards' behaviour, and accordingly simulate the algorithm on the full network, using Matlab.

Test results are shown in Table I. A significant improvement of network lifetime of 94.5% and 88.9% is observed, which is almost twice as long, when comparing our proposed method with a deterministic node selection process, against the CH and C methods. A similar improvement of 94.9% and 91.3% has been measured also if nodes are selected randomly. The centralized optimization method CO performs better by roughly 30%, but as previously discussed it requires the solution of an optimization problem whose complexity grows exponentially with the size of the network and therefore has intractable complexity in large scale networks and furthermore a centralized implementation may not be feasible. Indeed, while methods DLMA, C and CH were implemented on Arduino UNO and DUE, to implement method CO a modern desktop computer was needed.

With respect to node selection, results are quite similar, but we observe that stochastic node selection slightly outperforms the deterministic one.

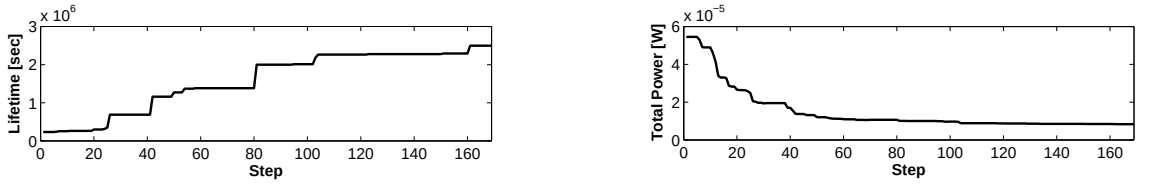
Table II shows how the two algorithms affect memory availability of a node, compared to the CO algoritm [8]. An Arduino UNO board represents a sensor node while an Arduino DUE represents a smart node. Since the amount of resources needed by the CO method was too high for an Arduino node, it has been run on an Intel Core i7 2.40 GHz machine with 6,00 GB RAM. Results reflect the proportionality between computational complexity and number of

tasks that nodes are able to perform. We observe that memory is not critical for MLE and DLMA execution, with the exception of the RAM required on sensing nodes, which are usually equipped with less memory and processing resources. Results show that nodes need at least 2 kB RAM to perform DLMA. Fortunately, with current technology typical WSN nodes that are able to perform simple tasks usually have more than 2 kB RAM.

In Table III results related to processing capabilities to perform DLMA compared to CO are provided. In particular, the following results are reported:a) the probability that the algorithm is triggered on a node; b) the average number of steps per node, i.e. the average time each node performs DLMA before it converges to a solution; c) the average energy spent by each node per step of the algorithm execution; d) the average time needed by a node per step of the algorithm execution. Whenever DLMA execution on a node is not expected to improve the network lifetime, either because its outcome cannot be better than the current one or because its execution would be too demanding for that node, it is not triggered. From test results we estimated that the probability that DLMA is triggered on a node is more than 95%: its execution is not convenient only less than 5% of the times. The deterministic node selection mechanism seems to lead to a lower amount of steps, and consequently it requires lower processing costs. Even though processing capabilities are much lower on the considered sensing and smart nodes than those of the computer used to run method CO (respectively 0.67% and 3.5%), energy cost due to processing resulted to be much lower with DLMA rather than method CO.

In Table IV, the average convergence times t_{MLE} and t_{DLMA} to perform algorithms MLE and DLMA are compared to the related average time t_{CO} to execute method CO. These results are computed taking into account not only execution time, but also signalling transmission and reception time due to the algorithms. Note that convergence time depends on the optimization complexity, which depends on the number of variables, and consequently on the number of nodes and tasks involved in the optimization. Although this number is quite high for the reference scenario, the total amount of time for the algorithms to converge is still low, especially if compared with centralized algorithm CO. In fact, the DLMA complexity only depends on the number of processing tasks that a node can perform (1 for sensing nodes, 4 for smart nodes) and the number of incoming neighbors. On the other hand, the CO complexity depends on the whole number of tasks and nodes that the network is made of.

It needs to be noticed that conditions (12a) and (12b) ensure that the local optimization algorithm is never started if its outcome cannot be better than the current one or if its execution would be too demanding for that node. This implies that the local optimization is not executed if it involves a significant number of nodes and tasks and the energy reserve is low. Therefore, only local optimization which take little time to be solved are likely to be executed thus shortening the convergence time of DLMA. The monotonic increase of lifetime is attested by the simulation in Fig. 4(a), where an example of the network lifetime improvement for each DLMA step is shown, with the network lifetime evaluated according to Eq. 1. Note that, for the sake of simplicity, in this figure only the power consumption due to the tasks assigned to the nodes is taken into account, disregarding other causes of power consumption (e.g. sleep-wake cycle). Each step corresponds to the local optimization computed on one node. The same simulation of DLMA execution is used to compute the expected total power consumption in Fig. 4(b): we can observe that the expected total power consumption decreases while DLMA converges, though not monotonically.



(a) Network lifetime

(b) Total Power Consumption

Fig. 4: Example of expected network lifetime and total power consumption improvement during DLMA execution

VI. Conclusions

In this paper the problem of deploying distributed applications in a heterogeneous WSN has been investigated. We proposed a distributed approach to maximize the network lifetime by distributing efficiently the workload among sensor nodes. The proposed algorithm requires only asynchronous local state updates and exploits local information. We provided simulations and experimental results on Arduino boards that show significant advantages with respect to approaches where data is processed only by the Coordinator, by Cluster Heads or a centralized off-line optimization is performed.

References

[1] P. Spachos, P. Chatzimisios, and D. Hatzinakos, "Energy aware opportunistic routing in wireless sensor networks," in *IEEE Globecom Workshops*, 2012, pp. 405–409.

[2] K. Lin, M. Chen, S. Zeadally, and J. J. Rodrigues, "Balancing energy consumption with mobile agents in wireless sensor networks," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 446–456, 2012.

[3] G. Boggia, P. Camarda, M. Castellano, O. Fiume, L. Grieco, and S. Mascolo, "A collision free mac protocol for energy saving in wireless ad hoc networks," in *Int. Workshop on Wireless Ad-Hoc Networks*, 2004.

[4] G. Boggia, P. Camarda, F. A. Favia, L. A. Grieco, and S. Mascolo, "Providing delay guarantees and power saving in ieee 802.11e network," in *Int. Conference on Wired/Wireless Internet Communications*, 2005.

[5] N. Edalat, W. Xiao, C. Tham, E. Keikha, and L. Ong, "A price-based adaptive task allocation for wireless sensor network," in *IEEE 6th Int. Conference on Mobile Adhoc and Sensor Systems*, 2009.

[6] Y. Jin, J. Jin, A. Gluhak, K. Moessner, and M. Palaniswami, "An intelligent task allocation scheme for multihop wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 3, pp. 444–451, 2012.

[7] J. He, L. Duan, F. Hou, P. Cheng, and J. Chen, "Multiperiod scheduling for wireless sensor networks: A distributed consensus approach," *IEEE Transactions on Signal Processing*, vol. 63, no. 7, pp. 1651–1663, 2015.

[8] V. Pilloni and L. Atzori, "Deployment of distributed applications in wireless sensor networks," *Sensors*, vol. 11, no. 8, pp. 7395–7419, 2011.

[9] M. Franceschelli, D. Rosa, C. Seatzu, and F. Bullo, "Gossip algorithms for heterogeneous multi-vehicle routing problems," *Nonlinear Analysis: Hybrid Systems*, vol. 10, no. 1, pp. 156–174, 2013.

[10] S. Chopra and M. Egerstedt, "Heterogeneous multi-robot routing," in *IEEE American Control Conference*, 2014.

[11] M. Franceschelli, A. Giua, and C. Seatzu, "Fast discrete consensus based on gossip for makespan minimization in networked systems," *Automatica*, vol. 56, no. 0, pp. 60 – 69, 2015.

[12] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52 (6), pp. 2508–2530, 2006.

[13] K. Cai and H. Ishii, "Quantized consensus and averaging on gossip digraphs," *IEEE Transactions on Automatic Control*, vol. 56, no. 9, pp. 2087–2100, Sept 2011.

[14] C. Hadjicostis and T. Charalambous, "Average consensus in the presence of delays in directed graph topologies," *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 763–768, 2014.

[15] A. Dimakis, A. Sarwate, and M. Wainwright, "Geographic gossip: Efficient averaging for sensor networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 3, pp. 1205–1216, March 2008.

[16] C. S. L. S. Madan, R. and A. Goldsmith, "Cross-layer design for lifetime maximization in interference-limited wireless sensor networks," in *24th IEEE Conference on Computer Communications, 2005*.

[17] Q. Wang, M. Hempstead, and W. Yang, "A realistic power consumption model for wireless sensor network devices," in *IEEE SECON*, 2006.

[18] J. He, P. Cheng, L. Shi, J. Chen, and Y. Sun, "Time synchronization in wsns: A maximum-value-based consensus approach," *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 660–675, 2014.

[19] Arduino, "Arduino UNO," http://arduino.cc/en/Main/ArduinoBoardUno.

[20] ——, "Arduino DUE," http://arduino.cc/en/Main/ArduinoBoardDue.

[21] Digi, "XBee-PRO DigiMesh 2.4 datasheet," http://www.digi.com/pdf/ds_xbeedigimesh24.pdf.

[22] A. Makhorin, "Glpk (gnu linear programming kit)," 2008.

APPENDIX

A. Proof of Proposition 4.1

When the root node sends the first message to its in-neighbors, after $T \cdot d$ units of time all the nodes whose minimum path from the root node is less than or equal to d have performed a local state update, setting their state to the minimum value found along the path. If $\mathcal{G}_{\mathcal{X}}$ is connected then all nodes will be triggered. If it is acyclic, when the furthest nodes at distance $\bar{D}$ from the root are reached by these messages the algorithm stops. Thus, after $T \cdot \bar{D}$ units of time, variable x_i of each node corresponds in value to the minimum node lifetime of nodes along the path that connects node i with the root node, i.e., set $\mathcal{X}_i$. □

B. Proof of Theorem 4.3

Let $V(t) = \frac{1}{\tau(t)} = \min_{i \in X} \frac{P_i(\boldsymbol{S})}{\gamma_i(t)}$ be the inverse of the minimum lifetime in the network at time t, held by nodes with the smallest ratio between power consumption and energy reserve. During the algorithm execution, at each iteration k two cases may occur. **Case 1**: the state matrix $\boldsymbol{S}$ of the network does not change, then $V(t_{k+1}) = V(t_k)$ by definition. **Case 2**: the state matrix $\boldsymbol{S}$ changes according to the solution of the local optimization problem (10)-(11). The solutions of problems (10) and (11) minimize locally the power consumption for the node with the shortest lifetime between nodes i and $j \in \mathcal{N}_i$ only if the lifetime of the node performing the optimization is greater than the minimum lifetime in its neighborhood by at least $\Delta\tau_i$. Thus, the state $\boldsymbol{S}$ is updated only if $\min_{j \in \mathcal{N}_i^{in} \bigcup \{i\}} \frac{\gamma_j}{P_j(s_k : k \in \mathcal{N}_i \bigcup \{i\})} > \min_{j \in \mathcal{N}_i^{in} \bigcup \{i\}} \tau_j(\bar{\boldsymbol{S}})$ causing $V(t) < V(t_k)$ for $t > t_k$, or $\tau_i > x_i + \Delta\tau_i$ causing $\sum_{j \in \mathcal{N}_i^{in}} \boldsymbol{\Phi}_j^{out}$ to be minimized. Since $x_i(t_k) \leq \min_{j \in S_i} \tau_j$ thanks to the execution of Algorithm MLE, the constraints of the optimization ensure that locally the lifetime is not reduced to less than the minimum lifetime among nodes in $\mathcal{X}_i$. A local state update is performed only if each information flow passing from node i to nodes in $\mathcal{N}_i^{out}$ does not increase, i.e., $\boldsymbol{\Phi}_i^{out} \leq \bar{\boldsymbol{\Phi}}_i^{out}$. The transmission cost of nodes in $\mathcal{N}_i^{in}$ can only decrease as $\boldsymbol{\Phi}_i^{out}$ is decreased, as shown in (7), and their processing cost cannot increase, as shown in (6). Nodes in the downstream path towards the sink receive a smaller information flow thus consume less power. The nodes in the upstream path

have their information flow unchanged. Thus, after each local state update, the global network lifetime may only improve, i.e., $\tau(t) \geq \tau(t_k)$ for $t > t_k$. □

C. Proof of Proposition 4.5

Consider the case in which the conditions for the stop criterion in Definition 4.4 are verified and the network is executing a deterministic node selection process.

Whenever the minimum network lifetime is improved, due to Proposition 4.1 after at most $T\bar{D}$ units of time all nodes interested by the change become aware of it. Thus, if any node in the network does not change its state nor detects a change of state of its neighbors or on its estimation of the minimum network lifetime for at least $\frac{\bar{D}(\bar{D}+1)}{2}T + T\bar{D}$ units of time it implies that each node in the network already attempted a local state update without changing its own state. Therefore, no further execution of Algorithm DLMA may change the state of the network and consequently cannot improve the current minimum network lifetime. □

Algorithm: Decentralized Life Maximization for WSNs (DLMA)

Each node $i \in X$ is initialized with the residual energy of the battery $\gamma_i(t_0)$ and state $s_{i\lambda} = \{0\}$, $\forall \lambda \in T^p$ (no processing assigned). Let $k = 0$ and $t_0 = 0$.

while *Stop criterion=False* **do**

1 - According to the node selection process $\mathbf{v}(t_k)$, node i is triggered at time t_k. Node i, with current state $\bar{\boldsymbol{s}}_i$ and output traffic flow $\bar{\boldsymbol{\Phi}}_i^{out}$ enquires the state $\bar{\boldsymbol{s}}_j$ and input traffic $\bar{\boldsymbol{\Phi}}_j^{in}$ for all $j \in \mathcal{N}_i^{in}$.

2 - Node i executes the following **Local state update rule:**

• **if** $\tau_i > min_{j \in \mathcal{N}_i^{in} \bigcup \{i\}} \tau_j + \Delta\tau_i$, **then**

solve the next local MILP problem:

$$
\begin{aligned}
&\min \quad \alpha \\
&s.t. \\
&\tfrac{P_j(\boldsymbol{s}_k : k \in \mathcal{N}_i^{in} \bigcup \{i\})}{\gamma_j(t_k)} < \alpha \quad && \forall j \in \mathcal{N}_i^{in} \\
&\tfrac{P_i(\boldsymbol{s}_k : k \in \mathcal{N}_i^{in} \bigcup \{i\})}{\gamma_i(t_k) - \hat{E}_i} < \alpha \\
&\boldsymbol{d}_j \geq \boldsymbol{s}_j && \forall j \in \mathcal{N}_i^{in} \bigcup \{i\} \\
&\boldsymbol{\Phi}_i^{out} \leq \bar{\boldsymbol{\Phi}}_i^{out} \\
&\alpha \in \mathbb{R}^+ \\
&s_{i\lambda} \in \{0,1\} && \forall j \in \mathcal{N}_i^{in} \bigcup \{i\}, \lambda \in T^p
\end{aligned}
\tag{10}
$$

and denote the solution α_{opt}, $\boldsymbol{s}_{opt,k} : k \in \mathcal{N}_i^{in} \bigcup \{i\}$. Set the new assignments $\boldsymbol{s}_j = \boldsymbol{s}_{opt,j}$, $\forall j \in \mathcal{N}_i^{in} \bigcup \{i\}$.

• **if** $\tau_i > x_i + \Delta\tau_i$, **then**

let $\alpha = \max_{j \in \mathcal{X}_i} \frac{1}{\tau_j}$ and solve the following local MILP problem:

$$
\begin{aligned}
&\min \quad \boldsymbol{\Phi}_i^{out}(\boldsymbol{\Phi}_j^{in}, \boldsymbol{s}_j, \boldsymbol{s}_i) \quad && j \in \mathcal{N}_i \\
&s.t. \\
&\tfrac{P_j(\boldsymbol{s}_k : k \in \mathcal{N}_i^{in} \bigcup \{i\})}{\gamma_j(t_k)} < \alpha \quad && \forall j \in \mathcal{N}_i^{in} \\
&\tfrac{P_i(\boldsymbol{s}_k : k \in \mathcal{N}_i^{in} \bigcup \{i\})}{\gamma_i(t_k) - \hat{E}_i} < \alpha \\
&\boldsymbol{d}_j \geq \boldsymbol{s}_j && \forall j \in \mathcal{N}_i^{in} \bigcup \{i\} \\
&\boldsymbol{\Phi}_j^{out} \leq \bar{\boldsymbol{\Phi}}_j^{out}, && \forall j \in \mathcal{N}_i^{in} \\
&s_{i\lambda} \in \{0,1\} && \forall j \in \mathcal{N}_i^{in} \bigcup \{i\}, \lambda \in T^p
\end{aligned}
\tag{11}
$$

and denote the solution $\boldsymbol{s}_{opt,j} : j \in \mathcal{N}_i^{in} \bigcup \{i\}$. Set new assignments to $\boldsymbol{s}_j = \boldsymbol{s}_{opt,j}$, $\forall j \in \mathcal{N}_i^{in} \bigcup \{i\}$.

3 - Let $k = k + 1$.

TABLE I: Improvement of network lifetime due to the DLMA algorithm with respect to methods C, CH and CO

	Deterministic selection	**Stochastic selection**
DLMA-C	93.4%	94.9%
DLMA-CH	88.9%	91.3%
DLMA-CO	-32.2%	-27.4%

TABLE II: Memory usage per node required by MLE and DLMA algorithms, compared with CO

	MLE + DLMA		**CO**
	Sensing node	**Smart node**	
Memory req. to store data	19.3 kB (60.3%)	20.6 kB (4.0%)	0.39 MB
Max RAM req.	1.8 kB (90%)	15.6 kB (16.3%)	13.9 MB

TABLE III: Resource usage per node: DLMA compared with CO

		DLMA		**CO**
		Det. sel.	**Sto. sel.**	
Trigger Probability		95.0%	95.8%	–
Number of steps		2.4	3.0	–
Energy per step	**Sensing node**	0.25 μJ		16.3 J
	Smart node	2.04 μJ		
Time per step	**Sensing node**	158 μsec		377 sec
	Smart node	239 μsec		

TABLE IV: Average conv. time (sec) of MLE, DLMA and CO

	Deterministic selection	**Stochastic selection**
t_{MLE}[s]	0.10	0.13
t_{DLMA}[s]	0.65	0.83
t_{CO}[s]	377	