View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Archivio istituzionale della ricerca - Università di Cagliari

Cooperative Task Assignment for Distributed Deployment of Applications in WSNs

Virginia Pilloni*, Pirabakaran Navaratnam†, Serdar Vural†, Luigi Atzori*, Rahim Tafazolli†
*DIEE, University of Cagliari, Italy {*virginia.pilloni,l.atzori*}*@diee.unica.it*
†CCSR, University of Surrey, Guildford, UK {*p.navaratnam,s.vural,r.tafazolli*}*@surrey.ac.uk*

***Abstract*—Nodes in Wireless Sensor Networks (WSNs) are becoming more and more complex systems with the capabilities to run distributed structured applications. Which single task should be implemented by each WSN node needs to be decided by the application deployment strategy by taking into account both network lifetime and execution time requirements. In this paper, we propose an adaptive decentralised algorithm based on non-cooperative game theory, where neighbouring nodes negotiate among each other to maximize their utility function. We then prove that an increment of the nodes utility corresponds to the same increment of the utility for the whole network. Simulation results show significant performance improvement with respect to existing algorithms.**

***Index Terms*—Wireless Sensor Networks, network lifetime, game theory.**

I. Introduction

The great progress in Wireless Sensor Networks (WSNs) nodes' technology, both in terms of processing capability and energy consumption reduction, has made them more complex systems capable not only of gathering information about the monitored environment, but also of making decisions and acting upon them. These developments have contributed in considering WSNs as one of the pillars of the Future Internet, where they interface with other technologies in order to create an horizontal ambient intelligent infrastructure wherein the sensing, computing and communicating infrastructure is set with a programmable middleware that allows for quickly deploying different applications running on top of it.

Since nodes in a WSN are mainly battery powered, one of the major issues related to WSNs has always been the maximization of the network lifetime. A great effort has been spent by researchers into this problem, taking into account many different approaches. Only recently the attention has been drawn on the development of algorithms that, starting from the partition of applications into small tasks, are able to find more rational task assignments that could help in improving the network performance, be it the improvement of the network lifetime or the reduction of the completion time to perform the application assigned to the network.

In this paper, an adaptive distributed algorithm for the improvement of the network lifetime and the contemporary reduction of the application completion time is proposed. We consider a heterogeneous hierarchical network, where neighbouring nodes negotiate among each other following the rules of non-cooperative game theory to maximize their own utility function. In the following, we will prove that the game under examination is a potential game, which means that every improvement of the nodes' utility corresponds to the same improvement in the utility perceived by the whole network, and this implies that the problem has a unique outcome that is reachable in a finite time. Simulation results show significant performance improvement for both overall energy consumption and completion time in a typical reference scenario, with respect to gateway-oriented and DLMA-oriented task assignment approach presented in [1].

This paper is organized as follows. The second section provides the preliminaries; the third section introduces the problem and adopted approach; the fourth section describes the task assignment model whereas section five describes the resulting algorithm; the following section presents some simulation results and conclusions.

II. Preliminaries

A. Past Studies

Since nodes in WSNs are mainly battery powered, a great effort has been spent by researchers in finding strategies to increase the nodes' battery lifetime. Many different approaches have been studied to come to the same goal: convenient deployment of sensors [2], use of efficient routing techniques [3], use of relay nodes that help in balancing the network energy consumption among nodes [4] are some examples.

The increased processing capabilities of modern nodes have made them suitable for more complex applications, such as those of the forthcoming Future Internet [5]. For this reason, network lifetime optimization is not only centred on the reduction of the transmission power anymore, but convenient processing could also reduce the amount of data to be delivered to the Gateway, thus reducing the transmission energy consumption. This is the principle that lies behind LEACH [6], where sensors serve as Cluster Heads aggregating data and, indeed, decreasing the amount of data sent over the network.

A step forward could be taken considering not only aggregation, but also any processing task that could be performed on data, on the basis of network topology, battery power, and node processing capabilities. In [7] the maximization of the clusters lifetime is proposed. This approach has two main limitations: it focuses on homogeneous networks, that are not common in real scenarios, and it considers only communication tasks. An adaptive task allocation that aims at reducing the overall energy consumption by maximizing energy balancing was introduced in [8]. However, this mechanism requires the

exchange of messages among all the nodes in the network, indeed considerably increasing the overhead. [9] proposes an overlaying framework that determine the tasks assignment to WSN nodes by means of a centralised optimization algorithm aimed at maximizing the network lifetime. In [1], the same problem is faced using a distributed algorithm. However, one major drawback of these studies is that they do not take into account the network application deadline.

Completion time reduction is studied in [10], where an algorithm that improves the network lifetime while reducing the tasks execution time is proposed for a homogeneous network. The algorithm designed in this work is a centralised one, which thus lacks in scalability and dynamism.

In this paper, we propose a framework for a heterogeneous hierarchical WNS, which aims at maximizing the network lifetime and minimizing the application completion time in a decentralized fashion. The nodes of this network negotiate the best solution following the rules of non-cooperative game theory [11]. In the following, we will prove that the proposed algorithm can come to a good solution in a few simple steps.

B. Energy Consumption

There are several mechanisms that contribute in reducing the amount of residual energy for the nodes in a WSN. Since our goal in this work is the convenient assignment of tasks to the nodes, we focus on the mechanisms that are directly related to the execution (or not) of the tasks, that are processing energy consumption and communication energy consumption.

Processing energy consumption e_{proc} depends on the complexity of a *task*, that is, the number of instructions I_{task} needed to perform it, and on the average energy consumption per executed instruction $e_{instr}(x)$, that is determined on the basis of node x datasheet

$$e_{proc}(task, x) = I_{task} \times e_{instr}(x) \tag{1}$$

There are two main components that contribute in communication energy consumption: transmission and reception energy consumption [12]. We define the energy per bit necessary to send and receive data from node x to node y (one-hop neighbours) at a constant rate R, respectively as

$$\begin{aligned} e_{tx}(x,y) &= \frac{1}{R}\left(P_{T0_x} + \frac{\varphi_{xy} \times \delta_{xy}^{\gamma}}{\eta_x}\right) \\ e_{rx}(y) &= \frac{P_{R0_y}}{R} \end{aligned} \tag{2}$$

where: P_{T0_x} and P_{R0_y} are the power consumption components of the transmitting and receiving circuitry, for node x and node y respectively; η_x is the drain efficiency of the Power Amplifier in x; φ_{xy} is a coefficient proportional to the minimum reception power and the characteristic parameters of the antennas; δ_{xy} is the distance between transmitter and receiver; γ denotes the path loss component. These expressions are described in more detail in [12].

III. Problem Formulation

The reference scenario considered in this paper is that of a hierarchical heterogeneous WSN. This WSN needs to accomplish a determined application, given by the execution of a certain number of tasks by some of its nodes. Being a heterogeneous network, some nodes can perform the same task faster than others, or spending less energy. The objective of the algorithm described hereinafter is to get an application performed by the network, reducing the processing time and extending the network lifetime at the same time, by assigning the tasks to the nodes of the network in an adaptive and distributed way: whenever a node receives some data, along with the information of which tasks have to be performed on those data, it initiates a negotiation among its neighbours. In this negotiation, based on a non-cooperative game theory approach, each node decides whether it should perform some tasks or not, depending on the contribution it could give to the network in terms of faster processing time and lifetime improvement.

In our modelling, the network can be described as a Directed Acyclic Graph (DAG) $\mathcal{G}_X = (X, E_X)$, where the vertices are the nodes $X = \{1, \ldots, i, \ldots, N\}$, while the connections between each pair of nodes (i, j) is described by the set of edges $E_X = (e_{ij})$, where each edge represents a connection from node i to node j.

Since the network is hierarchically organised, a level is associated to each node i in X. At the top of the hierarchy is the Gateway (GW), which collects data received from the Cluster Heads (CHs). CHs reach the GW using a Cluster Based routing protocol [13]. Sensor Nodes (SNs) are at the lowest level of the hierarchy. They are grouped in clusters in such a way that each SN is one hop away from all the other SNs constituting the cluster, and from the CH to which it is associated. No communication is allowed between SNs belonging to different clusters. This means that, given two nodes i and j, $e_{ij} \in E_X$ if and only if i and j are in the same cluster. Such an architecture allows to reduce the overhead of the negotiations on the network: when a negotiation is started, all the nodes involved in this negotiation can directly communicate with each other without intermediaries, thus reducing the number of messages exchanged.

Taking into account what has just been said, each node i in X is characterised by:

- its hierarchical level $L(i)$;
- its CH to which it is associated $CH(i)$, with $CH(i) = i$ in case it is a CH;
- the completion time per single instruction $t_{instr}(i)$, that is the time needed by node i to perform the simplest instruction that its microprocessor can execute;
- the energy per single instruction $e_{ins}(i)$, that is the energy spent by node i to perform the simplest instruction that its microprocessor can execute, as defined in Section II-B;
- the vector $\boldsymbol{e}_{tx}(i) = (e_{tx}(i,j))$, where each element $e_{tx}(i,j)$ is the energy per bit spent to send data from node i to an adjacent node $j \in X : (i,j) \in E_X$, as described in Section II-B;
- the energy per bit spent to receive data $e_{rx}(i)$
- the residual battery capacity $e_{res}(i)$ of node i.

We suppose that a given application is assigned to the WSN. Since every application can be decomposed into a set of tasks, it can be described as a Directed Graph (DG) of

tasks $\mathcal{G}_T = (T, E_T)$, where $T = \{1, \dots, \lambda, \dots \Lambda\}$ is the set of tasks, while $E_T = (e_{vw})$ is the set of edges, with each edge e_{vw} representing a unidirectional data transfer from task v to task w. We suppose that every node in the network knows which is the required application and the relations among the tasks in which it is decomposed. This information could have been given initially to the nodes by the GW.

As to the tasks, a binary strategy vector $\boldsymbol{s}(i) = (s(i, \lambda))$ can be assigned to each node i in the network (the meaning of strategy will be better explained in Section IV), where $s(i, \lambda)$ is the current state of node i with reference to task λ, that is $s(i, \lambda)$ is equal to 1 when node i is performing task λ. Since the nodes are heterogeneous, different configurations of strategies of all the nodes in the network correspond to different tasks completion times and energy consumption. We will return to this concept in the following sections.

We can distinguish two groups of tasks: already performed and still to be performed. The set of already performed tasks is defined as $T_{prev} = \{1, \dots, h, \dots, H\}$, where each task has already been assigned to a node. This means that, for each task h, there is a node i for which its state $s(i, h)$ is equal to 1, and it cannot be changed. We assume that source tasks, that are the tasks that do not receive any input but return an output, are already assigned to the nodes. Hence, at the initial time $t = t_0$, T_{prev} is only populated with source tasks.

Furthermore, we define the set of tasks to be performed $T_{next} = \{1, \dots, k, \dots, K\}$, each of them characterised by:

- the deadline for successfully completing a task $t_d(k)$;
- a number of required single instructions $I(k)$.

Not all the nodes have the same abilities to perform some tasks. For this reason, we define a binary vector $\boldsymbol{d}(i) = (d(i, \lambda))$, where the element $d(i, \lambda)$ is equal to 1 if and only if node i is able to perform task λ. This means that if node i is not able to perform task λ, it cannot be assigned to the node: $\forall i \in X$ and $\forall \lambda \in T$, $d(i, \lambda) \geq s(i, \lambda)$.

IV. The Task Assignment Model

In this section, our task allocation problem will be defined as a non-cooperative game [14]. With reference to our model described in the previous Sections, a non-cooperative game is defined by the tuple $\Gamma = \langle X, \{\boldsymbol{s}(i), u_i\}_{i \in X} \rangle$: given the strategy vector $\boldsymbol{s}(i)$ assigned to node $i \in X$, a utility function $u_i : \boldsymbol{s}(i) \to \Re$ is associated to it. The goal of each node is to maximize its own utility (payoff) in a selfish and rational way. Therefore, a strategy $\boldsymbol{s}^*(i)$ will be preferred to a strategy $\boldsymbol{s}(i)$ if and only if $u_i(\boldsymbol{s}^*(i)) > u_i(\boldsymbol{s}(i))$. For simplicity of notation, we will often use $\boldsymbol{S} = \bigcup_{i \in X} \boldsymbol{s}(i)$ to refer to the strategy of all the nodes in the network.

A. Task Utility Function

Since the utility of a node is strictly bound to the utility for completing or not some tasks, before defining the node utility function, that is the function to be maximized by each node, we first need to define the utility function for a task $k \in T_{next}$ given the overall strategy $\boldsymbol{S}$

$$u_k(\boldsymbol{S}) = \max_{i \in X} \{[\Omega_t(i, k) + \alpha \times \Omega_\tau(i, k, \boldsymbol{S})] \times s(i, k)\}, \quad \text{with } s(i, k) \leq \prod_{l \in \mathcal{T}_{in}(k)} \sum_{j \in X} s(j, l) \tag{3}$$

where: $\Omega_t(i, k)$ is the component of the utility function related to the completion time of task k when it is performed by node i; $\Omega_\tau(i, k, \boldsymbol{S})$ is the component related to the lifetime of the network when task k is performed by node i, according to the nodes' strategy $\boldsymbol{S}$; $\alpha > 0$ is a weighting factor that takes into account any benefit of improving the network lifetime with respect to the completion time (we will further explain the meaning of α in the following); $\mathcal{T}_{in}(k) = \{\lambda \in E_T : \exists e_{\lambda k}\}$ is the set of tasks which output is needed by task k as input (i.e., previous tasks for task k). We recall that $s(i, k)$ is the current state of node i with reference to task k. The constraint on $s(i, k)$ ensures that if any of task k's previous tasks is not performed by any node, that is if any of its inputs is not available, there is no utility in performing task k, and therefore $s(i, k)$ cannot be equal to 1. The maximum operation ensures that the task can only be performed by the node that maximizes the utility function $u_k(\boldsymbol{S})$: since the goal of the game is to maximize the utility function, that is maximizing the lifetime due to the execution of some tasks, only the node that ensures the best outcome will be chosen to perform task k.

We only define a utility function for the tasks that still need to be performed: since performed tasks are already assigned and this assignment cannot be changed, there is no need to evaluate their utility.

1) Completion Time Component: We express the completion time component as

$$\Omega_t(i, k) = \frac{t_d(k) - t_c(i, k)}{t_d(k)} \tag{4}$$

where $t_c(i, k)$ is the completion time if task k is performed by node i and $t_d(k)$ is the deadline for successfully completing task k. Calling $I(k)$ the number of single instructions for task k and $t_{instr}(i)$ the completion time per single instruction for task i, as defined in Section III, $t_c(i, k)$ is defined as

$$t_c(i, k) = \begin{cases} I(k) \times t_{instr}(i), & \text{if } t_c(i, k) \leq t_d(k) \\ t_d(k), & \text{if } t_c(i, k) > t_d(k) \end{cases}$$

2) Lifetime Component: The lifetime component is defined as follows

$$\Omega_\tau(i, k, \boldsymbol{S}) = F_p(i, k) + F_{tx}(i, k, \boldsymbol{S}) \tag{5}$$

where $F_p(i, k)$ is the component related to the changing in lifetime due to the processing needed to perform task k in node i, while $F_{tx}(i, k, \boldsymbol{S})$ is related to the changing in lifetime due to transmission (and reception) in case task k is performed in node i with strategy $\boldsymbol{S}$.

We define the processing component as

$$F_p(i, k) = -I(k) \times \frac{e_{ins}(i)}{e_{res}(i)} \tag{6}$$

with $I(k)$ number of single instructions for task k, $e_{ins}(i)$ energy per single instruction for node i, and $e_{res}(i)$ residual

battery capacity for node i, as introduced in Section III. Since processing entails an energy consumption, $F_p(i,k)$ needs to be negative in order to decrement the payoff due to the execution of task k.

Let $\boldsymbol{p}_{i\to j} = \{(i,a),(a,b),\ldots,(e,f),(f,j)\}$ be the sequence of edges in the path that connects node i to node j according to some static routing. Calling HL the hierarchically higher-level node for nodes i and j, the transmission component is

$$F_{tx}(i,k,\boldsymbol{S}) = \sum_{l\in\mathcal{T}_{in}(k)}\sum_{j\in X}\Big\{C_{tx}(p_{j\to HL},l)*s(j,l) + - C_{tx}(p_{j\to i},l)*s(j,l)\Big\} - C_{tx}(p_{i\to HL},k) \tag{7}$$

with

$$C_{tx}(p_{i\to j},k) = \sum_{(x,y)\in p_{i\to j}}\left(\frac{e_{rx}(y)}{e_{res}(y)} + \frac{e_{tx}(x,y)}{e_{res}(x)}\right)\times n(k)$$

where: $\mathcal{T}_{in}(k)$ is the set of previous tasks for task k as defined in (3); $C_{tx}(p_{i\to j},k)$ is the cost to transmit (and receive) data from node i to node j; $e_{tx}(x,y)$, $e_{rx}(y)$ and $e_{res}(x)$ are the energy per bit spent to send and receive data from node x to node y, and the residual energy of node x as specified in (2); $n(k)$ is the number of output bits for task k. The difference operation is due to the fact that we are considering a difference in lifetime between the case where task k is not performed, that is the case where input data for task k are sent directly to the hierarchically higher-level node, and the case where input data for task k are sent to node i where task k is performed, and then output data for task k are sent to the hierarchically higher-level node. It may be noted that, contrary to the processing component that is always negative, the transmission component may be positive or negative: if the cost to transmit input data for task k to the higher-level node is higher than the cost to transmit them to node i and then transmit task k output to the higher-level node, the transmission component is positive, thus increasing the task utility function (3) and making it more convenient to perform task k rather than not doing it.

3) Considerations about α parameter: Parameter α introduced in (3) is a weight for the lifetime component. Its value is chosen depending on how much we wish that the optimization increases the network lifetime rather than decreasing the execution speed: the lower α is, the more the task utility function will be influenced by the completion time component with respect to the lifetime component, and viceversa. For space reason, this topic cannot be described here further.

B. Global Utility Function

Given the task utility function defined in (3), the global utility function for the whole network is

$$u_g(\boldsymbol{S}) = \sum_{k\in T_{next}} u_k(\boldsymbol{S}) \tag{8}$$

This means that the global utility function is maximized when the sum of all task utility functions for the tasks that still need to be performed is maximized. Since the negotiation to choose the strategy that maximizes the global utility function is performed by the nodes, this maximization could only be possible if all the nodes in the network could communicate with each other. The communication overhead resulting from a negotiation of this type would entail an additional transmission cost that would counter the benefit of the maximisation itself, particularly for large networks. For this reason, we choose to let each node negotiate only with its neighbours, that are its adjacent nodes.

C. Node Utility Function

Starting from the approximation described above, the node utility function can be derived so that the global utility in (8) is increased by any increment in the node utility. The node utility function u_i can then be written as an aggregation of the marginal contributions of node i to each task, and therefore to the global utility function. This marginal contribution can be defined as a *Wonderful Life Utility* (WLU) [15], which is the difference between the task utility for a given strategy $\boldsymbol{s}(i)$ of the node i and the task utility for a null strategy $\boldsymbol{s}_0(i)$ of the node, where all the elements of $\boldsymbol{s}_0(i)$ are equal to 0, which means that the node is not contributing to the task (i.e., $\boldsymbol{s}_0(i) \triangleq \{0\}^K$)

$$mu_k(\boldsymbol{s}(i),\boldsymbol{s}(-i)) = u_k(\boldsymbol{s}(i),\boldsymbol{s}(-i)) - u_k(\boldsymbol{s}_0(i),\boldsymbol{s}(-i)) \tag{9}$$

where $\boldsymbol{s}(-i)$ is the complimentary set of $\boldsymbol{s}(i)$, that is the strategy of all the nodes in X with the exception of $\boldsymbol{s}(i)$. Of course, marginal utility is null when the node is not contributing to the task as its strategy.

The node utility function is then defined as

$$u_i(\boldsymbol{s}(i),\boldsymbol{s}(-i)) = \sum_{k\in T_{next}} mu_k(\boldsymbol{s}(i),\boldsymbol{s}(-i)) \tag{10}$$

From (9) we can infer that the marginal contribution of the node to a task is not null only if the node is contributing to the task. This means that, since the global utility function in (8) is a summation of task utility functions (3), if the node contributes to the a task, it follows that it contributes to the global utility. In other words, a change in strategy for node i that increases its utility, entails the same increment in the global utility. This property is particularly desirable because it implies that the game we are focusing on is a potential game, where the potential function is given by the global utility function. A consequence of this is that this game has at least one pure Nash equilibrium [16]: pure Nash equilibria are characterised by a unique outcome, contrary to mixed Nash equilibria where the outcome is stochastically variable. Furthermore, another property of potential games is the *Finite Improvement Property (FIP)*: every sequence of changes in strategy that improves the global utility, converges to a Nash equilibrium in finite time. The FIP ensures that many simple adaptive processes, such as the Distributed Stochastic Algorithm (DSA) [17], converge to Nash equilibria.

V. Proposed Algorithm

Given the network described in Section III, in order to improve the network lifetime while reducing the completion

time for the whole application, the global utility function defined in (8) should be maximized. As demonstrated in Section IV-C, a change in the node utility reflects in the same change in the global utility. This means that maximising the node utility function for all the nodes in the network, would reflect in a maximisation of the $u_g(\boldsymbol{S})$. In order to find the best outcome, a greedy local search algorithm such as the DSA is proposed. We use DSA because, with respect to other similar algorithms, it is proved to come to a solution quicker [18].

DSA is a synchronous algorithm; in our case, during the DSA execution the involved nodes negotiate the best strategy to maximize the global utility. At each time step, each node involved in the negotiation, that changed its strategy in the previous time step sends, to all the other involved nodes, a *strategy update* message (SUM) with its new strategy [17]. If a node receives any SUM, it has some probability p of activation to compute a new strategy that maximizes its utility (10). If the utility is already maximized by the current strategy, no changes occur. The probability of activation p is known as the *degree of parallel executions*. A particular attention has to be taken in choosing p value: the higher it is, the higher the probability that all the nodes change their strategy at the same time, thus increasing the overhead due to the algorithm, and of course its convergence time; on the other hand, low values of the degree of parallel executions could lead to an unacceptable increment of DSA's convergence time as well, because changes of strategy would be less frequent.

It seems evident that a negotiation of this type, that involves all the nodes in the network at the same time, would require exceedingly high communication costs and convergence time. For this reason, we propose a suboptimal algorithm, called Task Allocation Negotiation (TAN), where the nodes negotiate only with their neighbours. The TAN algorithm consists of the whole procedure to assign the tasks in T_{next} to the nodes in X, in order to maximize the network utility function $u_g(\boldsymbol{S})$.

The algorithm starts as soon as source tasks are performed, and runs until the set T_{next} is empty, i.e. there are no remaining tasks to be performed. Let X_{data} be the set of nodes that have some output data. Initially, X_{data} is the set of nodes that have just performed the source tasks. If the set T_{next} is not empty, i.e. if there are some remaining processing tasks that can be performed on the data, then each node $i \in X_{data}$ sends an *information* message (INFO) to its neighbours, $n(i)$. All the nodes in $n(i)$ reply by sending an INFO message with their own information to their neighbours.

An INFO message includes: (i) $\boldsymbol{s}(i)$, $e_{ins}(i)$, $e_{tx}(i)$, $e_{res}(i)$; (ii) the subset $T'_{prev} \subseteq T_{prev}$ of tasks that are already performed on the data that node i currently holds. Note that initially the only tasks that are already performed are source tasks, i.e. T'_{prev} is made of the source tasks.

After all nodes exchange INFO messages with their neighbours, the DSA algorithm is initiated at each node. Once DSA converges, each node will have chosen the strategy that maximizes the network utility function $u_g(\boldsymbol{S})$ (Equation 8).

Once that the strategy is chosen, tasks need to be executed by nodes according to their strategy, before sending data to the higher level node. Let $X_{proc} = \{x^1_{proc}, \dots, x^k_{proc}, \dots\}$ be the sequence of nodes in $n(i)$, for which the strategy of node x^k_{proc} entails the execution of processing task t^k_{proc}, and let $T_{proc} = \{t^1_{proc}, \dots, t^k_{proc}, \dots\}$ be the related sequence of processing tasks to be performed. Here, the nodes in set X_{proc} are ordered according to the order in which the tasks in set T_{proc} need to be performed. Since a node can perform more than one task, it is possible that two consecutive tasks t^k_{proc} and t^{k+1}_{proc} are executed by the same node, i.e. $x^k_{proc} = x^{k+1}_{proc}$. Each node x^k_{proc} first performs the task t^k_{proc}, and updates the sets T_{next} and T_{prev} accordingly. Then, if there are no other tasks assigned to it, node x^k_{proc} sends a *data* message (DATA) to node x^{k+1}_{proc} for which t^{k+1}_{proc} receives as input data the t^k_{proc} output data ($t^{k+1}_{proc} : e_{k(k+1)} \in E_T$. DATA messages contain the set T_{prev}, along with the output data resulting from the processing tasks. When all the tasks in T_{proc} are performed, a DATA message is sent to the higher-level node.

The maximization of the node utility function $u_i(\boldsymbol{s}(i))$ defined in Equation 10 is a Mixed Integer Linear Programming (MILP) problem. It is well known that MILP problems are optimally solved using branch-and-bound algorithms, which, in the worst case, have a complexity that grows exponentially with respect to the number of variables. However, computational complexity can be considerably reduced using sub-optimal heuristic algorithms, such as genetic algorithms [19].

VI. Performance Analysis

The proposed algorithm has been tested by means of simulations carried out for a realistic heterogeneous WSN application in a smart city context. The scenario under examination is that of an urban environment, where nodes have been positioned along the streets as shown in Figure 1. Solid markers represent nodes equipped with sensors for speed measurement of vehicles passing through; speed measurements are represented as double numerical values (64-bit long). Every stretch of street is a cluster, where the CHs are represented by the empty markers. They are more capable nodes (with an initial battery charge three times higher than the others), that do not perform any sensing task, but they are able to perform all the other tasks for the appropriate stretches.

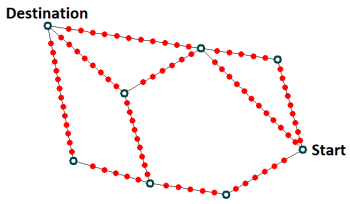

Fig. 1. Example of topology for Scenario B. Solid markers represent nodes equipped with speed sensors, while empty markers are more capable nodes which do not perform any sensing task, but can perform more complex processing tasks

Suppose that a driver, placed in the Start point, would like to know the fastest way to his Destination, based on the speed information collected by sensor nodes. We identified 89 different *speed sensing* tasks ($\lambda_1 \div \lambda_{89}$), one for each sensing node. Since we need to know the mean travelling time for each stretch of street in order to confront them to each other, we defined 78 *mean speed computing* tasks ($\lambda_{90} \div \lambda_{167}$), and

TABLE I
PERCENTAGE VALUES OF ENERGY CONSERVATION AND COMPLETION TIME GAIN USING TAN, FOR COMPARISONS TAN-C AND TAN-DLMA

	TAN-C [%]		
	$\alpha = 0$	$\alpha = \overline{\Omega}_t/\overline{\Omega}_\tau$	$\alpha = \infty$
Energy conservation	73.6	74.9	77.5
Completion time gain	22.1	16.8	11.8
	TAN-DLMA [%]		
	$\alpha = 0$	$\alpha = \overline{\Omega}_t/\overline{\Omega}_\tau$	$\alpha = \infty$
Energy conservation	10.7	12.0	14.6
Completion time gain	25.5	20.4	15.6

11 *mean travelling time computing* tasks ($\lambda_{168} \div \lambda_{178}$), one for each stretch of street. To find the best path, that is the best combination of stretches of streets that leads from the Start point to Destination, we then need to sum the mean travelling times of all the combination of stretches that can be driven one after the other, and confront them. Therefore, the remaining tasks are: 8 different *summation of mean travelling times for differents stretches* ($\lambda_{179} \div \lambda_{186}$) and 3 different *choice of the best path* ($\lambda_{187} \div \lambda_{189}$). With reference to Figure 1, solid markers represent nodes that are only allowed to perform the *speed sensing* and *mean speed computing* for the stretch of the street where they are placed (8 in total).

Nodes communicate using IEEE 802.15.4 radio interfaces on the 2.4 GHz ISM frequency band. We simulated the described scenario in a MatLab environment, where TAN was implemented along with two alternative approaches: all the data sent to the GW and processed only by the GW, that is the Start node (that is the centralised mechanism that we refer to with C); data processed according to DLMA algorithm described in [1] (DLMA approach).

The obtained results for energy consumption and completion time have been compared to those obtained using TAN algorithm. Table I shows the percentage of energy conservation and completion time gained when using TAN with respect to the alternative approaches. We refer to these comparisons as TAN-C and TAN-DLMA.

We ran simulations for: $\alpha = 0$ (null lifetime component); $\alpha = \overline{\Omega}_t/\overline{\Omega}_\tau$ (comparable Ω_t and Ω_τ, where $\overline{\Omega}_t$ and $\overline{\Omega}_\tau$ are calculated as the mean values of Ω_t and Ω_τ when every node is performing every task at the same time); $\alpha = \infty$ (null completion time component).

We observe a marked improvement of energy conservation and a good improvement in completion time gain with respect to the centralised mechanism. Good results are also observable both for energy conservation and completion time gain with respect to TAN-DLMA. It has to be noted that results for $\alpha = 0$ and $\alpha = \overline{\Omega}_t/\overline{\Omega}_\tau$ in comparison TAN-DLMA have been reported just for completeness, but DLMA algorithm does not take into account completion time, therefore the most significant results are those obtained for $\alpha = \infty$.

As we expected, when α increases, $u_g(\boldsymbol{S})$ is more over-balanced in favour of its lifetime component, and thus energy conservation percentage increases, while the completion time gain decreases; on the contrary, when α decreases, $u_g(\boldsymbol{S})$ is over-balanced in favour of its completion time component: the energy conservation percentage decreases, while the completion time gain increases.

VII. CONCLUSIONS

The algorithm presented has proven to overcome the results obtained for both C and DLMA mechanisms, and both energy conservation and completion time gain. Since results are slightly different for different values of α, it should be chosen according to the application deployment requirements.

REFERENCES

[1] V. Pilloni and L. Atzori, "A decentralized lifetime maximization algorithm for distributed applications in wireless sensor networks," in *Communications (ICC), 2012 IEEE International Conference on*, 2012, pp. 1392–1397.

[2] J. Luo, J. Panchard, M. Piórkowski, M. Grossglauser, and J. Hubaux, "Mobiroute: Routing towards a mobile sink for improving lifetime in sensor networks," *Distributed Computing in Sensor Systems*, pp. 480–497, 2006.

[3] D. Wang, B. Xie, and D. Agrawal, "Coverage and lifetime optimization of wireless sensor networks with gaussian distribution," *Mobile Computing, IEEE Transactions on*, vol. 7, no. 12, pp. 1444–1458, 2008.

[4] X. Xu and W. Liang, "Placing optimal number of sinks in sensor networks for network lifetime maximization," in *Communications (ICC), 2011 IEEE International Conference on*, 2011, pp. 1–6.

[5] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010.

[6] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, pp. 660–670, October 2002.

[7] Y. Yu and V. K. Prasanna, "Energy-balanced task allocation for collaborative processing in wireless sensor networks," *Mobile Networks and Applications*, vol. 10, pp. 115–131, 2005.

[8] N. Edalat, W. Xiao, C. Tham, E. Keikha, and L. Ong, "A price-based adaptive task allocation for wireless sensor network," in *Mobile Adhoc and Sensor Systems, 2009. MASS'09. IEEE 6th International Conference on*, 2009, pp. 888–893.

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

[10] Y. Jin, J. Jin, A. Gluhak, K. Moessner, and M. Palaniswami, "An intelligent task allocation scheme for multihop wireless networks," *Parallel and Distributed Systems, IEEE Transactions on*, pp. 444–451, 2012.

[11] K. Ritzberger, *Foundations of non-cooperative game theory*. Oxford University Press, 2002.

[12] Q. Wang and W. Yang, "Energy consumption model for power management in wireless sensor networks," in *IEEE Sensor, Mesh and Ad Hoc Communications and Networks (SECON) 2007*, 2007, pp. 142 –151.

[13] J. N. Al-karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: A survey," *IEEE Wireless Communications*, vol. 11, pp. 6–28, 2004.

[14] A. C. Chapman, R. A. Micillo, R. Kota, and N. R. Jennings, "Decentralised dynamic task allocation: A practical game-theoretic approach," in *Proc. of the 8th International Conference on Autonomous Agents and Multiagent Systems*, vol. 2, 2009, pp. 915–922.

[15] G. Arslan, J. R. Marden, and J. S. Shamma, "Autonomous vehicle-target assignment: A game-theoretical formulation," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 584–596, 2007.

[16] D. Monderer and L. Shapley, "Potential games," *Games and economic behavior*, vol. 14, pp. 124–143, 1996.

[17] W. Zhang and Z. Xing, "Distributed breakout vs. distributed stochastic: A comparative evaluation on scan scheduling," in *AAMAS-02 Third International Workshop on Distributed Constraint Reasoning*, 2002, pp. 192–201.

[18] M. Vinyals, J. Rodriguez-Aguilar, and J. Cerquides, "A survey on sensor networks from a multiagent perspective," *The Computer Journal*, vol. 54, no. 3, pp. 455–470, 2011.

[19] M. Sanna and M. Murroni, "Optimization of non-convex multiband cooperative sensing with genetic algorithms," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 5, no. 1, pp. 87–96, 2011.