PH.D. IN ELECTRONIC AND COMPUTER ENGINEERING
Dept. of Electrical and Electronic Engineering
University of Cagliari

Dynamic Deployment of Applications in Wireless Sensor Networks

Virginia Pilloni

Advisor: Prof. Luigi Atzori

Curriculum: ING-INF/03 Telecomunicazioni

XXV Cycle
2011/2012

to Sandra and Andrea

Contents

List of Figures

List of Tables

Acknowledgments

My apologies to those who are not able to understand Italian, but the following acknowledgements are written in my mother tongue, in order for me to be able to express myself better.

Prima di tutto desidero ringraziare il mio tutor, Luigi, per avermi guidata in questi anni, per i continui nuovi stimoli e suggerimenti, per avermi dato la possibilità di confrontarmi con realtà diverse, e con sfide di cui ancora aspettiamo con ansia i risultati.

Un ringraziamento particolare va a tutto lo staff dell'MCLab, per le occasioni di confronto e di discussione (specialmente per quelle accese, in sala riunioni ;)), ma soprattutto per le pause caffè e i pranzi "domenicali".

Non potrò mai ringraziare abbastanza i miei genitori, Sandra e Andrea, non solo perché senza di loro non sarei qua, ma perché sono sempre stati i miei più grandi sostenitori. Grazie per avermi insegnato che, quando si cade, la prima cosa di cui ci si deve preoccupare è trovare il modo di rialzarsi prima possibile, e che gli obiettivi sicuramente impossibili da raggiungere sono quelli per cui non si prova nemmeno a lottare.

Grazie ad Alessandro, per sopportarmi anche la mattina quando arrivo già nervosa a causa del traffico, per le mille interminabili chiacchierate, per essere sempre mio complice, per essermi sempre vicino nei momenti difficili e per condividere con me ogni piccola gioia o conquista.

Grazie ad Alessandra, la mia sorella – quasi – gemella separata alla nascita, per esserci sempre. Anche se non abbiamo lo stesso sangue, sei sempre stata accanto a me esattamente e forse anche più di come avrebbe fatto una sorella vera.

Ultimi ma non ultimi, grazie a tutti gli amici: quelli che sono un po' la mia seconda famiglia, quelli che non vedo spesso quanto vorrei, quelli che sono partiti alla ricerca di una terra un po' più fortunata, e quelli delle pause pranzo. Grazie per ogni risata insieme.

Introduction

Over the past decades, the progress in WSN technology, both in terms of processing capability and energy consumption reduction, has evolved WSNs into complex systems that can gather information about the monitored environment and make prompt and intelligent decisions. In the beginning, military applications drove the research and development of WSNs [2], with large-scale acoustic systems for underwater surveillance (e.g. Sound Surveillance System, SOSUS [3]), radar systems for the collection of data on air targets (e.g. Cooperative Engagement Capability, CEC [4]), and Unattended Ground Sensor (UGS) systems for ground target detection (e.g. Remote Battlefield Sensor System, REMBASS [5]). Typical civil WSNs are basically not complex monitoring systems, whose applications encompass environment and habitat monitoring [6][7][8][9], infrastructure security and terror threat alerts [10][11], industrial sensing for machine health monitoring [12][13], and traffic control [14][15][16][17]. In these WSNs, sensors gather the required information, mostly according to a fixed temporal schedule, and send it to the sink, which interfaces with a server or a computer. Only at this point data from sensors can be processed, before being stored.

Recent advances in Micro-Eletro-Mechanical Systems (MEMS), low power transceivers and microprocessor dimensions have led to cost effective tiny sensor devices that combine sensing with computation, storage and communication. These developments have contributed to the efforts on interfacing WSNs with other technologies, enabling them to be one of the pillars of the *Internet of Things* (IoT) paradigm [18]. In this context, WSNs take a key role in application areas such as domotics, assisted living, e-health, enhanced learning automation and industrial manufacturing logistics, business/process management, and intelligent transportation of people and goods. In doing so, a horizontal ambient intelligent infrastructure is made possible, wherein the sensing, computing and communicating tasks can be completed using programmable middleware that enables quick deployment of different applications and services.

One of the major issues with WSNs is the energy scarcity, due to the fact that sensors are mainly battery powered. In several cases, nodes are deployed in hostile or unpractical environments, such as underground or underwater, where replacing battery could be an unfeasible operation. Therefore, extending the network lifetime is a crucial concern. Lifetime improvement

has been approached by many recent studies, from different points of view, including node deployment [19][20][21], routing schemes [22][23][24], and data aggregation [25][26][27].

Recently, with the consistent increase in WSN application complexity, the way distributed applications are deployed in WSNs is another important component that affects the network lifetime. For instance, incorrect execution of data processing in some nodes or the transmission of big amounts of data with low entropy in some nodes could heavily deplete battery energy without any benefit. Indeed, application tasks are usually assigned statically to WSN nodes, which is an approach in contrast with the dynamic nature of future WSNs, where nodes frequently join and leave the network and applications change over the time [28]. This brings to issue talked in this thesis, which is defined as follows.

Dynamic deployment of distributed applications in WSNs: given the requirements of WSN applications, mostly in terms of execution time and data processing, the optimal allocation of tasks among the nodes should be identified so as to reach the application target and to satisfy the requirements while optimizing the network performance in terms of network lifetime. This issue should be continuously addressed to dynamically adapt the system to changes in terms of application requirements and network topology.

The rest of the thesis is structured as follows.

In Chapter 1, the state of the art related to WSN will be presented. First, some definitions and the principle regarding WSN related standards will be given. Then, an overview of the main WSN applications, from the dawn of the first military sensor networks to the nowadays IoT related applications, will be provided. Finally, the state of the art for the lifetime optimization problem will be analyzed.

In Chapter 2, the dynamics of a WSN will be investigated, in order to infer an energy consumption model and a network model that will be used throughout the rest of the thesis.

In Chapter 3, a framework for convenient allocation and deployment of the tasks of WSN applications will be proposed. This framework is based on a centralized optimization algorithm which first evaluates the application subdividing it into tasks; then, it assigns the tasks to the network nodes in order to accomplish the application target while minimizing its impact on the network lifetime. The performance of the algorithm will be evaluated by means of simulation results.

In Chapter 4, a distributed algorithm, the DLMA, based on gossip will be presented. DLMA is proven to improve the WSN capability to adapt to energy consumption changes, while contemporary reducing the message exchange overhead for the application deployment, and reducing the computational complexity making it scaling well with the network dimension.

Simulation results will be shown and analyzed to evaluate the performance of the algorithm.

In Chapter 5, the TAN distributed algorithm, based on non-cooperative game theory, will be described. This algorithm not only improves the previous algorithms performance reducing computational complexity, but also takes into account both network energy consumption and application execution time, so that the application deadline is not reached before its execution is done.

Finally, in Chapter 6, conclusions will be drawn regarding the effectiveness of the proposed algorithms, and some possible future works will be sketched.

Chapter 1

State of the Art

WSNs have recently gained worldwide attention. The phenomenal advances in Very Large Scale Integration (VLSI) and MEMS technology, contributed in the development of smaller and smarter sensors, and inexpensive low-power transceivers [1]. The impressive progress on research projects such as Smart Dust Project [29] and Wireless Integrated Network Sensors (WINS) Project [30] enabled the development of tiny, low-power and low-cost sensors, controllers and actuators. The miniaturization of computing and sensing technologies have considerably broaden the WSN application field.

In this Chapter, the state of the art related to WSNs will be presented. Section 1.1 gives an overview of the background and general principles concerning WSNs. The evolution of WSN from simple data acquisition systems to complex smart and interoperable systems placed in the wider context of the Future Internet (FI) and IoT will be surveyed in Section 1.2. In Section 1.3, the lifetime optimization problem will be analyzed and addressed considering different approaches: node deployment, routing techniques, data aggregation, and task assignment.

1.1 Background of WSNs

A *sensor* is a device which gathers information related to physical objects or processes (Table 1.1). This information, collected under the form of parameters or events, is transduced into signals that can be measured and analyzed. These signals are then transmitted to a *sink*, which is connected either to the Internet network or to controllers and processing station directly. An example of WSN is depicted in Figure 1.1.

WSNs can be classified either as hierarchical or flat. In a flat network, nodes (grey devices in Figure 1.1) act as sensors and *routers* contemporarily, and transfer data to the sink (blue devices in Figure 1.1) through multi-hop routing. In a hierarchical WSN, the network is organized into *clusters*, with cluster members and *cluster heads*. Cluster heads (green devices in

Table 1.1: Example of sensors [1]

Measure	Type
Temperature	Thermistors, thermocouples
Pressure	Pressure gauges, barometers, ionozation gauges
Optical	Photodiodes, phototransistors, infrared sensors, CCD
Acoustic	Piezoelectric resonators, micrphones
Mechanical	Strain gauges, tactile, capacitive diaphgrams, piezoresistive cells
Motion, vibration	Accelerometers, gyroscopes, photo
Flow	Anemometers, mass air flow
Position	GPS, ultrasound-based, infrared-based, inclinometers
Electromagnetic	Hall-effect sensors, magnetometers
Chemical	PH, electrochemical, infrared gas
Humidity	Capacitive and resistive, hygrometers, MEMS-based humidity
Radiation	Ionization detectors, Geiger-Mueller counters

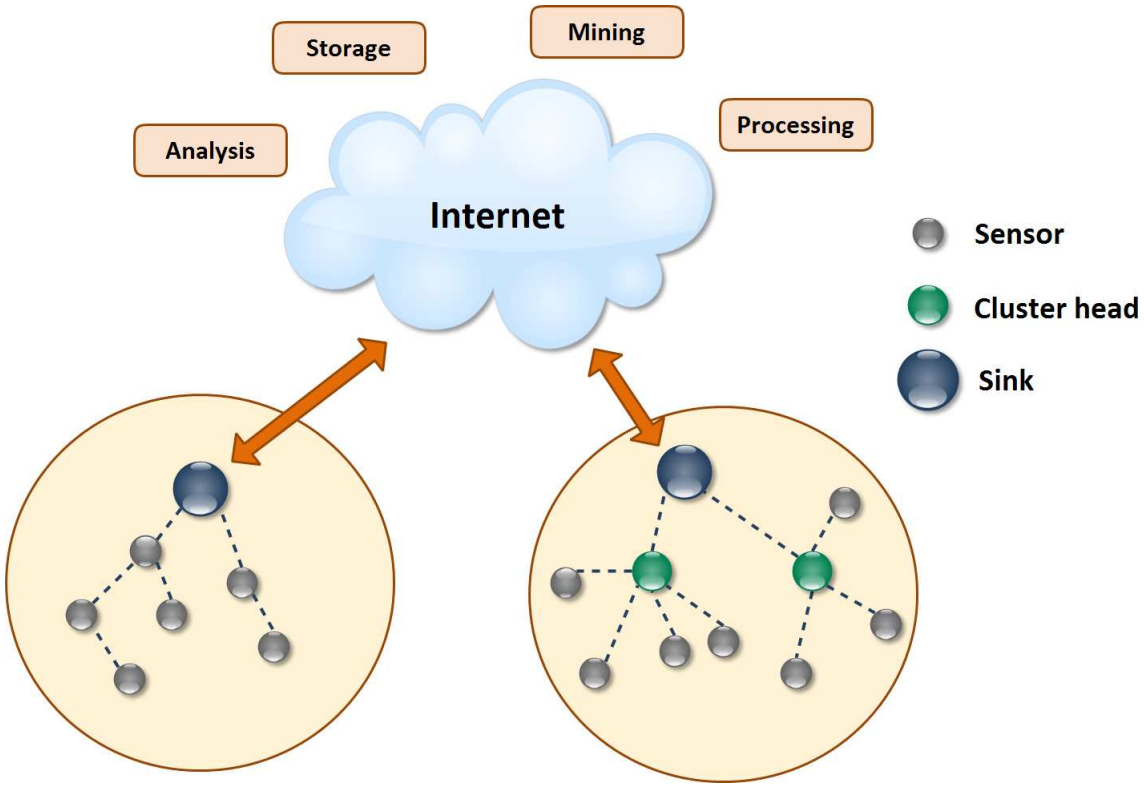


Figure 1.1: Wireless Sensor Network

Figure 1.1) are more powerful fixed or mobile relay nodes, which collect data from their cluster members and forward them to the sink.

Actuators are often included into WSNs to allow them to directly control the physical world. Such networks are usually referred to as *Wireless Sensor and Actuator Network* (WSANs). Information from the distributed sensor nodes is processed and transformed into commands to be sent to the actuator. The actuator then interact with the physical world, thereby forming a closed control loop. Some examples of WSANs are: the Heating, Ventilation and Air-Conditioning (HVAC) system [31], where actuators cooperate with sensors in order to keep a good indoor air quality; smart monitoring systems, which keeps energy flowing efficiently and economically through the system, maintaining power quality while reducing the risk of brownouts and blackouts [32]; smart city applications such as smart metering, traffic and parking management systems [33].

1.1.1 WSN Protocol Stack

According to [34], the protocol stack of a wireless sensor node (Figure 1.2) consists of five layers: a *physical layer*, a *data link layer*, a *network layer*, a *transport layer* and an *application layer*. The physical layer selects the required frequency, generates the carrier frequency, detects and modulates signals, and encrypts data. The data link layer is responsible for the multiplex of data, the detection of data frames, the medium access and the error control. Since the environment is typically noisy and nodes can be mobile, the Medium Access Control (MAC) protocol of the data link layer must be able to minimize collision with neighbors' broadcast. Furthermore, it must be power aware. The network layer takes care of routing the data supplied by the transport layer. The transport layer help to maintain the flow of data required by the application. Depending on the sensing tasks, different types of application software can be implemented and used on the application layer.

The stack also includes three planes which cross-cut all the layers: the *power management plane*, the *mobility management plane* and the *task management plane* [34]. These planes help the nodes coordinate their tasks and lower the overall power consumption. The power management plane is responsible for the node power management. If required, it may turn off unnecessary functionalities in order to preserve energy. The mobility management plane detects and registers the nodes movements in order for them to always maintain a route back to the sink, and to keep track of their neighbor nodes. The task management plane balances and schedules the tasks given to a specific node.

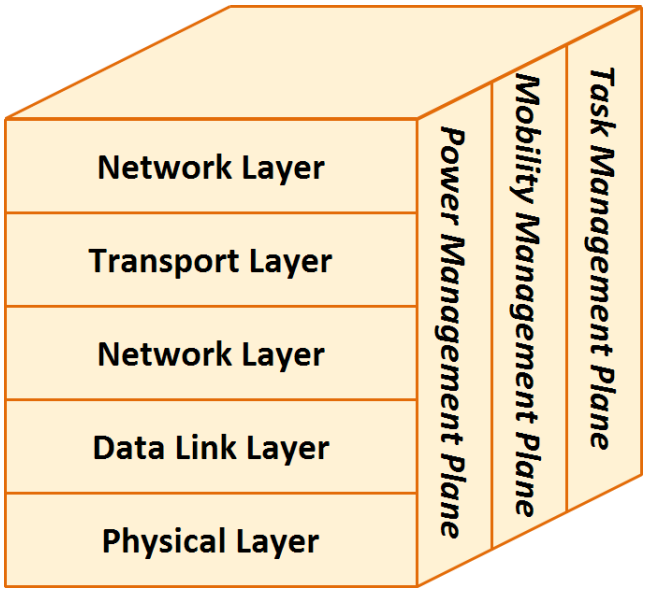


Figure 1.2: WSN protocol stack

1.1.2 Communication in WSNs

The earliest WSNs communicated using the IEEE 802.11 [35] family of standards. However, this standard is unsuitable for low-power sensor networks, due to consistent overheads. Furthermore, with the exception of high bandwidth applications (e.g. multimedia monitoring), typical data rate requirements are comparable to dial-up network bandwidth. Therefore, IEEE 802.11 data rates are usually much higher than necessary.

A whole range of WSN-related standards have been defined [36]. The most widespread are based on the IEEE 802.15.4 [37] and IEEE 802.15.3 [38] protocols.

IEEE 802.15.4

The IEEE 802.15.4 is the wireless standard that specifies the physical layer and the MAC layer for Low-Rate Wireless Personal Area Network (LR-WPAN). It is designed for short range wireless networks, where battery lifetime is critical. Its main strengths are: low power consumption, low complexity, and low deployment costs. The physical layer operates on the $868/915$ MHz and 2.4 GHz unlicensed frequency bands. The access to the radio channel is controlled by the MAC layer using the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol. ZigBee [39], WirelessHART [40][41], ISA100.11a [42], and 6LoWPAN [43][44] are the most common higher layer communication protocols based on the 802.15.4.

ZigBee It is the most commonly used communication protocol for WSNs. Thanks to its characteristics of robustness, reliability, simple deployment, and simple maintenance, ZigBee is

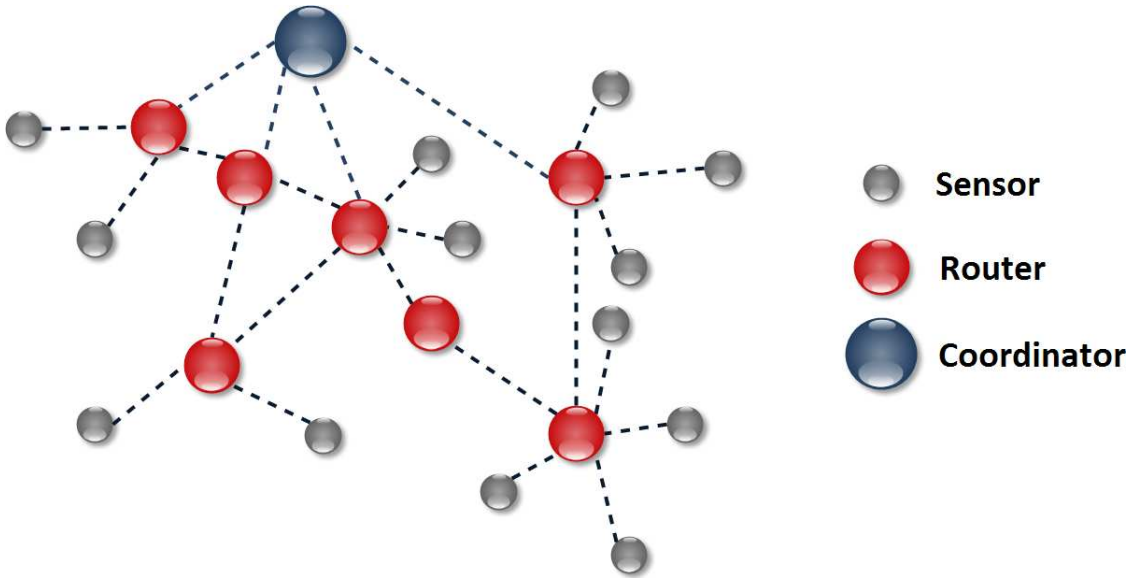


Figure 1.3: Example of a ZigBee network

designed for low-data rate, low-power consumption applications. ZigBee devices are often deployed as mesh networks that are able to organize themselves forming ad-hoc multihop networks. A ZigBee network consists of: a *coordinator*, *routers*, and *end devices*. Figure 1.3 shows an example of ZigBee network. The coordinators (blue devices in Figure 1.3) initiate the network, store information, and can bridge networks together. The routers (red devices in Figure 1.3) connect groups of nodes, and provide multi-hop communications across the network. The end devices (grey devices in Figure 1.3) can communicate only with the router or the coordinator. They can be sensors, actuators, or controllers.

WirelessHART It is a wireless communication standard mostly used in the industrial field for process measurement and control applications. It supports mesh networking, channel-hopping, and time-synchronized messaging. There are three types of WirelessHART devices: a *network manager*, *gateways*, and *wireless field devices*. The network manager is responsible for configuring the network, scheduling communication devices, managing message routes, and monitoring network health. The gateways enable communication between wireless field devices and host applications connected to a wired network. The wireless field devices are connected to process o plant equipment.

ISA100.11a It is a wireless communication protocol designed for low-data rate monitoring and process automation applications. The ISA100.11a only specifies tools for constructing an interface. It does not specify a process automation protocol application layer or an interface to an existing protocol. It focuses on scalability, reliability and interoperability with other wireless devices.

6LoWPAN It is the IPv6-based Low-power communication protocol for Wireless Personal Area Networks. The 6LoWPAN is designed for low-data rate devices that require Internet communication. In a 6LoWPAN WSN, devices communicate using IP-based protocol. Since IPv6 packet sizes are larger than the IEEE 802.15.4 frame size, an adaptation layer is provided by the standard. The IP packet header is compressed by the adaptation layer in order for the packet to fit into an IEEE 802.15.4 frame size. The 6LoWPAN is prospected to be widely used for Internet of Things applications.

IEEE 802.15.3

The IEEE 802.15.3 is the wireless standard that specifies the physical and MAC layers for High-Rate Wireless Personal Area Network (HR-WPAN). It is designed for real-time multimedia streaming applications. Its physical layer operates on the 2.4 GHz unlicensed frequency band, at data rates that go from 11 Mbps to 55 Mbps. Quality of Service (QoS) is ensured by the use of the Time Division Multiple Access (TDMA) channel-access scheme. It supports both synchronous and asynchronous data transfer. The most common higher-layer protocol based on IEEE 802.15.4 is Wibree [45].

WiBree It is a wireless communication standard designed for low-cost, low-power devices. WiBree allows the communication among small battery-powered devices and Bluetooth devices. WiBree limits are given by Bluetooth characteristics: it works at data rates up to 1 Mbps, for distances between devices up to 10 meters.

1.2 State of the Art in WSN Applications

The origin of WSNs can be traced back to the military research, when the term ”sensor networks” was not even coined. The first systems were designed ad-hoc, with specialized computers and communication capabilities [2].

During the Cold War, a large-scale acoustic system for underwater surveillance, the SOSUS, was deployed by the US Navy at strategic locations for a anti-submarine warfare purpose. The SOSUS is still currently used by the National Oceanographic and Atmospheric Administration (NOAA) to monitor blue whale sounds as well as seismic activity in the North Pacific Ocean [3]. Not only the ocean was monitored during Cold War, but also the sky: air defense radar systems were developed to defend the continental USA and Canada.

In 1978, the Defense Advanced Research Projects Agency (DARPA) started the Distributed Sensor Network (DSN) program, laying the foundation for modern WSN research.

By that time, the Advanced Research Projects Agency Network (ARPANET) counted about 200 hosts at universities and research institutes. The initial aim was to extend the ARPANET approach for communication to sensor networks. The DSN organized the Distributed Sensor Network workshop [46] to identify the technology components on which the research would focus. These included sensors, high-level communication protocols [47], processing techniques and algorithms, and distributed software. The first application of DSN consisted in a helicopter tracking system, developed at the Massachussets Insitute of Technology (MIT) [48]. Later, the Advanced Decision Systems developed a multiple-hypothesis tracking algorithm to deal with difficult situations involving high target density, missing detections, and false alarms, and decomposed the algorithm for distributed implementation [49].

The 1990s saw the evolution of military sensor networks, with the birth of systems for air, ground and water surveillance. The CEC [4], developed by the US Navy, consists of multiple radars which collect data on air targets. Anti-submarine sensor networks such as Fixed Distributed System (FDS) and Advanced Deployable System (ADS) [50] were deployed using acoustic sensor arrays. Ground monitoring was undertaken by means of UGS such as the REMBASS [5] and the Tactical REmote Sensor System (TRSS).

Modern research in the WSN field started exactly in the 1990s, with the advent of smarter and smaller devices. Before that time, sensor nodes were rather large, and communications among them were wired. In 1996 the University of California at Los Angeles (UCLA), in collaboration with the Rockwell Science Center, developed the Low Power Wireless Integrated Microsensor (LWIM) [51], funded by DARPA. It was an intelligent wireless low-power system, whose nodes were required to: 1) be reconfigurable by their base station, 2) be autonomous to permit local control of operation and power management, 3) self-monitor themselves for reliability, 4) be power efficient for long term operation, 5) incorporate diverse sensor capability with highly capable, low power microelectronics. In 1998 DARPA selected for funding the Smart Dust project [29], conducted by the University of California at Berkley. The aim of the project was to demonstrate that a complete sensor/communication system can be integrated into a cubic millimeter package called *mote*. Berkely, and more precisely the Berkely Wireless Research Center (BWRC) was responsible for another important project: the PicoRadio project [52]. PicoRadio focused on the design of what they called *PicoNodes*, small, light and low-cost nodes characterized by ultra-low power consumption. The goal was to reach a power-dissipation level so low that PicoNodes could be able to power themselves using energy extracted from the environment, such as solar or vibrational energy. The MIT conducted the Micro-Adaptive Multi-domain Power-aware Sensors (μAMPS) project [53], which studied the design of a low-power wireless sensor device that is able to scale voltage dynamically, and the techniques to restructure data processing algorithms to reduce power requirements at the soft-

ware level. In 2003 the IEEE released the first version of the IEEE 802.15.4 standard, which specifies the physical and MAC layers for LR-WPAN (see Section 1.1.2).

All these efforts have resulted in the evolution of sensors into tiny devices combining sensing with communication, computation and storage. In the following, some examples of the most common monitoring applications will be given.

1.2.1 Environment and Habitat Monitoring

Environment and habitat monitoring lend themselves well to sensor networks, because the variables under monitoring (e.g. temperature, light exposure, pollution) are distributed over a large region. There is a huge amount of environment and habitat monitoring applications involving WSN. One example is the Environmental Observation and Forecasting System (EOFS) [6]. This type of large-scale distributed embedded system is designed to monitor, model, and forecast wide-area physical processes such as river systems. A prototype of EOFS, CORIE, was deployed to study the Columbia river estuary and plume. Another example is [7], where the requirements of a sensor network for volcanic data collection are presented. The application focuses on the detection of triggered event and retrieval of reliable data, taking into account bandwidth and data-quality demands. The WSN was deployed on Volcán Tungurahua and Volcán Reventador in Ecuador. A low-cost WSN based system for monitoring the alpine environment is SensorScope [8]. SensorScope was deployed on top of a rock glacier in Switzerland. The aim was to monitor a micro-climate phenomenon leading to cold air release from a rock-covered glacier in a region of high alpine risks. An example of air pollution monitoring application is the Wireless Sensor Network Air Pollution Monitoring System (WAMPS) [9], a large scale WSN studied to monitor air pollution in Mauritius.

1.2.2 Infrastructure Security and Terror Threat Alerts

Infrastructure monitoring concerns infrastructure security and counterterrorism applications. Critical buildings and facilities such as power plants and communication centers need to be protected from structural failures, as well as terrorist threats [2]. WSNs can provide early detection of these risks. An example of a structural health monitoring application is given in [11]. In this study, an indirect damage detection approach which monitors the changes in structural properties is considered. In particular, they focused on vibration monitoring, by means of accelerometers. This system was deployed on the Berkley pedestrian footbridge and on the Golden Gate Bridge in San Francisco. As far as terrorist threats are concerned, WSNs are particularly suitable to hostile domains, and therefore they have been widely used in wireless UGS systems contexts. An adaptive autonomous UGS application is represented by the Sense, Decide,

Act, Communicate (SDAC) [10]. In this system, sensors not only sense the environment and communicate the information gathered, but they also make decisions and act upon them.

1.2.3 Industrial Sensing for Machine Health Monitoring

Commercial industry has long been interested in sensing as a means of lowering cost and improving machine performance and maintainability. Unplanned downtime events can be considerably reduced by monitoring machine health through the use of sensors nodes that can be deeply embedded into machines, and can reach regions inaccessible by humans. The collaborative nature of industrial WSNs brings several advantages over traditional wired industrial monitoring and control systems, including self-organization, rapid deployment, flexibility, and inherent intelligent-processing capability. Industrial WSNs are able to create a highly-reliable and self-healing industrial system that rapidly responds to real-time events with appropriate actions [12]. In [13] Predictive Maintenance technologies, and more precisely vibration analysis, are used to detect impending failures in advance. The system developed was deployed in a central utility support building at a semiconductor fabrication plant, and aboard an oil tanker operating in the North Sea.

1.2.4 Traffic Control and Intelligent Transportation System

As traffic congestion has become a critical issue, traffic control applications based on WSNs have recently mushroomed, so as to make WSNs a crucial technology within the *Intelligent Transport System* (ITS) paradigm. These distributed sensing systems gather information about the position, density, sizes and speed of vehicles on roads or parking lots, and process it in order to give suggestions to drivers such as alternative roads, or the nearest vacant car parking space. An example of traffic surveillance system is given by Traffic-Dot [14]. In Traffic-Dot sensors gather traffic condition data that are sent to a traffic management center, which analyzes them and then take actions such as adjusting the traffic light duration. A highway Cooperative Collision Avoidance (CCA) in the context of vehicle-to-vehicle wireless communication is described in [15]: as soon as a vehicle detects a collision, the information is sent to the vehicles behind it, so that successive collisions can be avoided. In [16], an example of intelligent car park management system based on WSNs is described. This system is able to find vacant parking lots, auto-toll, manage security, and report statistics. A system architecture enabling mobile nodes to query a largely deployed WSN in an ITS scenario is defined in [17].

1.2.5 The Future Internet and the Internet of Things

According to [54], WSANs are among those technologies recognized as the atomic components that will link the real world with the digital world. Sensor networks will play a crucial role in the IoT [18], given their inclination to cooperate with other technologies such as Radio-Frequency IDentification (RFID) and Near Field Communications (NFC), and the standardization of communication protocols that can give them the possibility to perfectly integrate within a big heterogeneous network of things (e.g. using the 6LoWPAN protocol described in Section 1.1.2). Sensor network is a key technology for *Ubiquitous Sensor Network* (USN), where heterogeneous and geographically dispersed WSANs are integrated into rich information infrastructures for accurate representation and access to different dynamic user's physical contexts [55]. Small physical dimensions of sensor nodes are often one of the musts in a typical USN, since it is dedicated to unobtrusive integration into living, working, scientific, industrial, and other environments [56]. This implies high integration of sensing, computing, and communication capabilities of the devices meeting at the same time application-specific demands. Using mostly wireless infrastructure, ubiquitous technologies are supposed to interface the physical environment in various scenarios such as Smart Cities [57], Smart Homes [58][59][60] and Smart Grids [61].

1.3 Lifetime Optimization Problem in WSNs

One of the main challenges for WSNs is the extension of the network lifetime. As already seen in the previous Sections, huge progresses have been made to improve sensor node hardware. Furthermore, a great deal of effort has been made by researchers to find effective strategies to increase network lifetime. These strategies encompass network node deployment, routing mechanisms, data aggregation, and optimal task assignment. In the following, the state of the art regarding these mechanisms will be presented.

1.3.1 Node Deployment

An appropriate node deployment is probably the most critical issue to be addressed to reduce communication costs within a WSN. In [19], nodes are non-uniform spaced as a function of their distance. Since nodes close to the sink feel the effects of their higher traffic more than other nodes, spacing are adjusted in such a way that nodes with higher traffic have a shorter hop distance than nodes with lower traffic. An algorithm for contemporarily improve coverage and lifetime is presented in [20]. In this work, the authors model the coverage and lifetime of a node as a Gaussian random variable, whose parameters depend on some network settings. The

nodes are then deployed according to the selected policies, which could be either lifetime- or coverage-oriented. A step forward is taken in [21], where an algorithm to maximize the area of coverage, minimize the network energy consumption, maximize the network lifetime and minimize the number of deployed nodes, while assuring connectivity of each node to the sink is proposed. In this study, the problem is modeled as a multi-objective optimization problem where the aim is to find the node deployment corresponding to the optimal trade-off.

1.3.2 Routing

Once the network is deployed, the use of appropriate routing mechanisms could help to considerably increase its lifetime: a convenient choice of paths to route data may result in significant energy conservation. In [22], five power-aware metrics for determining energy-efficient routes are presented. In these metrics, routes are chosen taking into account the energy consumption at each node. Nevertheless, since node residual energy is not considered, nodes belonging to the minimum energy path will be drain-out of batteries quickly. This issue is addressed in [23], where algorithms of maximization of the system lifetime are proposed. MobiRoute [24] introduces the concept of mobile sinks, which are mobile nodes with significantly more resources than normal nodes. Whenever is needed, mobile sinks take charge of forwarding data instead of normal nodes. Other energy-efficient routing techniques are described in [62][63][64].

1.3.3 Data Aggregation

Besides these traditional techniques that either modify network topology or provide an energy-efficient routing protocol, due to the recent advances in MEMS, low power transceivers and microprocessor dimensions, which provide additional capabilities to sensor nodes, more advanced methods that extend network lifetime are now possible. Therefore, network lifetime optimization not only is centered on reduction of packet transmission power, but also involves convenient data processing that reduces the amount of data delivered to data sinks. This is the principle behind node clustering protocols, such as LEACH [25], EC [26] and the clustering algorithms in [27], in which cluster head nodes aggregate data and reduce transmitted data volume, which in turn reduces the overall transmission energy consumption of the network.

1.3.4 Optimal Task Assignment

A step forward in extending network lifetime is to consider not only data aggregation to reduce data volume, but also any possible data processing task assignment. Task assignment is performed taking into account various aspects related to energy consumption such as network

topology, battery power, and node processing capabilities. However, existing methods have limited scope in studying lifetime extension with regards to application data processing. For instance, in [65], maximization of cluster lifetimes is studied. However, this approach considers only communication tasks, but not the tasks generated by applications and assigned to the network for execution. Furthermore, it only focuses on homogeneous networks, which are not common in real scenarios. In contrast, [66] considers execution of application tasks, and provides an adaptive task allocation algorithm that aims at reducing the overall energy consumption by balancing node energy levels overall the network. However, this mechanism requires exchange of some additional messages among all the nodes in the network, which considerably increases packet overhead.

One method to perform task assignment is the use of a central controller that divides large application programs into smaller and easily executable tasks and then distributes these tasks to nodes. Task allocation solutions that consider a central controller are called centralized solutions. A centralized solution for the maximization of the WSN lifetime will be described in Chapter 3, and is based on the study proposed in [67]. Some other centralized lifetime maximization algorithms are studied in [68][69][70]. The problem with centralized algorithms is that they suffer from computational complexity, as well as large control packet overhead due to frequent updates collected from nodes in order to adapt to network dynamism.

To address this problem, the DLMA [71], will be presented in Chapter 4. DLMA is an overlaying framework that determines the distribution of tasks among the nodes in a WSN by means of a distributed optimization algorithm, based on a gossip communication scheme, aimed at maximizing the network lifetime. A similar approach is studied in [72], where the distributed algorithm is based on particle swarm optimization. However, the major drawback of these studies is that they do not take into account the deadline of the applications assigned to the network.

Chapter 2

Modeling of WSNs

To model an energy-efficient task assignment algorithm, the perfect knowledge of all the parameters and variables that intervene in the network dynamics is required. In order to do it, a model that characterizes the network nodes, the tasks in which the application assigned to the WSN is subdivided, and the energy consumption dynamics is needed. Therefore, prior to proceeding with the description of the proposed algorithms, a careful evaluation of the characteristics parameters that influence the network behavior is necessary. In this Chapter, the models that will be used throughout the thesis are presented. The energy consumption model will be analyzed in Section 2.1. A model for WSN topology and node devices is provided in Section 2.2.

2.1 Energy Consumption Model

Energy consumption in WSNs is determined by three main components: sensing, processing and transmission.

Sensing energy consumption e_i^{sens} for sensor node i is determined by the specific characteristics of the sensor. Its value is determined on the basis of the device datasheet.

Processing energy consumption e_{ih}^{proc} for sensor node i and task h is proportional to the complexity of task h – i.e. the number of instructions I_h needed to complete it – and to the average energy consumption per instruction e_i^{ins} related to node i. Hence

$$e_{ih}^{proc} = I_h \times e_i^{ins} \tag{2.1}$$

As far as communication energy consumption is concerned, there are two main components that contribute to it: transmission and reception energy consumption. As mentioned in [73]

$$\begin{cases} P_{ij}^T = P_i^{T0} + P_i^A(\delta_{ij}) = P_i^{T0} + P_i^{Tx}(\delta_{ij})/\eta_i \\ P_j^R = P_j^{R0} \end{cases} \tag{2.2}$$

where: P_{ij}^T and P_j^R are radio frequency power consumption values for transmitting and receiving

respectively; $P_i^A(\delta_{ij})$ is the power consumption of the Power Amplifier (PA), which depends on the distance δ_{ij} between transmitting node i and receiving node j; P_i^{T0} and P_j^{R0} are the components of power consumption of the transmitting and receiving circuitry respectively; P_i^{Tx} is the output power at node i antenna which, for reliable transmissions, depends on the distance δ_{ij}; η_i is the drain efficiency of the PA at node i.

Considering a channel in which the path loss component is predominant, and thus secondary effects such as multipath and Doppler can be neglected, the transmitted power $P_i^{Tx}(\delta_{ij})$ can be expressed as

$$P_i^{Tx}(\delta_{ij}) = P_j^{Rx} \times A_{ij} \times \delta_{ij}^{\alpha_{PL}} \tag{2.3}$$

where A_{ij} is a parameter determined by the characteristics of the antennas (such as gain and efficiency) and α_{PL} denotes the path-loss exponent, which is about 2 for free space. This kind of modeling is typical of free space propagation. Of course, the model might be extended to account for other fading effects.

From (2.2) and (2.3) follows that

$$P_{ij}^T = P_i^{T0} + \frac{P_j^{Rx} \times A_{ij} \times \delta_{ij}^{\alpha_{PL}}}{\eta_i}$$

Considering $\varphi_{ij} = P_j^{Rx_{min}} \times A_{ij}$, where $P_j^{Rx_{min}}$ is the minimum reception power at node j for a reliable communication

$$P_{ij}^T = P_i^{T0} + \frac{\varphi_{ij} \times \delta_{ij}^{\alpha_{PL}}}{\eta_i}$$

Defining as e_{ij}^{tx} the energy per bit necessary to transmit data at rate R from node i to its adjacent node j, and e_j^{rx} the per-bit energy consumed to receive data at node j

$$\begin{aligned} e_{ij}^{tx} &= \frac{P_{ij}^T}{R} = \frac{1}{R}\left(P_i^{T0} + \frac{\varphi_{ij} \times \delta_{ij}^{\alpha_{PL}}}{\eta_i}\right) \\ e_j^{rx} &= \frac{P_j^R}{R} = \frac{P_j^{R0}}{R} \end{aligned} \tag{2.4}$$

The model described does not take into account mechanisms such as sleep schedule and route discovery, which may produce overhead. Therefore, it could be necessary to consider not just the single packet transmission, but also the energy consumption due to the overhead.

2.2 Network Model

The goal of the WSN under consideration is to accomplish a given application tasks, mostly based on some measurements performed on the relevant environment. In this scenario, each

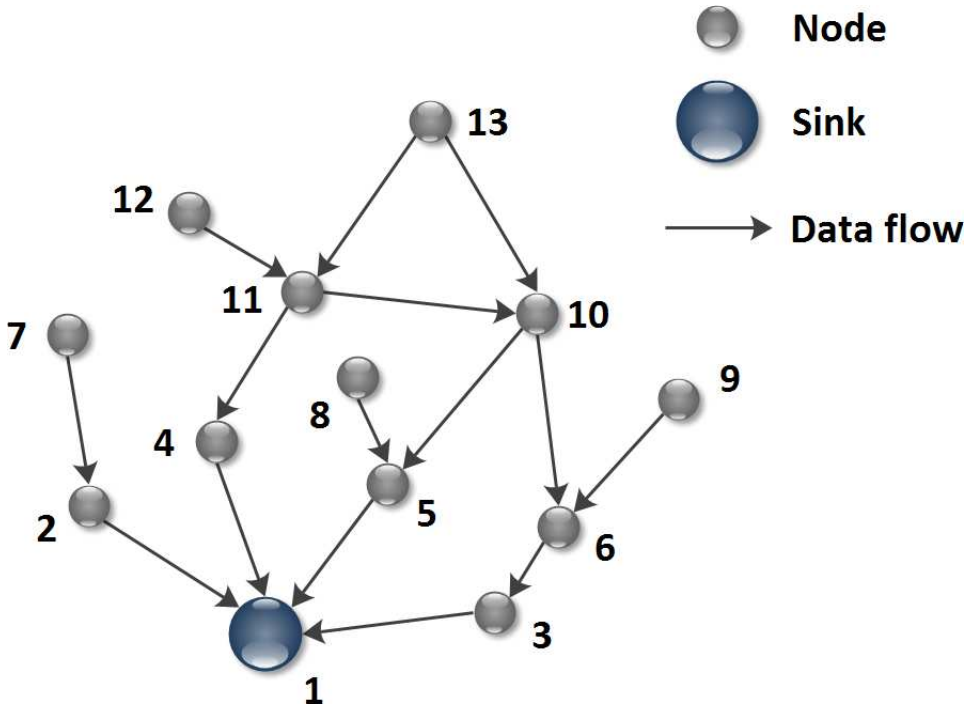


Figure 2.1: Example of network model

node can perform a given set of tasks, such as: data processing, temperature measurement, video monitoring, data transmission, and data storage.

The WSN is modeled as a Directed Acyclic Graph (DAG) $\mathcal{G}_X = (X, E_X)$, where the vertices represent the nodes $X = \{1, \ldots, i, \ldots, N\}$, while the links are described by the set of edges $E_X = (e_{ij}^X)$, where each edge represents a connection from node i to node j. Node i can be a sensing node, a router or an actuator (or a node with a combination of this roles). The network is considered to have only one sink, referred to as node 1. Figure 2.1 shows an example of network topology, where an ID is assigned to each node.

The network is characterized by two parameters: the distance matrix $\mathbf{\Delta} = (\delta_{ij})$ and the matrix $\mathbf{\Phi} = (\varphi_{ij})$, described in Table 2.1. Furthermore, each node in the network is represented by the parameters presented in Table 2.2, necessary to compute energy consumption values.

Table 2.1: Network parameters

Parameter	**Description**
$\mathbf{\Delta} = (\delta_{ij})$	Matrix of the pairwise distances (in meters) between adjacent nodes. If nodes i and j are not adjacent, then $\delta_{ij} = \infty$
$\mathbf{\Phi} = (\varphi_{ij})$	Matrix of parameters φ_{ij} introduced in Section 2.1. If nodes i and j are not adjacent, then $\varphi_{ij} = \infty$

Table 2.2: Node parameters

Parameter	**Description**
e_i^{sens}	Average energy spent by node i to perform a sensing task
e_i^{ins}	Average energy spent by node i to perform a single instruction, as defined in Section 2.1
$\boldsymbol{e}_i^{tx} = (e_{ij}^{tx})$	Each element $e_i^{tx}j$ of this vector is the energy spent per bit to send data from node i to an adjacent node $j \in X : e_{ij}^X \in E_X$, as described in Section 2.1
e_i^{rx}	Per-bit reception energy at node i
e_i^{res}	Residual energy of node i

Chapter 3

Centralized Task Allocation Algorithm

In the previous Chapters, the evolution of WSNs from simple monitoring networks to more complex low-cost systems able to dynamically interact with the surrounding environment has been discussed. Still, extending the network lifetime is an open challenge. These considerations contribute to the vision of an horizontal ambient intelligence infrastructure wherein sensing, computing and communicating infrastructure is set with a programmable middleware that allows for quickly deploying different applications running on top of it so as to follow the changing ambient needs, e.g. monitoring a given geographical area and alerting when something is happening herein; activating the heating system when the ambient is getting cold; tracking the processing chain in industrial plants to prevent hazardous scenarios. In this case, the focus is put on the need of a logic that, starting from the desired application, can be set up in complex scenarios with hundreds of nodes, evaluate every possible decomposition of the application into sensing and processing tasks, and decide which nodes should perform the required tasks so as to accomplish the application target while minimizing its impact on the network lifetime [67].

This Chapter is organized as follows. Section 3.1 describes the problem and how it has been approached. Section 3 defines the centralized algorithm used to extend the network lifetime. In Section 3.3 the algorithm performance are evaluated by means of simulation results.

3.1 Problem Formulation

The goal of a WSN is to accomplish a given number of tasks mostly based on some measurements performed on the relevant environment. In the proposed scenario, not all the nodes have the same capabilities. In Figure 3.1, three sets of possible tasks have been considered (e.g. data processing, temperature measurement and video monitoring). Given the status of the network in terms of node capacities, topology, and energy distribution, the problem addressed is to assign to each node the tasks that, combined together, contribute to the target network application

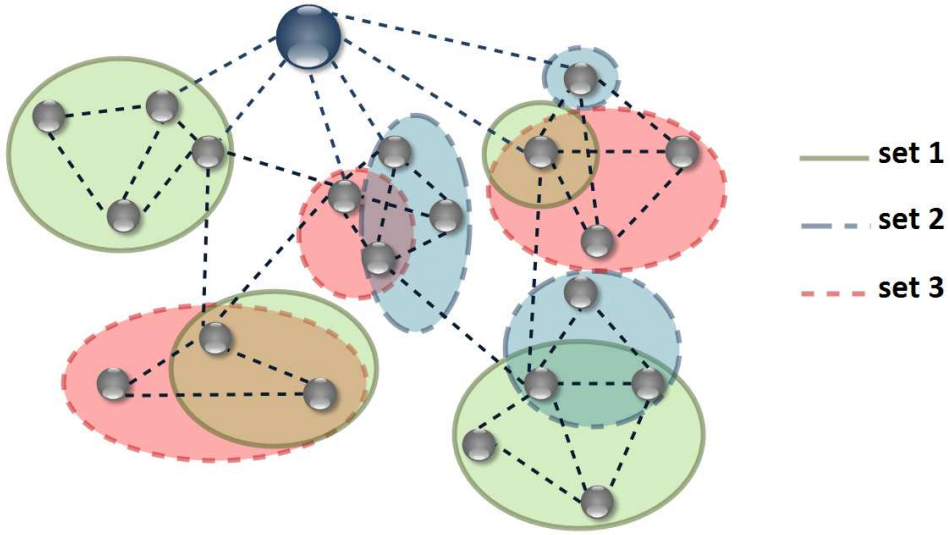


Figure 3.1: Example of a WSN. Nodes belonging to set 1 are able to perform task t_1; nodes in set 2 are able to perform task t_2; nodes in set 3 are able to perform task t_3

while minimizing its impact on the residual lifetime.

The network model used in this framework has been defined in Section 2.2.

The application assigned to the network can be decomposed into a sequence of distributed tasks. This application could represent diverse operation, such as: computing the average of the temperature in a given geographical area, measuring the light intensity in a room, video-surveillance of a specific geographical area, or a combination of these. In the following, a specific reference application, named *spatial and temporal monitoring*, is considered as an example to better explain the model. In this application, a spatial and temporal mean operation over an hour is performed on the temperature values sensed every 10 minutes by the sensors from 3 different locations; the average values are stored in the sink.

Three significant parameters can be associated to the application to be deployed:

- the total cost value P^{tot}, which takes into account the overall power consumption related to the application under consideration;
- the DG that describes the relations among tasks $\mathcal{G}_T = \{T, E_T\}$, where $T = \{t_1, \ldots, t_l, \ldots, t_L\}$ is the sequence of tasks in which the application can be subdivided, while $E_T = (e_{uv}^T)$ is the set of edges e_{uv} representing a unidirectional data transfer from task u to task v. With reference to the *spatial and temporal monitoring* example, t_1, t_2 and t_3 are the sensing tasks: t_1 is *temperature sensing in area* 1, t_2 is *temperature sensing in area* 2, and t_3 is temperature sensing in area 3. The processing tasks are: the *temporal mean* t_4, and the *spatial mean* t_5. The DG for this example is shown in Figure 3.2, while the task descriptions are listed in Table 3.1.

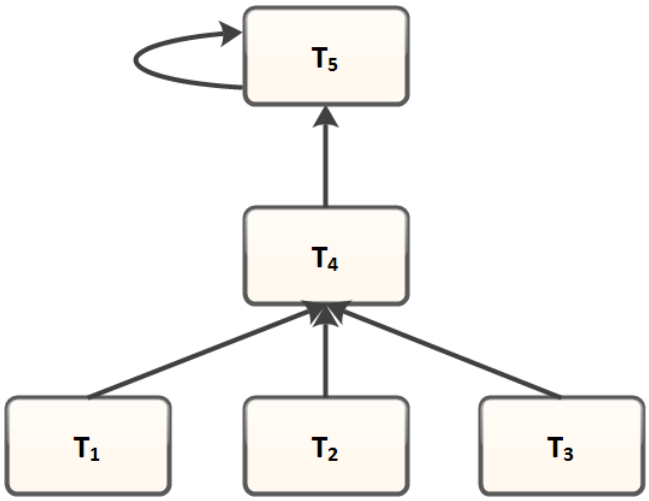


Figure 3.2: DG corresponding to the *spatial and temporal monitoring* example

Table 3.1: Tasks for the *spatial and temporal monitoring* example

$\mathbf{t}_i$	**Description**
t_1	Temperature sensing in area 1
t_2	Temperature sensing in area 2
t_3	Temperature sensing in area 3
t_4	Temporal mean
t_5	Spatial mean

Furthermore, two parameters are associated to each node with reference to the tasks to be performed:

- the set $D_i = \{d_{i1}, ..., d_{im}, ..., d_{il_i}\}$, where the elements of D_i are the tasks that the node i is able to perform;
- the status s_i, that defines which task t_l is assigned to node i. The status s_i can only be chosen among the set of tasks D_i that the node is able to perform. If node i is not assigned to any task, $s_i = 0$. In this case, it only forwards received data, if any.

Thanks to the greater processing power and storage capacity of modern sensors, contrary to the past, the same application can be performed in several different ways: gathered data can be immediately sent to a sink or it can be processed before being transmitted. In the case of the latter, the number of bits to be sent would be smaller, and therefore the transmission energy consumption would be lower as well; however, processing energy consumption could be higher in this second case. Quantifying the energy consumption in both cases, it could be possible to establish which one determines a reduction of battery consumption in the sensors, incrementing the network lifetime.

The framework described further takes a high-level code as input, evaluates which combination of the set of statuses $\boldsymbol{S} = \{s_1, ..., s_i, ..., s_N\}$ permits the application to be performed with the lowest possible cost function P^{tot}, and finally elaborates and assigns among the nodes the most appropriate tasks to be performed. Hence, it is evident that the cost function P^{tot} will vary depending on the status of each node, that is, how the tasks are assigned to network nodes. The problem addressed is then defined as the set of statuses $\boldsymbol{S}$ that minimizes the impact of the application on the network lifetime. In the following, the considered scenario will be elaborated by defining further constraints that solve the problem.

3.2 Deployment of Distributed Applications

In the following, the proposed solution towards a distributed application deployment in WSN is presented. The following Subsections present: the constraints on the traffic generated by the distributed application; the concept of virtual nodes, which are duplicates of real nodes that are introduced to deal with nodes that perform more than a single task; the cost functions built on the basis of the energy consumption formulas; the network lifetime maximization procedure; a summary of the proposed framework. Note that the modeling proposed in this work and presented in this Section is aimed at evaluating all the possible solutions of application deployment in terms of data transmission and processing. The parameters, constraints and cost functions are introduced for the sole aim of evaluating the viability of the solution.

3.2.1 Constraints on Traffic Flows

In the reference scenario it is assumed that the sources of traffic in the network (the sensors) generate samples of k bits at a certain frequency f. The processing in the network is performed on this type of traffic flow coming from different nodes. The generic node i receives the traffic $\boldsymbol{\Theta}_i^{in}$ over which it performs the task corresponding to its assigned status s_i. The effect of this task is the generation of the output traffic $\boldsymbol{\Theta}_i^{out}$, which is computed by function p as follows

$$\boldsymbol{\Theta}_i^{out} = p(\boldsymbol{\Theta}_i^{in}, s_i) \tag{3.1}$$

The output traffic is then sent to the next node towards the sink.

The data generated by p in node i is modeled by the H-dimensional vector $\boldsymbol{\Theta}_i^{out} = (\theta_{i1}^{out}, ..., \theta_{ih}^{out}, ..., \theta_{iH}^{out})$, with element $\theta_{ih}^{out} = \{k_{ih}^{out}, f_{ih}^{out}\}$ corresponding to a traffic flow where each sample of k_{ih}^{out} bits is transmitted at the frequency f_{ih}^{out}. Each sample described by θ_{ih}^{out} results from a spatial processing or a sensing. The data $\boldsymbol{\Theta}_i^{out}$ is then sent to the following node j, according to adjacency matrix $\boldsymbol{E}_X$.

Node j receives data from all the adjacent nodes that reach the sink through j

$$\mathbf{\Theta}_j^{in} = \bigcup_{i=1}^{N} \mathbf{\Theta}_i^{out} \times e_{ij}^{X} \tag{3.2}$$

This implies that, if node j's adjacent nodes do not have any output flows, node j is not transmitting any data, i.e. $\mathbf{\Theta}_j^{out}$ has all its fields set to 0.

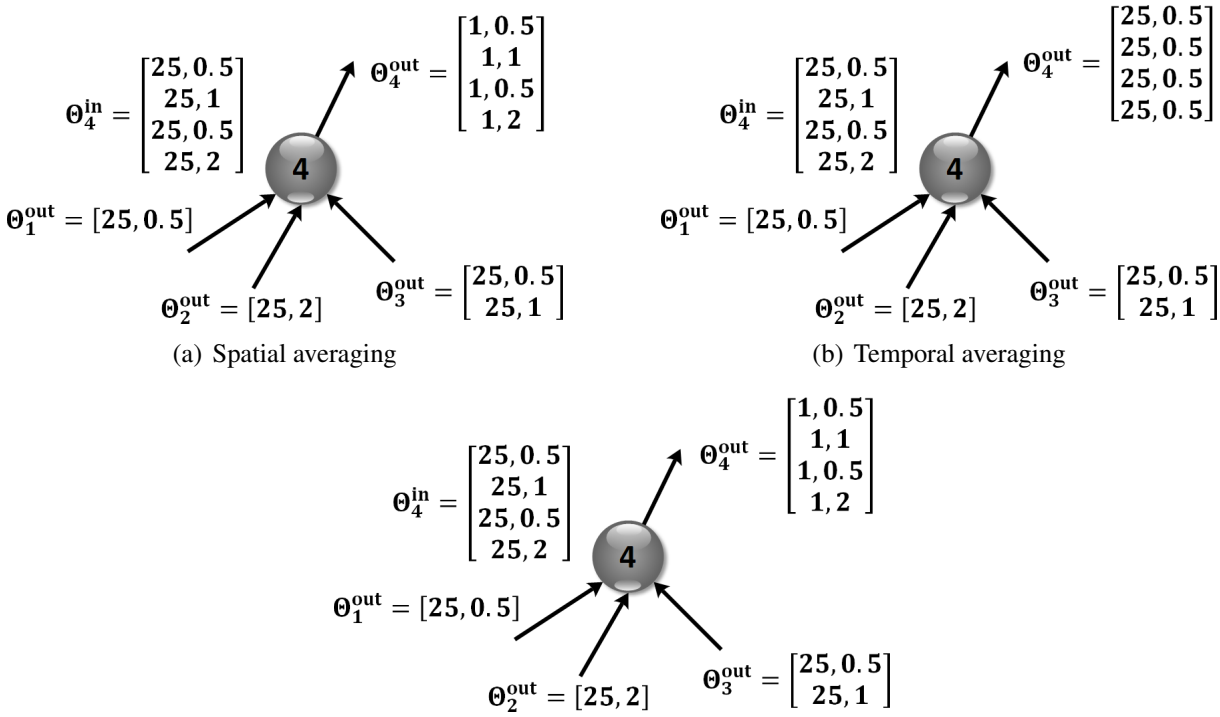
As defined by Equation 3.1, data $\mathbf{\Theta}_j^{in}$ received by node j is processed according to its status:

- if s_j is a sensing status, p does not take any $\mathbf{\Theta}_j^{in}$ as input and the output is defined by the specific sensing task;
- if $s_j = 0$, the output of p is exactly equal to $\mathbf{\Theta}_j^{in}$;
- if s_j is a processing status, $\mathbf{\Theta}_j^{out}$ can be the most diverse depending on the specific processing objectives, which are coded in s_j and that control the specific function p. In the following certain cases will be analyzed.

Referring to the *spatial and temporal monitoring* example, processing can be a spatial averaging, a temporal averaging, or a combination of both. In a spatial processing, the samples coming from different paths are processed together, as shown in an example in Figure 3.3(a). Here, four flows of 25 bits per second are received by node 4, and are then averaged to produce a single flow of 25 bits per second. Accordingly, the resulting $\mathbf{\Theta}_j^{out}$ is made of only one element $\theta_{j1}^{out} = \left\{k_{j1}^{out}, f_{j1}^{out}\right\}$, where the number of bits per sample k_{j1}^{out} and the frequency f_{j1}^{out} are equal to those of each input flow. Note that, in general, k_{j1}^{out} is not necessarily equal to the number of bits of each input flow, but it may be different according to the processing output.

Differently, the temporal averaging is performed on every traffic flow in $\mathbf{\Theta}_j^{in}$. The resulting $\mathbf{\Theta}_j^{out}$ contains the same number of traffic flows as in $\mathbf{\Theta}_j^{in}$, where each element θ_{jh}^{out} is characterized by the same number of bits per sample k_{jh}^{out} and the same frequency f_{jh}^{out} corresponding to the averaging frequency associated to the node status s_j. Indeed, different status codes may be associated to the temporal averaging, each one distinguished by a different processing frequency. Figure 3.3(b) shows an example for this kind of processing, where, in this case, node status is assumed to correspond to temporal averaging with frequency 0.5 Hz.

Other processing tasks can be performed on every single sample of each received traffic flow without involving other samples. This is the case, for instance, where one must evaluate whether the received values exceed a given threshold or not, consequently transmitting a boolean output value. The only thing that changes in the output traffic flows is the number of bits per sample; therefore, $\mathbf{\Theta}_j^{out}$ contains the same number of traffic flows as in $\mathbf{\Theta}_j^{in}$ at the same



(a) Spatial averaging

(b) Temporal averaging

(c) Single sample processing

Figure 3.3: Examples of data processing in node 4 that receives input traffic from nodes 1-3. The sketches correspond to the input and output traffic for: spatial averaging (a), temporal averaging (b), and single sample processing (c)

frequency f_{jh}^{out}, but with different bits per sample k_{jh}^{out}. Figure 3.3(c) shows the traffic flows for the described processing.

There are many other processing tasks that can be performed in a given network. For each one of these, an operator $p(x, y)$ is defined. Note that for the required objective, this operator is needed to figure out the traffic flows that will be traversing the network for each deployment scenario.

3.2.2 Virtual Nodes

It is possible that a single node has to perform more than one task. For instance, referring to the *spatial and temporal monitoring* scenario, it may happen that a single node has to compute both spatial and temporal average values on the received data. To take into account this type of scenario, the concept of virtual nodes is introduced. These are copies of real nodes, each one able to perform only a specific task and sending the resulting data at zero-energy cost to the next virtual node (except the last one that sends the data to the next node). The set of tasks that a single node i can perform consecutively for the implementation of the assigned application is

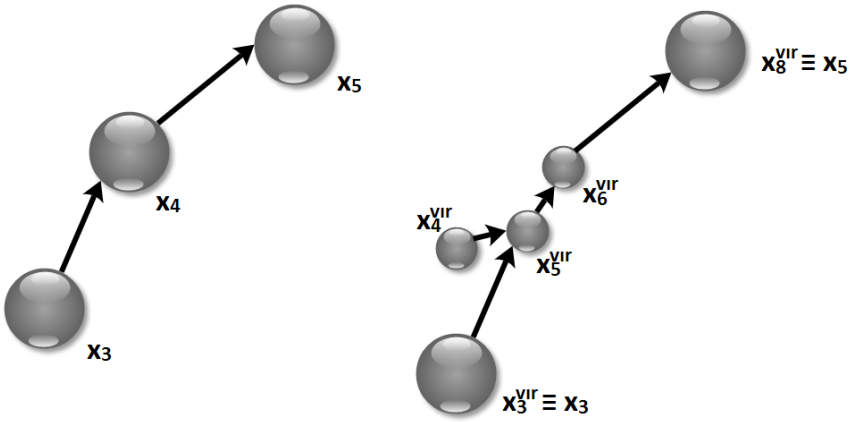


Figure 3.4: Example of virtual nodes substituting node x_4, which can perform a sensing action

defined as

$$G_i = D_i \cap T \tag{3.3}$$

If $|G_i| > 1$, node i is divided into $|G_i|$ virtual nodes. Each virtual node is created so as to be able to execute only one of the tasks in G_i. Each virtual node can then be assigned to perform only one task. Hence, the new network will have a number N^{vir} of total nodes $X^{vir} = (x_v^{vir})$

$$N^{vir} = \sum_{i=1}^{N} \Gamma_i, \qquad \text{with } \Gamma_i = \begin{cases} |G_i| & \text{if } |G_i| > 0 \\ 1 & \text{otherwise} \end{cases}$$

The set of possible tasks for each node x_v^{vir} is D_v^{vir}. Figure 3.4 draws an example of sequence of virtual nodes for node x_4, which is substituted by nodes x_4^{vir}, x_5^{vir} and x_6^{vir}. With reference to the *spatial and temporal monitoring* example and the associated DG of tasks $\mathcal{G}_T$ shown in Figure 3.2. Additionally, let $\{t_1, t_4, t_5\} \subset D_4$, i.e. node x_4 can monitor area 1, perform a temporal averaging, and a spatial averaging

$$\begin{aligned} D_4^{vir} &= \{t_1\} \\ D_5^{vir} &= \{t_4\} \\ D_6^{vir} &= \{t_5\} \end{aligned} \tag{3.4}$$

A new adjacency matrix $\boldsymbol{E}_X^{vir}$ is defined to incorporate additional virtual nodes. Such matrix is built simply by substituting the real node with the sequence of virtual nodes, so that the first virtual node is connected to the nodes from which node i received the data, while the last virtual node is connected to the node to which node i sent the data. The other nodes are connected in sequence. An exception happens if the real node can also perform some sensing functions. In this case, the corresponding virtual node is kept outside this sequence and it merely

sends the data to the subsequent virtual node. This rule has been introduced because the sensing operation does not need any data from other nodes. This scenario is depicted in the example in Figure 3.4, where the temperature sensing function can be performed by node x_4^{vir}.

A set of characteristic parameters $e_{vir,i}^{ins}$, $\boldsymbol{e}_{vir,ij}^{tx}$, $e_{vir,i}^{rx}$, and e_i^{res}, corresponding to those defined in Table 2.2, has to be associated to each virtual node. Taking into account the virtual nodes substituting node i, the transmission cost from a virtual node to the other must be null. Therefore, only the last virtual node, that has to transmit the data to the network, has the same characteristic parameters as node i: the virtual nodes before it must have the parameters $e_{vir,ij}^{tx}$ and $e_{vir,i}^{rx}$ set to 0.

A new matrix $\boldsymbol{\Delta}^{vir}$ is defined, where its elements are null for adjacent virtual nodes from the same original node. In the following, a network with virtual nodes will be considered; however, to make the presentation clearer, the subscript "vir" will be skipped, as it is unnecessary.

3.2.3 Cost Functions

The objective of the proposed algorithm is to evaluate the viability of each deployment solution on the basis of a cost function that is connected to energy consumption. Quite often in similar scenarios, past studies have proposed the evaluation of the network lifetime and have aimed at maximizing it. Since in the proposed framework more than one application is assumed to be assigned simultaneously to the network, there is no sense in computing the network lifetime since it is affected by other applications which are not considered in the same analysis. For this reason, the energy consumption is minimized for the application under analysis, allowing the network administrator to also include a parameter that takes into account the current node residual battery energy level, as shown in the following.

Three cost functions are considered: one for the sensing, one for the processing and one for the transmission.

The sensing cost function for node i is expressed as

$$P_i^{sens} = f_i^{out} \times \gamma_i \times e_i^{sens} \times y_i, \qquad \text{with } y_i = \begin{cases} 1 & \text{if } s_i \equiv \text{sensing code} \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

with e_i^{sens} representing the sensing energy consumption as defined in Table 2.2. Recall that f_i^{out} is the node output traffic frequency, which also represents the sensing frequency. The parameter γ_i is a coefficient in inverse proportion to the residual energy e_i^{res} of node i, which can be set to drive the deployment of the application towards nodes with higher residual energy levels, as anticipated above. When performing the experiments, γ_i has been set to 1 when the battery is fully loaded, while γ_i has been set to 5 when the battery level is lower than 20% of the total

charge. From 1 to 5, γ_i changes linearly.

The processing cost function is defined as follows

$$P_i^{proc} = \sum_{h=1}^{H} f_{ih}^{out} \times \gamma_i \times e_{ih}^{proc}(t_{s_i}, \mathbf{\Theta}_i^{in}) \times v_i \qquad \text{with } v_i = \begin{cases} 1 & \text{if } s_i \equiv \text{processing code} \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

where γ_i is the coefficient defined above, and e_{ih}^{proc} is the processing energy consumption defined by Equation 2.1, which depends on the energy e_i^{ins} spent by node i to perform a single instruction, the number of instructions I_h related to task h, and the received data $\mathbf{\Theta}_i^{in}$ described in Equation 3.2. Because P_i^{proc} depends on the number of processing per second performed by node i, it is proportional to the frequency f_{ih}^{out} of each of the H egress traffic flows, where H is the size of $\mathbf{\Theta}_i^{out}$ as described in Section 3.2.1. The number of samples to calculate e_i^{proc} is defined differently for each kind of processing detected in Section 3.2.1. For a spatial processing, the number of processed samples is equal to the number of ingress traffic flows; for a temporal processing, the number of processed samples for each traffic flow θ_{ih}^{out} is the ratio between the frequency of arrival of the samples f_{ih}^{in} and the processing frequency f_{ih}^{out}; by definition, for a single sample processing the number of processed samples is 1.

Both sensing and processing are followed by a transmission. The related cost function is

$$P_i^{tx} = \gamma_i \times \sum_{j=1}^{N} \sum_{h=1}^{H} f_{ih}^{out} \times k_{ih}^{out} \times (e_{ij}^{tx} + e_j^{rx}) \times e_{ij}^{X} \tag{3.7}$$

with f_{ih}^{out} and k_{ih}^{out} transmission frequency and number of bits for the egress traffic flow h, γ_i residual energy coefficient, e_{ij}^{tx} and e_j^{rx} transmission and reception energy consumption values defined in Table 2.2, and e_{ij}^{X} edge connecting node i to node j, as defined in Section 2.1.

Given Equations 3.5, 3.6, and 3.7, the overall cost function is

$$P^{tot} = \sum_{i=1}^{N} \left(P_i^{sens} + P_i^{proc} + P_i^{tx} \right) \tag{3.8}$$

3.2.4 Maximization of Network Lifetime

The goal of the maximization process is to find the set of the statuses $\boldsymbol{S} = \{s_1, ..., s_i, ..., s_N\}$ of the nodes that minimizes the network energy cost function.

Therefore, the optimization problem becomes

$$\text{minimize} \quad P^{tot} = \sum_{i=1}^{N} \left(P_i^{sens} + P_i^{proc} + P_i^{tx} \right) \tag{3.9a}$$

$$\text{subject to} \quad Q_l^{min} \leq \sum_{i=1}^{N} q_{il} \leq Q_l^{max} \qquad \text{with } q_{il} = \begin{cases} 1 & s_i \equiv t_l \\ 0 & \text{otherwise} \end{cases} \tag{3.9b}$$

$$\bigcup_{v} \{s_v\} = G_1 \qquad \forall v : v \mapsto 1 \tag{3.9c}$$

The condition in Equation 3.9b is a constraint on the minimum (Q_l^{min}) and the maximum (Q_l^{max}) number of nodes that have to perform the task t_l. This could be necessary, for example, when a given geographical area is monitored by a certain number of nodes, but the required information is not needed from all of them. If, for instance, the mean temperature value of an area monitored by 30 sensors is needed, it may be preferred the temperature information to be gathered just by 10 of those sensors, in order to consume less energy. In this case, both Q_l^{min} and Q_l^{max} would be equal to 10, and the algorithm would choose the 10 sensors which weight less on the network lifetime, among the 30 sensors which are able to sense temperature in the required area. When this constraint is not needed for a task l, Q_l^{min} is null and Q_l^{max} is set to N.

The condition in Equation 3.9c shows that the set of statuses of the virtual nodes corresponding to the original sink node 1 must correspond to the set $|G_1|$. This implies that all the virtual nodes corresponding to the original sink node are in a processing status, i.e. if there is any data still to be processed, those virtual nodes have to process them.

The problem defined in Equation 3.9 is a Mixed Integer Linear Programming (MILP) problem [74]: the unknown status of node i can be defined as a $|T|$-dimensional binary array, where T is the set of tasks as defined in Section 3.1. Because every node can only have one status, which means that it can perform only one task among those that it is able to perform, only one element of this array can be equal to 1, and it corresponds to one of the tasks that the relating node i is able to perform, according to D_i. The elements of the array represent the weights to the contributions (in Equations 3.6 and 3.5) of the node to the cost function.

MILP problems are classified as NP-hard. Their exact solution is usually found using branch-and-bound algorithms. The worst case complexity of branch-and-bound algorithms is the same as the complexity of exhaustive search, which means that its complexity scales exponentially with the problem size. In the case under consideration, the problem size is dominated by the number of tasks $|T|$ and the number of virtual nodes N^{vir}. Therefore, the worst case complexity would be $O(2^{|T| \times N^{vir}})$. Nevertheless, in most cases branch-and-bound is more efficient compared to exhaustive search. Furthermore, the problem's structure is such that only one element of the $|T|$-dimensional array representing the status of each node is nonzero. This condition allows to reduce the search space to $O(N^{vir|T|})$. It has to be noted that commercial

mathematical programming solvers such as CPLEX [75] or Xpress Optimization Suite [76] are claimed to use optimized branch-and-bound algorithms whose complexity scales linearly with the problem size.

In order to further reduce complexity, heuristic algorithms might be used as well, obtaining sub-optimal solutions which may be considered sufficient in most cases.

3.2.5 The Proposed Framework

Given a network similar to the one in Figure 3.1, the sink, which is responsible for initiating and maintaining the network, is the device on which the deployment algorithm is performed. The proposed algorithm needs to know the exact topology of the network, that is, how the nodes are connected to each other (i.e. matrix E_X) and what the distance between any two of them is (i.e. matrix Δ). In order to compute the cost function of Equation 3.8, further information is needed, such as the parameters to model the radio channel, the transmission, reception, sensing and processing energy consumption of each node, the residual energy of each node, the working frequency and the data rate.

In summary, the algorithm performs the following steps:

1. define DAG $\mathcal{G}_X$, matrices $\mathbf{\Delta}$ and $\mathbf{\Phi}$, set D_i, and parameters e_i^{sens}, e_i^{ins}, $\boldsymbol{e}_i^{tx}$, e_i^{rx}, and e_i^{res};
2. define DG $\mathcal{G}_T$;
3. define sets G_i;
4. define the new network with N^{vir} virtual nodes x_i^{vir}, and the characteristic parameters associated to it;
5. given Equation 3.9, solve it with a linear programming solver, in order to find set $\boldsymbol{S}$.

The solving algorithm has been implemented in Mosel language, and the solution has been found using Xpress Optimization Suite. The binary array $\Sigma_v^{vir} = \{\sigma_{v1}^{vir}, ..., \sigma_{vl}^{vir}, ..., \sigma_{vL}^{vir}\}$ has been associated to each node x_v^{vir}, where L is the cardinality of T, that is the number of tasks in which the application considered is subdivided, as described in Section 3.1. The elements of Σ_v^{vir} must satisfy the following constraints:

- the element σ_{vl}^{vir} can be equal to 1 if and only if x_v^{vir} is able to perform the task $t_l \in T$, which means that σ_{vl}^{vir} cannot be equal to 1 if the task t_l is not an element of the set of tasks D_v^{vir} that the node x_v^{vir} is able to perform

$$t_l \notin D_v^{vir} \Rightarrow \sigma_{vl}^{vir} \neq 1 \tag{3.10}$$

- only one element in Σ_v^{vir} can be equal to 1

$$\sum_{l=1}^{L} \sigma_{vl} = 1 \tag{3.11}$$

The elements of the array built this way are the weights y_i and v_i of the energy contributions in Equations 3.5 and 3.6 defined in Section 3.2.3.

The optimum way to accomplish the assigned application at issue and spend the least amount of energy as possible will then be found. The node or the combination of nodes that are able to perform it and consume the minimum amount of energy will be chosen; then, a low level code describing which tasks each node has to perform will be developed and distributed to the appropriate nodes.

3.3 Performance Analysis

3.3.1 Test Cases and Simulations Setup

To evaluate the effectiveness of the proposed strategy, three test cases have been taken into account according to some of the most significant realistic scenarios considered in past works, such as in [77]:

1. **case1**: uniform energy consumption and uniform initial energy (UC-UE) at each node (equal characteristic parameters and battery life for every node);
2. **case2**: non uniform energy consumption and uniform initial energy (NUC-UE) at each node (different characteristic parameters but same battery life for every node). The energy consumption of the nodes have been assigned according to a uniform distribution from 60% to 140% of the energy consumption for case1;
3. **case3**: uniform energy consumption and non uniform initial energy (UC-NUE) at each node (same characteristic parameters but different battery life for every node). The battery charge has been assigned randomly according to an uniform distribution from 20% to 100% of the total charge.

A rectangular-shaped outdoor environment (e.g., a vineyard, a seaport, a tourist plaza), divided into areas of $25\ m^2$, is monitored, where the nodes have been deployed with different densities:

- 0.2 nodes per square meter;

- 0.3 nodes per square meter;
- 0.4 nodes per square meter.

The nodes have been positioned randomly, following a uniform distribution. Each node is equipped with sensors for the measurement of temperature, humidity, PH and light exposure. The data are sent to the sink, which has identification number 1.

The analysis has been focused on the following two operations:

1. **OpA**: calculation and storage in the sink of the (temporal and spatial) mean values of temperature, humidity, PH and light exposure over an hour, starting from the values sensed every 10 minutes from every area;
2. **OpB**: aggregation of traffic coming from different areas of the network, carrying temperature, humidity, PH and light exposure values to the sink for later analysis by qualified staff.

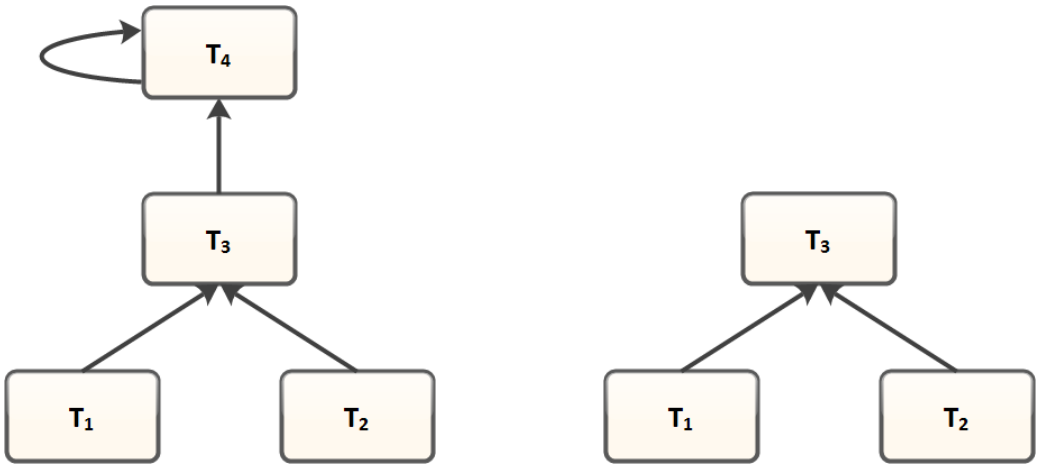
Each sensed value is assumed to be represented as a double numerical value, which is 64 bits long. Note that these two operations have been selected to compare the scenarios in which the network is required to perform significant data processing (OpA) and nodes have to perform only basic processing on the data, yet can significantly reduce the amount of transmitted data by aggregating the sensed samples (OpB).

For both OpA and OpB, the first tasks are the sensing ones: t_1 is the *temperature, humidity, PH and light exposure sensing for area* 1, t_2 is the *temperature, humidity, PH and light exposure sensing for area* 2, and so on for every area. In order not to weigh down the text with alternatives, just 2 areas are take into account, which is not the case of the simulation scenario. Similarly to the *spatial and temporal monitoring example* in Section 3.1, for OpA t_3 is *temporal mean*, and t_4 is *spatial mean*. In addition to t_1 and t_2, OpB has the *aggregation of samples* task t_3. The tasks for the two operations are summarized in Table 3.2 and described by their DGs in Figures 3.5(a) and 3.5(b).

The nodes communicate using IEEE 802.15.4 radio interfaces on the 2.4 GHz frequency band. To keep things simple, any possible overhead has not been taken into account.

Table 3.2: Tasks for OpA and OpB, for two monitored areas

t_i	OpA	OpB
t_1	Temperature, humidity, PH and light exposure sensing for area 1	
t_2	Temperature, humidity, PH and light exposure sensing for area 2	
t_3	Temporal mean	Aggregation of samples
t_4	Spatial mean	——



(a) OpA (b) OpB

Figure 3.5: DG for the tasks of OpA (a), and OpB (b), considering two monitored areas

Table 3.3: Simulation setup parameters

Parameter	**Value**
RF frequency	2400 MHz
Bit rate	250 kbps
Programmable output power range	Programmable in 8 steps from approximately -24 to 0 dBm
Receiver sensitivity	-94 dBm
P_{R0}	59.2 mW
P_{T0}	26.5 mW
η	50%
A	14 dB
e^{instr}	1 nJ
Packets header	12 bytes
Packets maximum payload	125 bytes

The resulting scenarios have been simulated in MatLab environment, where the proposed algorithm as been implemented along with alternative approaches as discussed in the following Subsection. The main setup parameters are listed in Table 3.3. More specifically, processing cost parameters can be found in [73], radio frequency parameters specified in [78], and IEEE 802.15.4 parameters listed in [39]).

3.3.2 Analysis of Case Studies

The optimization algorithm has been applied to each of the cases mentioned in 3.3.1. The resulting cost value has been compared with:

1. the cost value that is obtained if data are processed only by the sink. This means that each traffic flow generated by the sensors is sent to the sink without any processing at the intermediate nodes. This comparison is referred to with the letter *S*;
2. the data are processed (whenever needed) by every cluster head [1] found in the path to the sink. This comparison is referred to with the letters *CH*;
3. the mean cost value for all possible solutions that might be detected. This is introduced to make a comparison with a possible solution where the processing of the data is performed on fixed nodes, which is expected to bring results corresponding to the median solution. This comparison is referred to with the letter *M*.

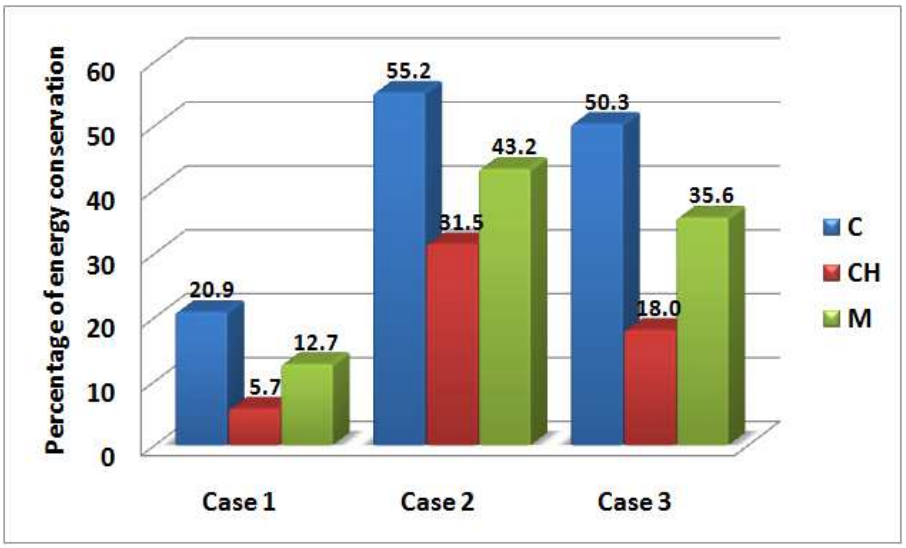
These comparisons are expressed as a percentage of the energy conservation that would result using the proposed technique with respect to the alternatives one.

Figure 3.6 shows the results for the two operations.

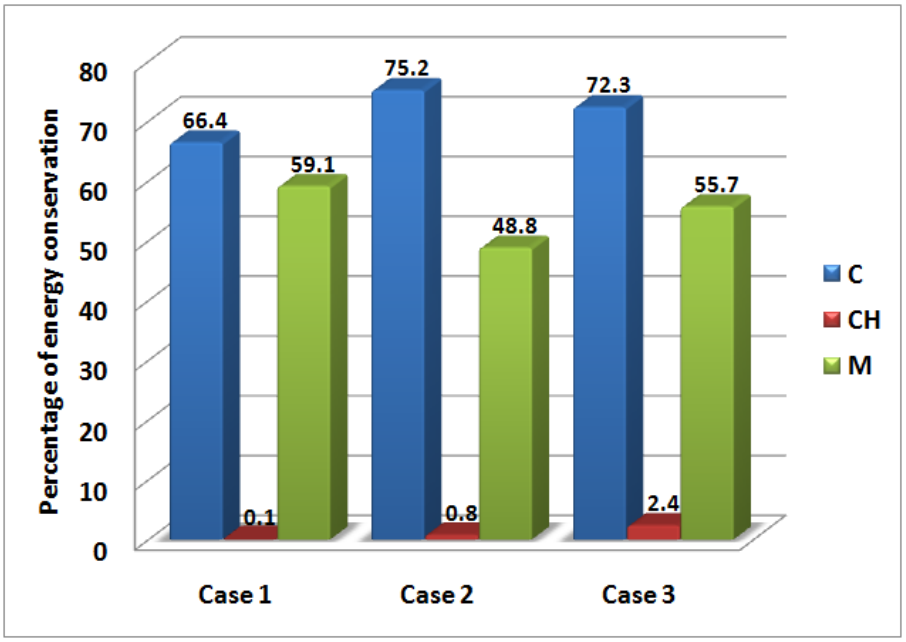
The two graphs show significant improvements of the proposed strategy with respect to the alternative ones with an average improvement of 36.3%. Limited benefits are observed in case UC-UE for both OpA and OpB. In fact, when all the nodes have the same parameters, and thus have a uniform energy consumption and the same initial energy, the choice of which node will perform the processing boils down to which cluster heads will do it. The scenario is illustrated in Figure 3.7, which depicts a cluster head CH1 connected to the cluster head CH2 by nodes 1, 2, and 3. Note that the cluster heads are just nodes that receive more than one traffic flow from different links. Because processing in CH1 weights on the network as much as processing on node 1, node 2 or node 3, any processing of the data before arriving to CH2 is more energy conserving. Processing the data on CH1 ensures spending less transmission energy than processing data on nodes 1, 2 or 3. Accordingly, the CH approach allows for obtaining results similar to the ones obtained with the proposed approach in case UC-UE. Slightly better results are obtained because when cluster heads are close to each other sometimes it is better to perform the processing only in the second cluster head rather than in both of them.

On the contrary, in cases NUC-UE and UC-NUE, devices' energy consumption does not weight the same amount on the entire network. This means that the nodes chosen by the proposed algorithm to perform the processing will be those that weight less on the network, regardless of whether they are cluster heads or not. Therefore, the detection of the lower cost solution determines the best results, in terms of energy consumption, for networks with heterogeneous parameters, which are the most common type of networks in real scenarios.

[1] For the definition of cluster heads see Section 1.1



(a) OpA



(b) OpB

Figure 3.6: Percentage of energy conservation using the proposed framework, for OpA and OpB

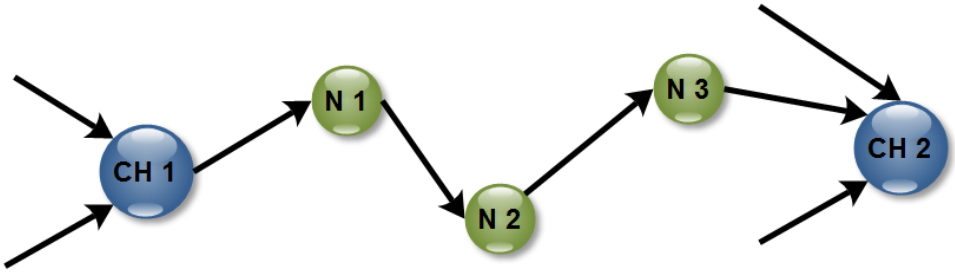


Figure 3.7: Example of a transmission from cluster head CH1 to cluster head CH2

The benefits with respect to the approach CH are lower than the case of using approaches S and M. In fact, literature dictates that the use of cluster heads is a convenient solution, because the aggregation of frames coming from different paths leads to a reduction in network energy consumption. For this reason, when using cluster heads the cost is much lower, compared to sending every single frame to the sink, or to the average of the other possible solutions; this determines less difference from the optimization algorithm solution, and thus a lower energy conservation. However, this approach requires every node in the network to be able to perform data processing, which is not always the case. In any case, the proposed approach is proven to always outperform the CH approach.

It could be noted that for OpA, in which processing is more elaborate and the number of instructions for every process greater, energy conservation is higher than it is for OpB. In fact, as could be expected, the lower the energy cost necessary for the processing, the more convenient it is to process the data in every cluster head encountered.

This fact is demonstrated by the results shown in Figure 3.8, which depicts the percentage of energy conservation while the ratio between the processing cost and the cost to transmit 137 bytes of data increases. The distance considered is equal to the average distance of all the nodes from the sink. Comparison has been made both in the case that data are processed only by the sink (solid lines) and cases in which data are processed by every cluster head (dashed lines). In the former, energy conservation decreases when processing cost to transmission cost ratio

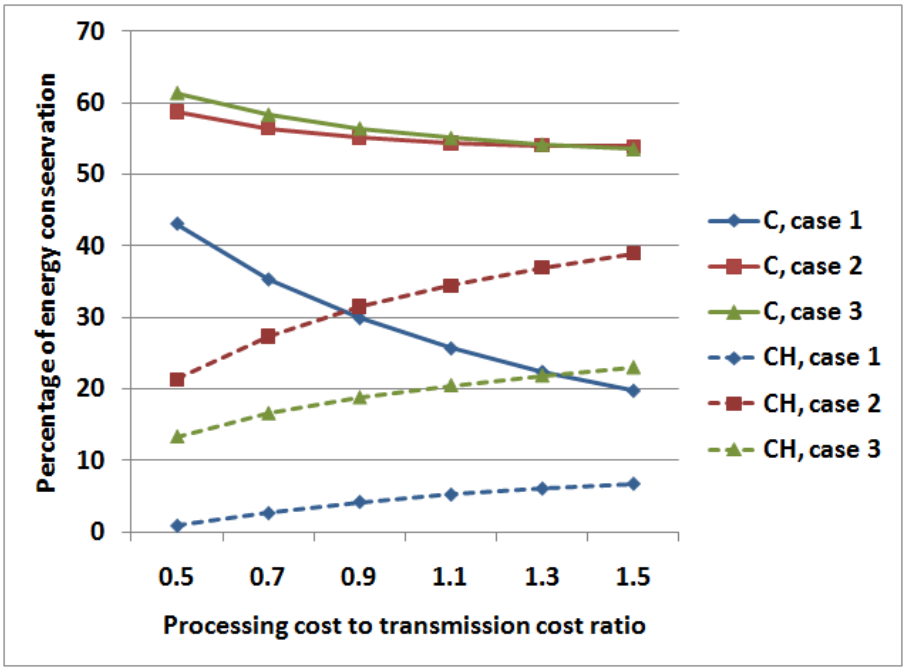


Figure 3.8: Percentage of energy conservation with respect to the ratio between processing cost and the cost to transmit 137 bytes of data. Solid lines show energy conservation with respect to data processed only by the sink; dashed lines show energy conservation with respect to data processed by every cluster head

increases. In fact, as could be expected, when the processing cost increases with respect to the transmission cost, it becomes more convenient to transmit data rather than process data. On the other hand, when compared with the CH approach, energy conservation increases when the processing cost to transmission cost ratio increases. In fact, when the processing cost increases, it is more convenient to accurately choose the nodes where processing might be performed rather than processing data every time that it is possible to do so.

Table 3.4 shows the results for OpA and OpB, for different node density of 0.2 and 0.4 $nodes/m^2$. The resulting tendency of an improved energy conservation when node density increases is basically due to two factors:

- in cases NUC-CE and UC-NUE, when the number of nodes in the same area increases, it is more likely that among neighboring nodes there are nodes where the processing cost is lower;
- the higher the number of nodes in the same area, the higher the number of clusters formed, and therefore the bigger the amount of data that can be processed before they arrive to the sink, reducing the energy cost.

It may be inferred from the results that using the framework would be particularly energy conserving when data from different nodes have to be processed together, the processing is pretty complex, and the energy consumption or the initial energy is not uniform for the network.

When the network experiences the failure of a node, that node must be bypassed and data addressed to it must be sent to the following node. In order to do it, an appropriate new routing path must be find. The number of packets exchanged to find a new routing path depends on the routing algorithm used by the network. Supposing to be in a bad case scenario where an average of 50 packets have to be sent among 10-hops distant nodes where each node is 2 m far from the other ones, supposing to have one node failure every hour, energy conservation would decrease by about 7.8%. It has been estimated that the decrease in energy conservation for OpA and for

Table 3.4: Percentage values of energy conservation using the proposed framework

	Node density $[nodes/m^2]$	**Case 1 [%]**			**Case 2 [%]**			**Case 3 [%]**		
		S	**CH**	**M**	**S**	**CH**	**M**	**S**	**CH**	**M**
OpA	0.2	19.5	5.5	11.7	50.0	29.3	42.2	47.5	16.4	33.6
	0.4	25.6	5.9	16.5	58.0	35.3	46.2	56.7	23.5	43.7
OpB	0.2	28.9	0.1	17.0	33.3	0.7	19.9	38.6	1.6	19.7
	0.4	30.8	0.1	18.2	37.8	1.0	21.4	40.9	3.5	21.1

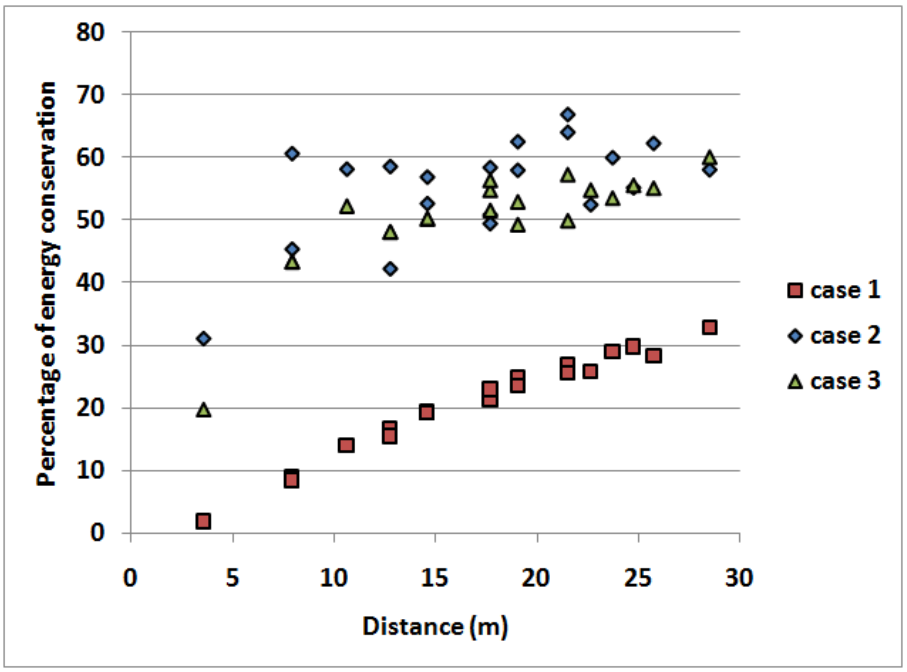


Figure 3.9: Percentage of energy conservation for every area of the network for OpA, comparison S, for a density of $0.3\ nodes/m^2$, for an increasing distance of the area from the sink

a density of $0.3\ nodes/m^2$ would be about 8.2% in case UC-UE, 12% in csae NUC-UE and 14.7% in case UC-NUE.

A final observation may be made in regard to Figure 3.9, which depicts the percentage of energy conservation for OpA, comparison S, in cases UC-UE, NUC-UE and UC-NUE, for a density of $0.3\ nodes/m^2$, in relation to the distance of each area from the sink. As could be expected, the greater the distance of the sources from the sink, the more energy conservation is derived from the use of the framework.

Although they have not been reported, similar results have been obtained for all other cases.

Chapter 4

Decentralized Lifetime Maximization Algorithm

In the previous Chapter, a centralized solution for the optimal task assignment that ensures that the overall energy consumption is minimized has been described. However, centralized algorithms suffer from computational complexity, especially for WSNs with a high number of nodes. Furthermore, centralized algorithms have to frequently collect updates from nodes in order to adapt to network dynamism. This incurs a large control packet overhead.

In this Chapter, a distributed solution that improves the capability of the network to adapt to energy consumption changes, to reduce the message exchange overhead for the application deployment, and to provide a solution whose computational complexity scales well with the number of nodes. The proposed algorithm [71], is based on an iterative and asynchronous local optimization of the task allocations among neighboring nodes. The resulting scheme is based on *gossip*, which consists in a communication paradigm in which, at each instant of time, each node in the network has some positive probability to interact with one of its neighbors [79].

A first contribution consists in a decentralized and asynchronous algorithm, called MLE, that provides a simple local interaction rule between sensor nodes. This rule allows each node in the network to estimate in known and finite time the network lifetime within the set of nodes that compose their routing paths toward the root node sink. A second contribution consists in a decentralized and asynchronous algorithm, called *Decentralized Lifetime Maximization Algorithm* (DLMA), that maximizes the lifetime in the network in a distributed way without assuming any knowledge of network topology or global state information, apart from the information provided by Algorithm MLE. A significant feature of this algorithm is that the lifetime is maximized monotonically as function of time despite adopting a distributed approach. The algorithm consists in the update of the nodes' state with the solution of optimization problems that involve only local state variables. These updates do not need to be executed in any specific order, thus the algorithm can be executed with asynchronous updates that greatly simplify its

implementation in large-scale networks.

This Chapter is organized as follows. Section 4.1 defines the problem and introduces the notation used. Section 4.2 deals with the traffic flows analysis and the definition of the cost function. Section 4.3 describes the decentralized solution aimed at maximizing the network lifetime. Section 4.4 focuses on simulation results.

4.1 Problem Formulation

The considered scenario is analogous to the one described in Section 3.1. Again, the WSN under consideration is the one illustrated in Figure 3.1, where not all the nodes have the same capabilities. The model of the network has already been described in Section 2.2. The sink is referred to as node 1.

Let $\mathcal{N}_i = \{j \in X : (i,j) or (j,i) \in E_X\}$ be the neighborhood of node i, namely the nodes that share a communication channel with node i. Let $\mathcal{N}_{out,i} = \{j \in X : (i,j) \in E_X\}$ be the set of nodes that receive information from node i and $\mathcal{N}_{in,i} = \{j \in X : (j,i) \in E_X\}$ the set of nodes that send information to node i to reach the sink node 1. These two sets are defined with respect to the flowing of data from the sensors toward the sink.

As an example, consider the network shown in Figure 2.1, where to each node is assigned the corresponding ID. The sink node has always ID equal to 1. Furthermore, node 11 has in-neighborhood $\mathcal{N}_{in,11} = \{12, 13\}$ and out-neighborhood $\mathcal{N}_{out,11} = \{4, 10\}$. Note that the arrows represent the direction of the traffic when flowing from the sensors to the sink.

Once again, an application assigned to the network, which can be decomposed into a sequence of tasks, is considered. As far as tasks are concerned, two sequences are distinguished: one for the sensing tasks and one for the processing tasks. The sequence of the sensing tasks that have to be executed by the network is defined as $T^s = \{t_1^s, ..., t_W^s\}$, where W is the total number of sensing tasks required. The sequence of the processing tasks that have to be executed by the network is defined as $T^p = \{t_1^p, ..., t_L^p\}$, where L is the total number of processing tasks required. If, for instance, the application is the *spatial and temporal monitoring* example introduced in Section 3.1, the set of sensing tasks is $T^s = \{t_1^s, t_2^s, t_3^s\}$, where t_1^s is the *temperature sensing in area* 1, t_2^s is the *temperature sensing in area* 2, and t_3^s is the *temperature sensing in area* 3. The set of processing tasks is $T^p = \{t_1^p, t_2^p\}$, with t_1^p *temporal mean* and t_2^p *spatial mean*. The relation among tasks can be described as a DG $\mathcal{G}_T = (\{T^s, T^p\}, E_T)$, where $E_T = (e_{uv}^T)$ is the set of edges, with each edge e_{uv}^T representing a unidirectional data transfer from task u to task v. Figure 4.1 shows the DG for the *spatial and temporal monitoring* example.

As to the sensing, a binary state $\boldsymbol{m}_i \in \{0,1\}^W$ coding the sensing tasks executed by node

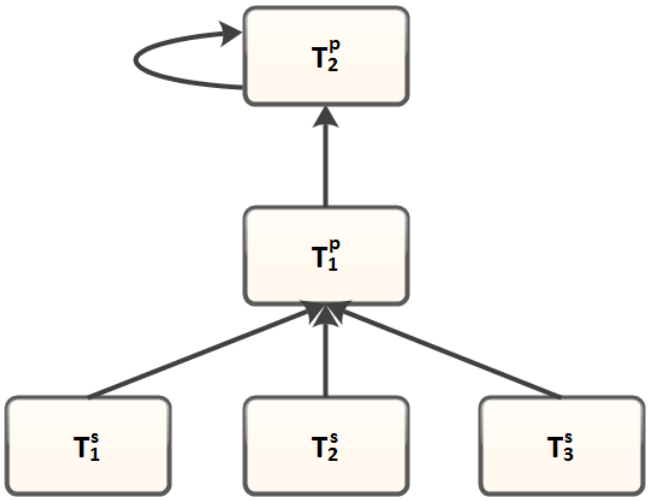


Figure 4.1: DG for the *spatial and temporal monitoring* example

i is assigned to each node. The state m_{iw} is equal to 1 if node i performs the sensing task w.

As to the processing tasks, a binary state vector $\boldsymbol{s}_i \in \{0,1\}^L$ representing the processing tasks currently assigned to the node i is assigned to each node. To each configuration of the node setting corresponds different energy consumption, as it will be better explained in the following Sections. Different nodes are assumed to consume different amounts of energy for the same processing to include heterogeneity of devices in the modeling. To each node i is associated a binary vector $\boldsymbol{d}_i \in \{0,1\}^L$, which represents the kinds of processing that node i is allowed to execute. In particular, the following holds: $d_{il} \geq s_{il}, \quad \forall i \in X$ and $\forall l \in \{1,\ldots,L\}$. To simplify the notation the matrix that collects the states of all nodes is denoted as $\boldsymbol{S} = [\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_N]^T$ and the matrix that represents all constraints as $\boldsymbol{D} = [\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_N]^T$.

In the considered scenario, the sensing tasks are already assigned to the network nodes (i.e., binary vectors $\boldsymbol{m}_i$ are given, for $i = 1,\ldots,N$). Differently, the processing tasks can be executed according to different solutions: gathered data can be immediately sent to a sink, or it can be processed before being transmitted. In the case of the latter, the number of bits to be sent would be smaller, reducing the transmission energy consumption; however, processing energy consumption could be higher in this second case. Quantifying the energy consumption in both cases, it could be possible to establish which one determines a reduction of battery consumption in the nodes, incrementing the network lifetime. The addressed problem is then defined as the processing status matrix $\boldsymbol{S}$ that minimizes the impact of the application on the network, maximizing the *lifetime of the network* τ, intended as the time in which at least one node has exhausted its energy reserve from the battery: in fact, when this condition is reached, the network topology is disrupted.

Denoting the energy reserve of node i at time t as $e_i^{res}(t)$, then the lifetime of the network

can be defined as

$$\tau = \inf\{t \mid (\exists i \in X) e_i^{res}(t) = 0\} \tag{4.1}$$

by assuming that all the transmission rates at the nodes are constant and that the processing assignment defined by matrix $\boldsymbol{S}$ does not change. Furthermore, it can be observed that the optimal processing assignment is

$$\boldsymbol{S}_{opt} = \arg\max_{\boldsymbol{S}} \tau(\boldsymbol{S}) = \arg\max_{\boldsymbol{S}} \min_{i \in X} \frac{e_i^{res}}{P_i(\boldsymbol{S})},$$

where P_i is the energy consumed per unit of time. The last equation puts in evidence the dependence of τ and P_i from the processing assignment $\boldsymbol{S}$ and follows from the fact that in a stationary state the time required for a node to drain its battery is e_i^{res}/P_i.

Since the tasks a node can process are limited, a constrained optimization problem of the form

$$\begin{aligned} &\max_{\boldsymbol{S}} \min_{i \in X} \frac{e_i^{res}}{P_i(\boldsymbol{S})}, \\ &s.t. \\ &\boldsymbol{d}_i \geq \boldsymbol{s}_i \quad \forall i \in X \\ &\boldsymbol{S} \in \{0,1\}^{N \times L} \end{aligned} \tag{4.2}$$

is obtained, where $\boldsymbol{d}_i$, the i-th column of matrix $\boldsymbol{D}$, is the characteristics vector of the tasks that can be processed by node i and $\boldsymbol{s}_i$, the i-th column of matrix $\boldsymbol{S}$, is the processing state of node i. In the following, the considered scenario is elaborated by defining the traffic flows and cost function. Then, the proposed distributed solution will be illustrated.

4.2 Analysis of Traffic Flows and Energy Consumption

4.2.1 Traffic Flows

In the reference scenario, the sources of traffic in the network (the sensors) are assumed to generate samples of k bits at a certain frequency f. The generic node i receives a number of *incoming traffic flows* represented by vector $\boldsymbol{\Theta}_i^{in} = (\theta_{i1}^{in}, ..., \theta_{ih_i'}^{in})$, where each element is a traffic flow (expressed in bit per second), where $h = 1, \ldots, h_i'$ are the labels of the traffic flows incoming from its in-neighbors. The value h_i' is the maximum number of traffic flows that can be received by node i. Over these traffic flows, node i performs the tasks corresponding to its assigned status $\boldsymbol{s}_i$. As a result from this processing, the generation of the *output traffic flows* $\boldsymbol{\Theta}_i^{out} \in \mathbb{R}^{h_i'}$, can be modeled as

$$\boldsymbol{\Theta}_i^{out} = p_i(\boldsymbol{\Theta}_i^{in}, \boldsymbol{s}_i), \tag{4.3}$$

where $p_i(\cdot) : \mathbb{R}^{h'_i} \times \{0,1\}^L \rightarrow \mathbb{R}^{h'_i}$ is a look-up table of node i that makes a correspondence between the incoming traffic flows, the processing performed by node i on these traffic flows and the output traffic of the node. Note that the number of elements in $\mathbf{\Theta}_i^{in}$ is set equal to those of $\mathbf{\Theta}_i^{out}$, that is, h'_i. Some of these elements can be null depending on the type of processing performed by node i. The generated output traffic is sent to the downstream nodes in the out-neighborhood of node i.

The elements of vector $\mathbf{\Theta}_i^{out} = (\theta_{i1}^{out}, ..., \theta_{ih'_i}^{out})$ correspond to traffic flows with samples of size k_{ih}^{out} bits transmitted at frequency f_{ih}^{out}. The data $\mathbf{\Theta}_i^{out}$ is then sent to the downstream nodes $j \in \mathcal{N}_{out,i}$ according to a DG $\mathcal{G}_X$.

If node i is aware of the incoming traffic flows $\mathbf{\Theta}_j^{in}$, the processing assignment $\boldsymbol{s}_j$ and the look-up tables of its in-neighbors $j \in \mathcal{N}_{in,i}$, then it can compute its output traffic flows as

$$\mathbf{\Theta}_i^{out} = p_i \left(\sum_{j \in \mathcal{N}_i} p_j \left(\mathbf{\Theta}_j^{in}, \boldsymbol{s}_j \right), \boldsymbol{s_i} \right). \tag{4.4}$$

There are many processing tasks that can be performed in a WSN. For each one of these, the look-up table of each node $p_i(\cdot)$ has to be experimentally identified. For the objective set, this operator is needed to figure out the traffic flows that will be traversing the network for each deployment scenario. Three common kinds of processing are identified: spatial, temporal and single sample processing. For further details about this distinction, see Section 3.1.

4.2.2 Cost Functions

The objective of the proposed algorithm is to evaluate the viability of each deployment solution on the basis of cost functions that are connected to power consumption. Three cost functions are considered: one for the sensing, one for the processing and one for the transmission. Each cost function represents the amount of energy consumed in the unit of time.

The sensing cost function for node i is expressed as

$$P_i^{sens} = \sum_{w=1}^{W} f_i^{out} \times e_{iw}^{sens} \times m_{iw}, \tag{4.5}$$

with e_{iw}^{sens} representing the sensing energy consumption for node i performing sensing task w if its status m_{iw} is equal to 1, as defined in Section 2.1. Recall that f_i^{out} is the node output traffic frequency, which also represents the sensing frequency.

The processing cost function for node i is defined as

$$P_i^{proc} = \sum_{l=1}^{L} \sum_{h=1}^{h'_i} f_{ih}^{out} \times e_{ih}^{proc}(t_{s_{il}}^p, \mathbf{\Theta}_i^{in}) \times s_{il}, \tag{4.6}$$

where e_{ih}^{proc} is the processing energy consumption defined in Equation 2.1, which depends on the task $t_{s_{il}}^{p}$ that has to be executed and on the input traffic $\mathbf{\Theta}_i^{in}$. The status s_{il} of node i with respect to each possible task l determines whether this energy consumption should be considered or not. Since the processing cost depends on the number of processing instructions per second performed by node i, P_i^{proc} is proportional to the frequency f_{ih}^{out} of each of the h egress traffic flows, where h_i' is the size of $\mathbf{\Theta}_i^{out}$ as described in Section 4.2.1.

Both sensing and processing are followed by a transmission. The related cost function is

$$P_i^{tx} = \sum_{j=1}^{N} \sum_{h=1}^{H} f_{ih}^{out} \times k_{ih}^{out} \times (e_{ij}^{tx} + e_j^{rx}) \times e_{ij}^{X} \tag{4.7}$$

with f_{ih}^{out} and k_{ih}^{out} transmission frequency and number of bits for the egress traffic flow h, e_i^{res} residual energy coefficient, e_{ij}^{tx} and e_j^{rx} transmission and reception energy consumption values defined in Table 2.2, and e_{ij}^{X} edge connecting node i to node j, as defined in Section 2.1.

Given Equations 4.5, 4.6 and 4.7, the overall cost function for any node i is

$$P_i = \left(P_i^{sens} + P_i^{proc} + P_i^{tx}\right). \tag{4.8}$$

In the following, the cost function for node i is denoted as $P_i(\boldsymbol{S})$. This is to show that the energy cost depends on the task assignment on the network represented by matrix $\boldsymbol{S}$. With a slight abuse of notation, the cost function of node i may also be denoted as $P_i(\boldsymbol{s}_k : k \in \mathcal{N}_{in,i} \bigcup \{i\})$ to show that — assuming the input flows $\mathbf{\Theta}_j^{in}$ for $j \in \mathcal{N}_{in,i}$ are known — it depends on the tasks assigned to node i and to its in-neighbors.

4.3 Proposed Distributed Solution

In the proposed framework of decentralized power saving for WSNs, a solution in which the communication scheme is based on *gossip* [79],[80],[81],[82] is adopted. Gossip algorithms are decentralized and asynchronous. They consist in a communication scheme in which at each instant of time each node in the network has some positive probability to interact with one of its neighbors. By the iterative interaction between nodes, several examples of emerging behaviors have been developed such as load balancing [83], distributed averaging [80], distributed convex optimization over networks [84], failure detection [85] and many more. Thus, communication schemes based on *gossip* that mimic the act of gossiping in a crowd of people, are easy to implement, do not require network routing or multi-hop communications, are inherently asynchronous and decentralized in nature.

The aim of the proposed solution is to solve the problem in Equation 4.2 in a decentralized way, by iteratively and asynchronously solving an equivalent local optimization problem that involves at each iteration only one node i and its in-neighbors $\mathcal{N}_{in,i}$.

The proposed method involves two decentralized algorithms. First, Algorithm 1, called MLE, 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 what is the minimum node lifetime in the set of nodes included in all the paths between itself and the sink.

Second, Algorithm 2, called DLMA, executes an iterative local optimizations (described in Definition 4.3.2) 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 1 to iteratively execute the local optimizations of Algorithm (DLMA) 2, while guaranteeing a monotonic increase in the network lifetime.

In the next two Subsections, the MLE and DLMA algorithms will be described. Then, the node selection process that drives the execution of the DLMA in the network nodes and the stop criterion will be analyzed. Finally, the algorithm computational complexity will be determined.

4.3.1 Algorithm 1 - Minimum Lifetime Estimation

In the DAG $\mathcal{G}_X = \{X, E_X\}$ that represents the network topology, 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. Let $\mathcal{X}_i = \{j \in X : \exists \mathbf{p}_{iN}\}$ be the set of nodes through which node i is reachable through a path from 1 (the sink). For example, with reference to Figure 2.1, considering nodes 3 and 1, the relevant sets are $\mathcal{X}_3 = \{10, 13\}$ and $\mathcal{X}_1 = \{3, 4, 8, 9, 10, 12\}$, respectively. 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 $\mathcal{X}_i$.

Let $t_1, t_2, \dots t_k$ with $k \in \mathbb{N}^+$ beset a set of instants of time which represent the instants of time in which local state updates are performed. The sink node 1 sends to all its in-neighbors $j \in \mathcal{N}_{in,1}$ the value of its lifetime τ_1. All the other nodes, at each instant of time t_k, 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

$$x_i(t_{k+1}) = \min\{\tau_i, x_j(t_k)\}. \tag{4.9}$$

The minimum lifetime estimation algorithm can now be stated

Algorithm 1 (Minimum Lifetime Estimation)**.**

1. *Each node $i \in X$ a variable $x_i(t_0) = \tau_i$ is initialized with the node lifetime.*

2. ***While*** *(true)*
3. *At time t_k, the sink $i = 1$ sends the value of $x_1(t_k)$ to all the nodes in its in-neighborhood $\mathcal{N}_{in,i}$.*
4. *Each node i that at time t_k receives a message from a parent node updates its local estimation according to*

$$x_i(t_k) = \min\{\tau_i, x_j(t_k)\},$$

and at t_{k+1} sends $x_i(t_{k+1})$ to in-neighborhood $\mathcal{N}_{in,j}$.

5. ***end while***. ■

Algorithm 1 can be seen as a wave of messages triggered by the sink node that propagates from the sink 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 4.3.1. *Let a network of N nodes that execute Algorithm 1 be described by a rooted connected DAG $\mathcal{G}_X$. Assume that a message reception and transmission cycle is performed by each node in at most T units of time. Let D be the length of the longest directed path in $\mathcal{G}_X$ that starts from the sink node. Assume that at $t = t_0$ the sink node initiates Algorithm 1.*

Then, in at most $T \cdot D$ units of time, each node i is aware of a lower bound on the minimum life time of the nodes in set $\mathcal{X}_i$

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

Proof. By assumption, there exists a sink node $i = 1$ from which each other node is reachable. Let d be the minimum number of edges that separate the generic node from the sink node. After the sink node sends the first message to its in-neighbors, at time Td all the nodes whose minimum path in $\mathcal{G}_X$ from the sink node is less than or equal to d have performed the local state update setting their state to the minimum value found along the path. By assumption, since $\mathcal{G}_X$ is connected and all nodes are reachable from the sink, all nodes are triggered. Since $\mathcal{G}_X$ is acyclic, when the furthest node j whose distance is at most D edges from the sink is reached, the algorithm stops. Thus, after $T \cdot D$ units of time, the state of each node is

$$\forall i \in X, \quad x_i(t) = \min_{j \in \mathcal{X}_i}\{x_j(t_D)\} = \min_{j \in \mathcal{X}_i}\{\tau_j(t - T \cdot D)\}. \tag{4.10}$$

■

In the following it is assumed that the sink node initializes 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 that will be presented in the next Subsection. On the contrary, the execution of Algorithm DLMA requires the information about the network lifetime that is computed by Algorithm MLE.

4.3.2 Algorithm 2 - Decentralized Lifetime Maximization Algorithm

The focus will be now shifted to the core of the proposed technique, namely the DLMA.

The DLMA consists in a local state update rule that neighboring nodes apply to their state if triggered by an incoming request message. Since the execution of the proposed state update has an energy cost, let $\hat{E}_i$ be the upper bound to the energy spent by solving the local optimization in node i. Then, the node that is expected to execute the optimization sees its lifetime reduced to

$$\tau_i = \frac{e_i^{res} - \hat{E}_i}{P_i},$$

thus, a total decrement in absolute terms of

$$\Delta\tau_i = \frac{\hat{E}_i}{P_i}.$$

Since the objective of the proposed algorithm is to iteratively maximize the lifetime, $\tau_i(t_k)$ is denoted as the lifetime that node i has at time t_k defined as

$$\tau_i(t_k) = \inf\{t \geq t_k | e_i^{res}(t) = 0\},$$

under the assumption that no more local optimizations are performed. Clearly, at each iteration of the algorithm, the lifetime of each node may change. Furthermore, in the following t_k will be referred to as the instant of time in which the local state update rule is triggered and t_{k+1} as the instant of time right after the given update is completed.

Such local state update rule is always triggered by a node toward its in-neighbor nodes. In the following, the dependencies of Θ_i^{out} on its input parameters will be implied.

Local state update rule 4.3.2.

Let node i be triggered.

1. *Node i, with a current processing state $\bar{\boldsymbol{s}}_i$ and current output traffic flow $\bar{\Theta}_i^{out}$ enquires the state $\bar{\boldsymbol{s}}_j$ and current input traffic $\bar{\Theta}_j^{in}$ for all $j \in \mathcal{N}_{in,i}$.*

2. ***If** $\tau_i > min_{j \in \mathcal{N}_{in,i} \bigcup \{i\}} \tau_j + \Delta\tau_i$, **then** solve the following local mixed integer linear programming problem in the unknown variables α and $\boldsymbol{s}_j$ for*

$j \in \mathcal{N}_{in,i} \bigcup \{i\}$

$$
\begin{aligned}
&\min \quad \alpha \\
&s.t. \\
&\frac{P_j(\boldsymbol{s}_k : k \in \mathcal{N}_{in,i} \bigcup \{i\})}{e_j^{res}} < \alpha \qquad \forall j \in \mathcal{N}_{in,i} \\
&\frac{P_i(\boldsymbol{s}_k : k \in \mathcal{N}_{in,i} \bigcup \{i\})}{e_i^{res} - \hat{E}_i} < \alpha \\
&\boldsymbol{d}_j \geq \boldsymbol{s}_j \qquad \forall j \in \mathcal{N}_{in,i} \bigcup \{i\} \\
&\boldsymbol{\Theta}_i^{out} \leq \bar{\boldsymbol{\Theta}}_i^{out} \\
&\alpha \in \mathbb{R}^+ \\
&\boldsymbol{s}_i \in \{0,1\}^L \qquad \forall j \in \mathcal{N}_{in,i} \bigcup \{i\}
\end{aligned}
\tag{4.11}
$$

and let its optimal solution be $(\alpha_{opt}, \boldsymbol{s}_{opt,k} : k \in \mathcal{N}_{in,i} \bigcup \{i\})$.

Set the new processing assignment to $\boldsymbol{s}_j = \boldsymbol{s}_{opt,j}$, *for all* $j \in \mathcal{N}_{in,i} \bigcup \{i\}$.

endif.

If $\tau_i > x_i + \Delta\tau_i$, ***then*** *solve the following local MILP problem [74] in the unknown variables* $\boldsymbol{s}_j$ *for* $j \in \mathcal{N}_{in,i} \bigcup \{i\}$. *Let* $\alpha = \max_{j \in \mathcal{X}_i} \frac{1}{\tau_j}$

$$
\begin{aligned}
&\min \quad \boldsymbol{\Theta}_i^{out}(\Theta_j^{in}, \boldsymbol{s}_j : j \in \mathcal{N}_i, \boldsymbol{s}_i) \\
&s.t. \\
&\frac{P_j(\boldsymbol{s}_k : k \in \mathcal{N}_{in,i} \bigcup \{i\})}{e_j^{res}} < \alpha \qquad \forall j \in \mathcal{N}_{in,i} \\
&\frac{P_i(\boldsymbol{s}_k : k \in \mathcal{N}_{in,i} \bigcup \{i\})}{e_i^{res} - \hat{E}_i} < \alpha \\
&\boldsymbol{d}_j \geq \boldsymbol{s}_j \qquad \forall j \in \mathcal{N}_{in,i} \bigcup \{i\} \\
&\boldsymbol{\Theta}_j^{out} \leq \bar{\boldsymbol{\Theta}}_j^{out}, \qquad \forall j \in \mathcal{N}_{in,i} \\
&\boldsymbol{s}_i \in \{0,1\}^L \qquad \forall j \in \mathcal{N}_{in,i} \bigcup \{i\}
\end{aligned}
\tag{4.12}
$$

and let its optimal solution be $(\boldsymbol{s}_{opt,j} : j \in \mathcal{N}_{in,i} \bigcup \{i\})$. *Set the new processing status to* $\boldsymbol{s}_j = \boldsymbol{s}_{opt,j}$, *for all* $j \in \mathcal{N}_{in,i} \bigcup \{i\}$.

endif. ■

Remark 4.3.3. *Note that the conditions that trigger the execution of the optimization algorithms*

$$\tau_i > \min_{j \in \mathcal{N}_{in,i} \bigcup \{i\}} \tau_j + \Delta\tau_i \tag{4.13a}$$

$$\tau_i > x_i + \Delta\tau_i \tag{4.13b}$$

ensure that the algorithm execution is never started in node i if its current lifetime τ_i is lower than the known minimum network lifetime plus its expected lifetime decrement $\Delta\tau_i$ due to the energy spent by solving the optimization. ■

The nodes which apply the local state update rule 4.3.2 at any given instant of time t_k are chosen by a node selection process $\mathbf{v}(t) : \mathbb{R}^+ \rightarrow X \bigcup \{\varnothing\}$. Two node selection processes that modify the algorithm performance under different sets of fundamental assumptions are studied. These are discussed in the following Subsection.

Algorithm DLMA can thus be summarized as follows.

Algorithm 2 (DLMA for WSNs)**.**

1. *Each node $i \in X$ is initialized with the residual energy of the battery e_i^{res} and state $\boldsymbol{s}_i = \{0\}^L$ (no processing assigned).*
2. *Let $k = 0$ and $t_0 = 0$.*
3. *While (Stop criterion=False) do*
4. *Each node executes Algorithm 1.*
5. *According to node selection process $\mathbf{v}(t_k)$, node i is selected and applies the local state update rule 4.3.2 to its in-neighborhood $\mathcal{N}_{in,i}$.*
6. *Let $k = k + 1$.*
7. *End while.* ■

The *Stop criterion* for Algorithm 2 is discussed in Subsection 4.3.4. Figure 4.2 shows a flux diagram that explains in words the execution of Algorithm 2. The formal proof that at each iteration k of the Algorithm 2, the objective function of Equation 4.2 is either improved or does not change will be given in Theorem 4.3.4, although the optimal solution of Equation 4.2 cannot be guaranteed as eventually found as k increases. However, Algorithm 2 offers several advantages with respect to centralized algorithms.

- Both problems in Equations 4.11 and 4.12 are hard to solve. However, the computational complexity of the local optimization is function only of the number of nodes involved in the optimization, thus despite being a MILP problem, its complexity does not grow by increasing the number of nodes in the network and is small in absolute terms if the number of processing that may be allocated locally between the nodes is small.
- Since the allocation of processing is dynamic, the algorithm reacts to unexpected drops in battery charge and in network topology by changing the status of the nodes involved.

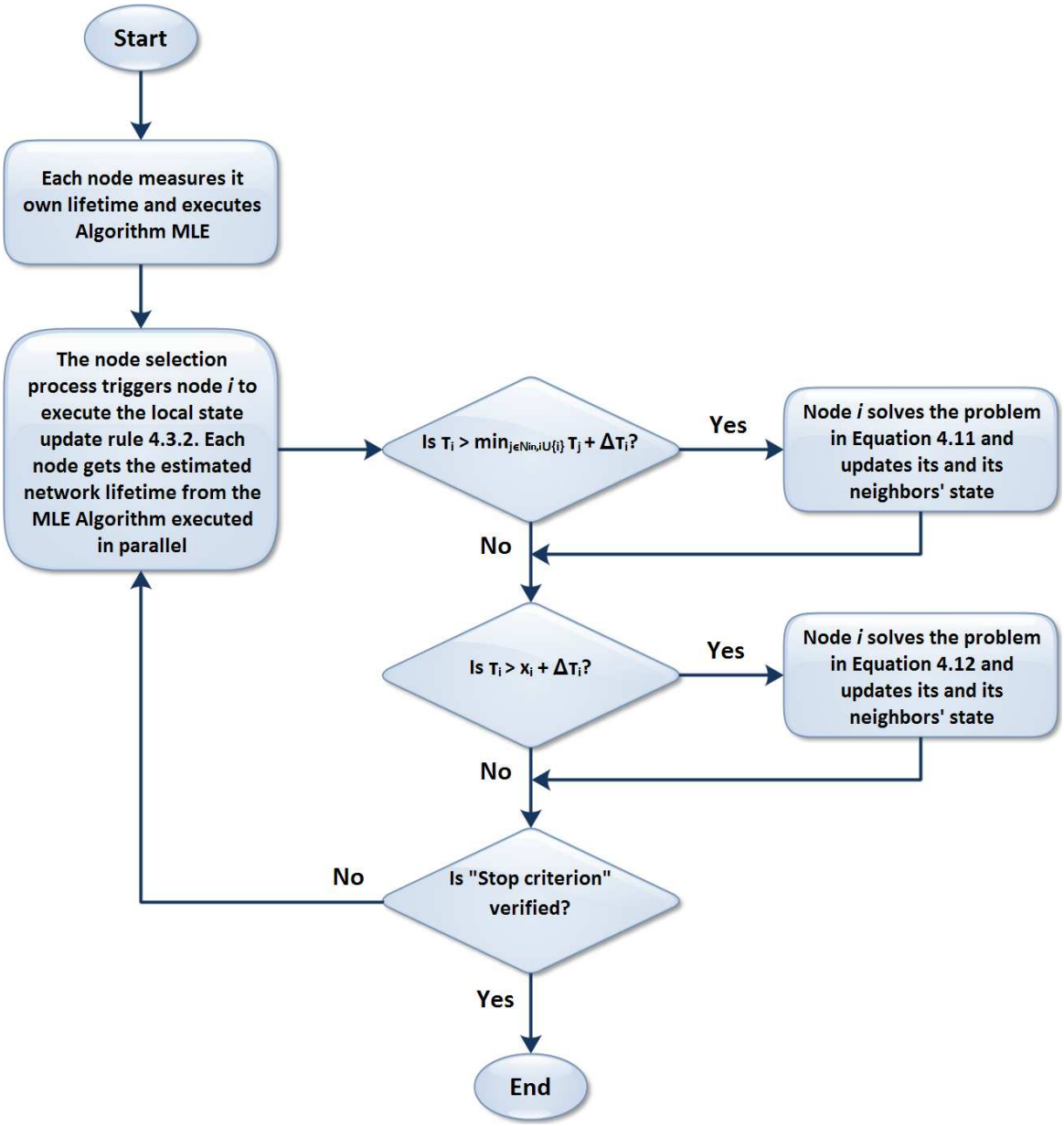


Figure 4.2: Flux diagram of Algorithm 2.

The behavior of the network while DLMA is being executed will be now characterized.

Theorem 4.3.4. *Consider a WSN that executes Algorithm 2. Let* $V(t) = \min_{i \in X} \frac{P_i(\boldsymbol{S})}{e_i^{res}}$ *be the inverse of the minimum lifetime in the network set by the nodes with the smallest ratio between power consumption and energy reserve. Then, if the network executes Algorithm 2*

$$\forall k = 1, \ldots, \infty : \quad V(t_{k+1}) \leq V(t_k).$$

Proof. During the algorithm execution, at each iteration k two situations may occur:

Case 1: The processing state matrix $\boldsymbol{S}$ of the network does not change, then $V(t_{k+1}) = V(t_k)$ by definition.

Case 2: The processing state matrix $\boldsymbol{S}$ changes according to the solution of the local optimiza-

tion problem described by Equations 4.11 and 4.12.

The solutions of problems in Equations 4.11 and 4.12 minimize locally the energy 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 a constant $\Delta\tau_i$. Thus, if a feasible solution is found, the processing state $\boldsymbol{S}$ is updated only if $\min_{j\in\mathcal{N}_{in,i}\bigcup\{i\}} \frac{e_j^{res}}{P_j(s_k : k \in \mathcal{N}_i \bigcup\{i\})} > \min_{j\in\mathcal{N}_{in,i}\bigcup\{i\}} \tau_j(\bar{\boldsymbol{S}})$ causing $V(t_{k+1}) < V(t_k)$, or $\tau_i > x_i + \Delta\tau_i$ causing $\sum_{j\in\mathcal{N}_{in,i}} \boldsymbol{\Theta}_j^{out}$ to be minimized. Thus, since $x_i(t_k) \leq \min_{j\in S_i} \tau_j$, the constraints of the optimization problem ensure that locally the lifetime is not reduced to a value smaller than the minimum lifetime between the nodes $\mathcal{X}_i$.

The nodes not involved in the local optimization will be now proved to not decrease their lifetime as a result. In the proposed local optimization a processing state update is performed only if each single information flow passing from node i to the nodes in $\mathcal{N}_{out,i}$ is not increased due to the constraint $\boldsymbol{\Theta}_i^{out} \leq \bar{\boldsymbol{\Theta}}_i^{out}$. Now the transmission cost for any node in $\mathcal{N}_{in,i}$ can only decrease as $\boldsymbol{\Theta}_i^{out}$ is decreased, as shown in Equation 4.7, and its processing cost cannot increase, as shown in Equation 4.6. Furthermore, also the nodes in the downstream path towards the sink receive a smaller information flow thus consuming less power. Finally, the nodes in the upstream path have their information flow left unchanged, thus $V(t_{k+1}) \leq V(t_k)$ for any $k = 1, \ldots \infty$, proving the statement. □

4.3.3 Node Selection Processes

The two main node selection strategies to apply the local update rule 4.3.2 will be now discussed. The first, called *deterministic node selection process*, is defined by function $\mathbf{v}(t) : \mathbb{R}^+ \rightarrow X \bigcup\{\varnothing\}$ that at each step k selects the out-neighbors of the nodes chosen at step $k-1$. Furthermore the sink node executes the local state update periodically $\mathbf{v}(t_{0+kT}) = 0$, for $k = 1, \ldots \infty$. In other words, with period T, the sink node $i = 1$ triggers its neighbors to perform a local optimization, thus triggering an avalanche mechanism that triggers all the nodes starting from the sink toward the leaf nodes. This mechanism is controlled by the sink node since the period T in which it is activated controls the frequency of optimizations that occur in the network. Since the graph is rooted, connected, directed and acyclic, it is ensured that all nodes are triggered in a not self-sustaining way so that after the leaf nodes are triggered, no more local optimizations are performed until the sink node triggers a new round.

The second, called *stochastic node selection process*, is defined by a random stochastic process $\mathbf{v}(t) : \mathbb{R}^+ \rightarrow X \bigcup\{\varnothing\}$, 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 sink node and ensures that local optimizations are triggered evenly throughout the network.

The advantages of the stochastic node selection process are that it is completely decentralized and due to the randomization process may get closer to the optimal processing assignment. Furthermore, the frequency of each local optimization may vary between the nodes, thus increasing for nodes with large lifetime with respect to the others and minimizing for those nodes close to the minimum lifetime in the network.

4.3.4 Stop Criterion

In this Section, the stop criterion for Algorithm 2 will be discussed.

Definition 4.3.5 (Stop Criterion)**.** *Let a sensor network execute Algorithm 2 and Algorithm 1. Let D be the length of the longest directed path in $\mathcal{G}_X$ and let T be the maximum time it takes for a node to complete a reception and transmission cycle. Assume that a deterministic node selection process is adopted so that at most every DT units of time every node has executed the local update according to Algorithm 2. Then, for each node i selected to perform Algorithm 1, 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 2 at node i.* ■

In the following proposition, the proof that when the stop criterion in Definition 4.3.5 is satisfied any further execution of Algorithm 2 cannot improve the network lifetime will be given.

Proposition 4.3.6. *Consider a sensor network that executes Algorithm 2 and Algorithm 1 with a deterministic node selection process. Assume that each node verifies the conditions for the stop criterion given in Definition 4.3.5. Then, any further execution of Algorithm 2 cannot improve the minimum network lifetime τ.*

Proof. Assume the conditions for the stop criterion in Definition 4.3.5 are verified and the network is executing a deterministic node selection process. Assume that at time t' a local update on node i occurs. The longest time it takes for such an optimization to have an effect, if any, on the minimum network lifetime, will be now computed. When node i updates its state, four cases may occur:

1. Node i does not improve its own lifetime nor reduces its output bitrate. In this case nothing happens.
2. Node i has the shortest lifetime in the network and improves it. In this case, after at most TD units of time all nodes that may affect its lifetime are aware

of the change due to Proposition 4.3.1.

3. Node i does not improve its own lifetime but reduces its output bitrate. In this case, if the minimum network lifetime has been improved and after at most TD units of time all nodes are aware of the change due to Proposition 4.3.1.
4. Node i does not have the shortest lifetime in the network but it improves its own. In this case, only the neighbors of node i may detect the change.

Thus, the worst case scenario that may take the longest time for a given node to detect a change in the minimum network lifetime is set by case (4) when a node changes its state but only its neighbors are ensured to detect it. Clearly, if after this update in the next TD units of time the neighbors do not make changes to their state, the network lifetime will not change and no further local state update will be performed due to the state change of node i. Therefore, the longest time it takes for node j that holds the shortest lifetime to be affected by a local state update due to case (4) corresponds to the time it takes for all the nodes in the path connecting it with node i to perform a local update and change their state according to case (4). Thus, in the worst case in which the path from node i to node j is equal to D, it takes at most $\frac{D(D+1)}{2}T$ units of time to ensure the propagation of the effect of the state change of node i to node j. Whenever the minimum network lifetime is improved, due to Proposition 4.3.1 after at most TD 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 $\frac{D(D+1)}{2}T + TD$ 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 2 may change the state of the network and consequently cannot improve the current minimum network lifetime. □

4.3.5 Computational Complexity and Energy Cost of the Optimization

The optimization problem defined in 4.3.2 is a MILP. It is known that to solve a MILP problem a number of operations that grow exponentially with respect to the number of variables is required in the worst case. Thus, the feasibility of solving problem 4.3.2 in an embedded system such as a sensor node, depends on the number of available processing tasks that can be activated in it and in its neighbors. 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 4.3.2 for a given assignment of processing tasks and verify that constraints of 4.3.2 are satisfied. Now, recalling that the columns of matrix $\boldsymbol{D} = [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n]$ are binary vectors whose elements are 1 if the corresponding processing may be activated in

the corresponding nodes, the number of possible processing assignments in the generic in-neighborhood $\mathcal{N}_{in,i}$ of node i, when it performs the local update 4.3.2, is

$$N_{i,assignments} = 2^{\mathbf{1}^T \sum_{j \in \mathcal{N}_{in,i} \cup \{i\}} \mathbf{d}_j}.$$

Thus, in the worst case, the energy cost to perform the local update 4.3.2 on node i is

$$\hat{E}_i = E^o \times N_{i,assignments} = E^o \times 2^{\mathbf{1}^T \sum_{j \in \mathcal{N}_{in,i} \cup \{i\}} \mathbf{d}_j}. \tag{4.14}$$

Clearly, while sensing, processing tasks and transmission occur continuously in the network, problem 4.3.2 is solved periodically whenever the local update is triggered. Thus, depending on the total cost $\hat{E}_i$, there is a tradeoff between how often the update is triggered and the increased consumption of energy due to its execution.

4.4 Performance Analysis

In this Section, the analysis for two completely different scenarios will be presented:

- **Scenario A**: a rectangular-shaped outdoor environment (e.g., a vineyard, a seaport, a tourist plaza), where nodes have been positioned randomly following a uniform distribution. Each node is equipped with sensors for the measurement of temperature, humidity, PH and light exposure. Data is then sent to the sink. The application assigned to this scenario consists in collecting the sensed data every 10 minutes, compute the average value for the entire environment, and compute the average value over an hour. Each sensed value is represented as a double numerical value, which is 64-bit long.

- **Scenario B**: an urban environment, where nodes have been positioned along the streets as shown in Figure 4.3. All solid markers represent nodes equipped with sensors for speed measurement of the vehicles passing through; speed measurements are sent every 2 minutes, and are represented as double numerical values (64-bit long). Empty markers are more capable nodes which do not perform any sensing task, but can perform more complex processing operations. The assigned application consists in a driver, placed in the Start point, which would like to know which is the fastest way to reach Destination, based on the speed information collected by sensor nodes.

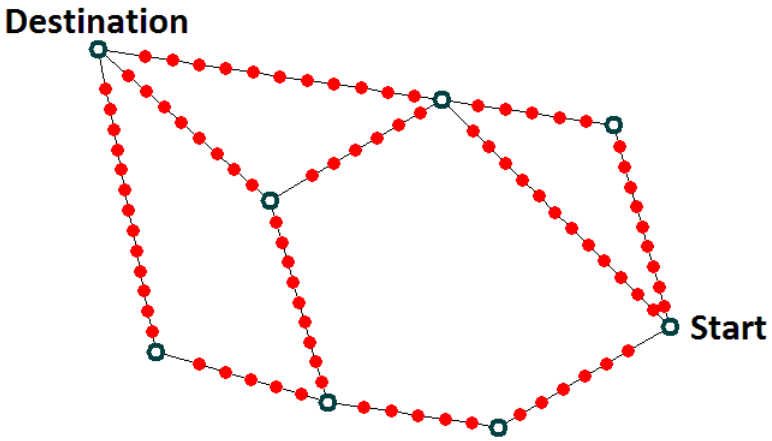


Figure 4.3: Example of topology for Scenario B. Solid markers represent nodes equipped with speed sensors, while empty markers are more capable nodes which do not perform any sensing task, but can perform more complex processing tasks

Two WSN settings are considered according to some of the most significant realistic scenarios considered in past works, such as in [86]:

- **UC-UE**: uniform energy consumption and uniform initial energy at each node;
- **NUC-NUE**: non uniform energy consumption and non uniform initial energy at each node (energy consumption parameters selected randomly in the range 60% to 140% of the values for case UC-UE; battery charge assigned randomly from 20% to 100% of the total charge).

In both scenarios, nodes communicate using IEEE 802.15.4 radio interfaces on the 2.4GHz ISM frequency band. To keep things simple, any possible overhead has not been taken into account. The resulting scenarios have been simulated in a MatLab environment, where the proposed algorithm was implemented along with three alternative approaches:

- mechanism S: data processed only by the sink so that sensed data is sent to the sink without any intermediate processing;
- mechanism CH: data processed by every cluster head found in the path towards the sink. Cluster heads are those nodes receiving flows from more than one node;
- mechanism CO: centralized optimization algorithm described in Chapter 3.

In the following Subsections, the comparisons between results obtained using the DLMA and those obtained with mechanisms S, CH and CO in terms of lifetime conservation gain will be presented. These results are referred to respectively as: DLMA-S, DLMA-CH and DLMA-CO. The main setup parameters are listed in Table 3.3.

4.4.1 Scenario A

The application assigned to Scenario A can be subdivided in 1 sensing task t_1^s - the *temperature, humidity, PH and light exposure sensing* - and 2 processing tasks: t_1^p is the *temporal mean computation*, while t_2^p is the *spatial mean computation*. The described tasks are listed in Table 4.1, and their corresponding graph is shown in Figure 4.4.

Table 4.2 and Figure 4.5 show results for this scenario, considering both deterministic and stochastic node selection (see Section 4.3.3). The mean and deviation values are computed on results found for networks with different node densities, different node deployments, different energy parameters for case NUC-NUE, and different node sequences for stochastic node selection.

Results show an average improvement of 80.2% obtained with the proposed strategy with respect to S and CH in terms of network lifetime. Differently, as it was clearly expected, a lifetime decrement is observed when making a comparison with the mechanism CO with an average decrement of 24.4%. As a matter of fact, this is the drawback of the DLMA with respect to the centralized optimization algorithm, which leads to an optimal task allocation.

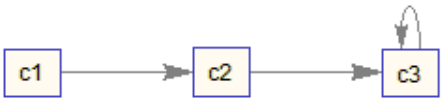


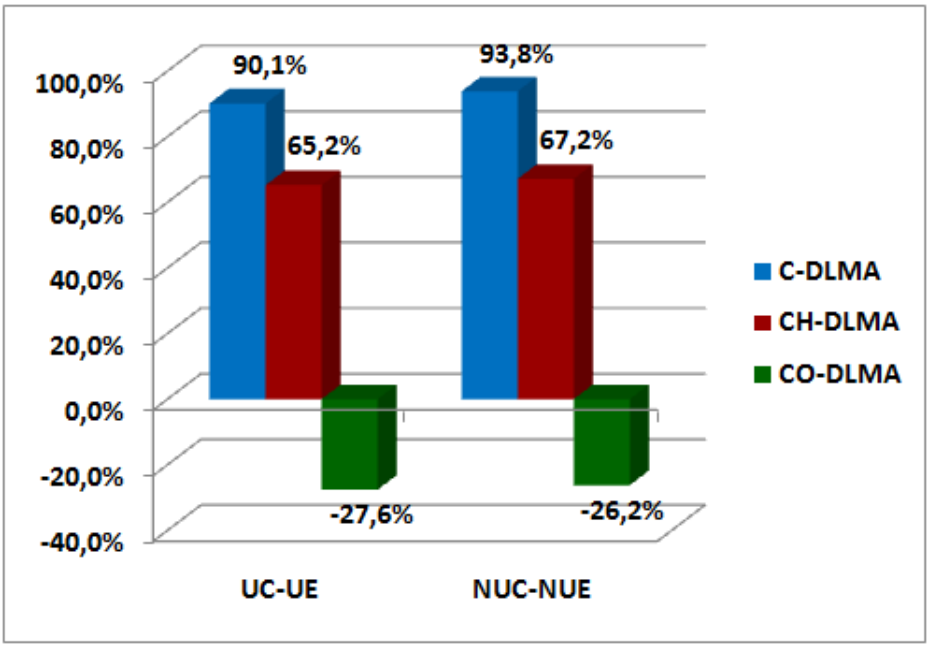
Figure 4.4: DG for the tasks of Scenario A

Table 4.1: Tasks for Scenario A

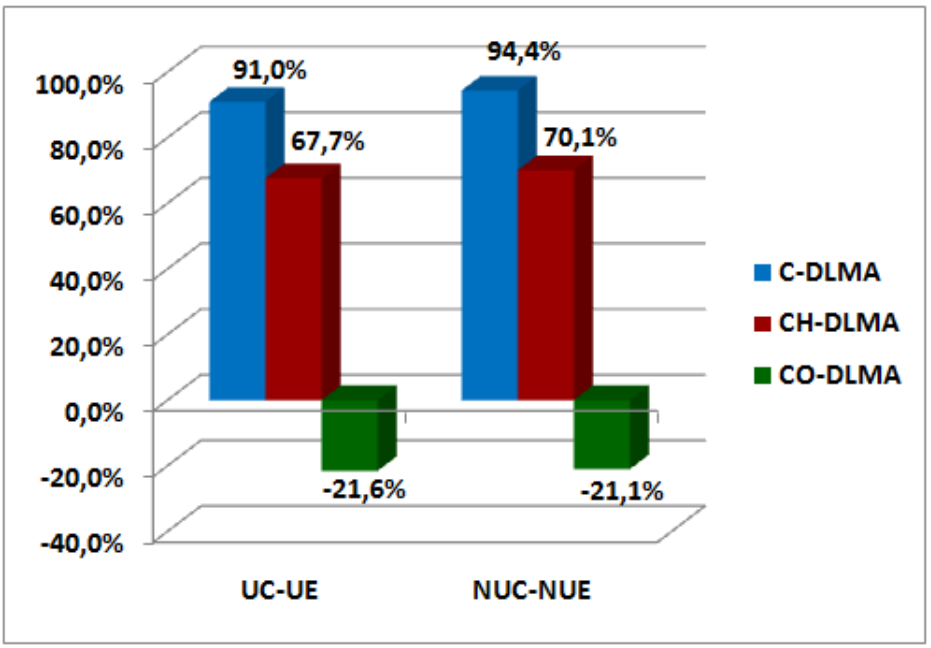
Task	**Description**
t_1^s	Temperature, humidity, PH and light exposure sensing
t_1^p	Temporal mean
t_2^p	Spatial mean

Table 4.2: Percentage values of lifetime conservation deviation for Scenario A

	Deterministic node selection		
	DLMA-S	**DLMA-CH**	**DLMA-CO**
UC-UE	1.2%	6.0%	15.5%
NUC-NUE	2.0%	9.4%	16.8%
	Stochastic node selection		
	DLMA-S	**DLMA-CH**	**DLMA-CO**
UC-UE	2.9%	13.4%	20.4%
NUC-NUE	2.9%	13.3%	23.2%



(a) Deterministic node selection



(b) Stochastic node selection

Figure 4.5: Percentage values of mean lifetime conservation using DLMA in Scenario A, considering deterministic (a) and stochastic (b) node selection

The best results are obtained for comparisons S and CH. This is because such mechanisms assign processing tasks in a fixed way, without taking into account how different allocations could affect the network lifetime. On the other hand, tasks are assigned by DLMA to those nodes that actually weight less on the network, in terms of processing and transmission costs. In fact, network lifetime is improved when tasks are assigned to nodes having lower processing cost parameters or higher residual battery charge, or when processing data (and therefore, in most

cases, reducing their amount), before sending them, considerably reduces transmission cost. This behavior is slightly stressed for heterogeneous networks, which are the most common in real scenarios. For this reason, results for NUC-NUC case are slightly better than UC-UE case.

It can be noted that deviation values are higher for NUC-NUE case with respect to UC-UE case because of the randomness intrinsic to the NUC-NUE case.

With respect to node selection, results are quite similar, but, as expected, stochastic node selection slightly outperforms the deterministic one. Nevertheless, once again the randomness, introduced by the stochastic method, leads to higher deviation values than the deterministic node selection.

4.4.2 Scenario B

The application assigned to Scenario B can be subdivided into 11 sensing tasks and 37 processing tasks, as defined in Table 4.3. 11 different *speed sensing* tasks are distinguished, one for each stretch of street. 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 the mean traveling time for each stretch is required in order to confront them to each other, 11 *mean speed computing* processing tasks and 11 *mean traveling time computing* processing tasks are defined, one for each stretch of street. To find the best path, i.e. the best combination of stretches of street that lead from the Start point to Destination, the sum of the mean traveling times of all the combination of stretches that can be driven one after the other are needed in order to confront them. Therefore, the remaining processing tasks are: 12 different *summation of mean traveling times for different stretches*, and 3 different *choice of the best path*. The relations among tasks are depicted in Figure 4.6. With reference to Figure 4.3, 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. Empty markers 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 processing tasks for the appropriate stretches.

Table 4.3: Tasks for Scenario B

Task	**Description**
$t_1^s \div t_{ll}^s$	Speed sensing
$t_1^p \div t_{ll}^p$	Mean speed computing
$t_{l2}^p \div t_{22}^p$	Mean travelling time computing
$t_{23}^p \div t_{34}^p$	Summation of mean travelling times for different stretches
$t_{35}^p \div t_{37}^p$	Choice of the best path

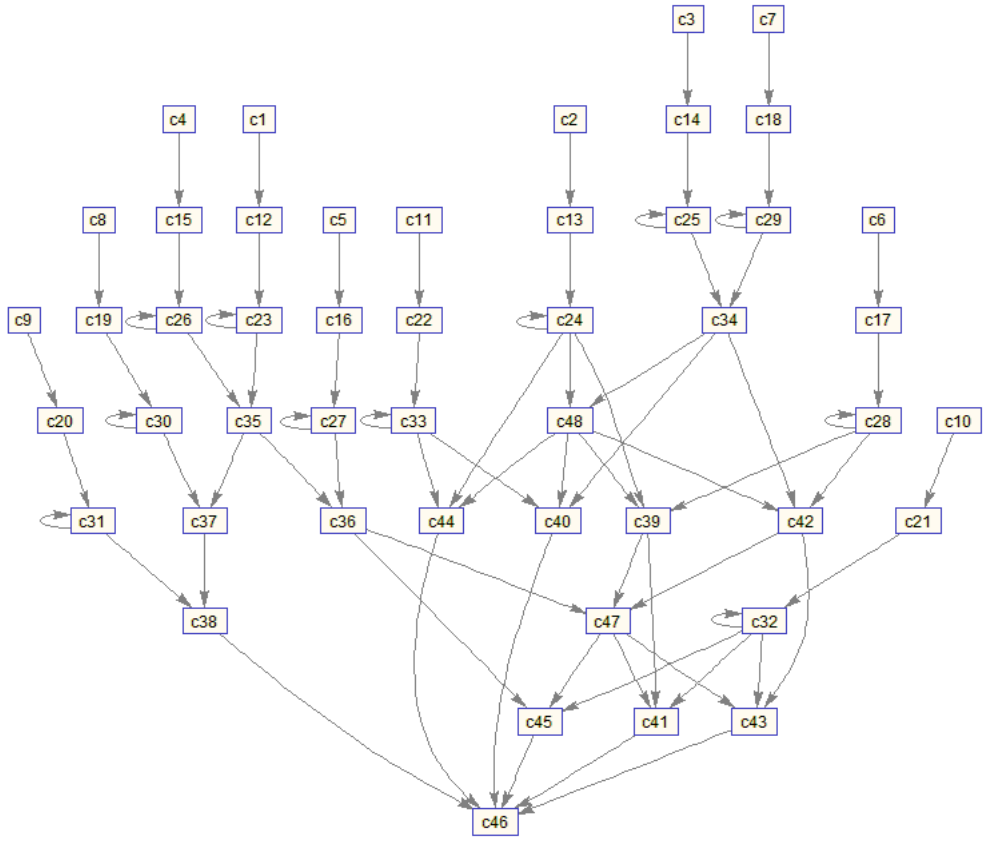


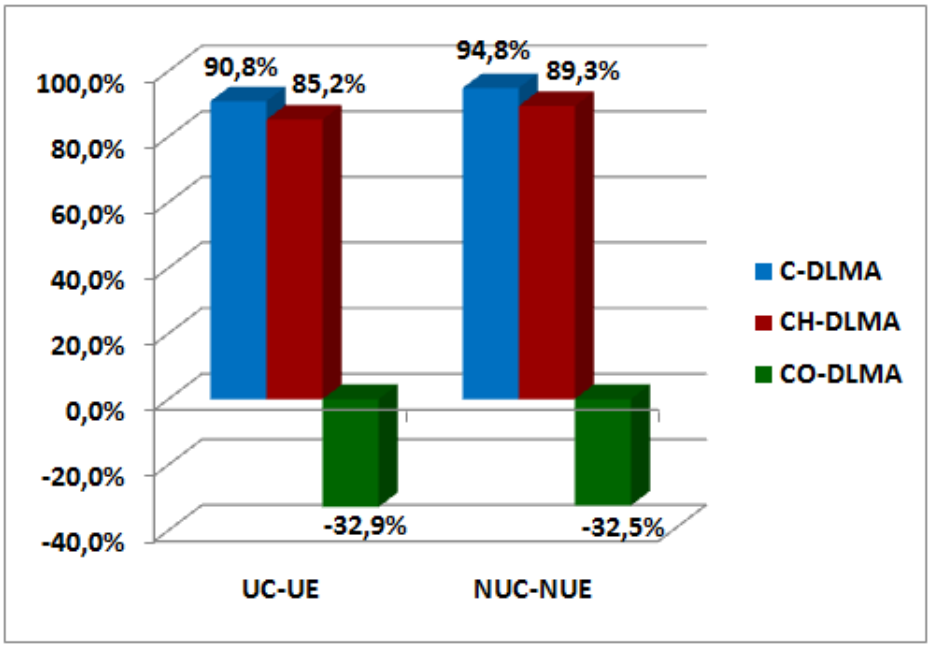
Figure 4.6: DG for the tasks of Scenario B

Results are shown in Table 4.4 and Figure 4.7. The mean and deviation values are computed on results found for networks with different node densities, different energy parameters for case NUC-NUE, and different node sequences for stochastic node selection.

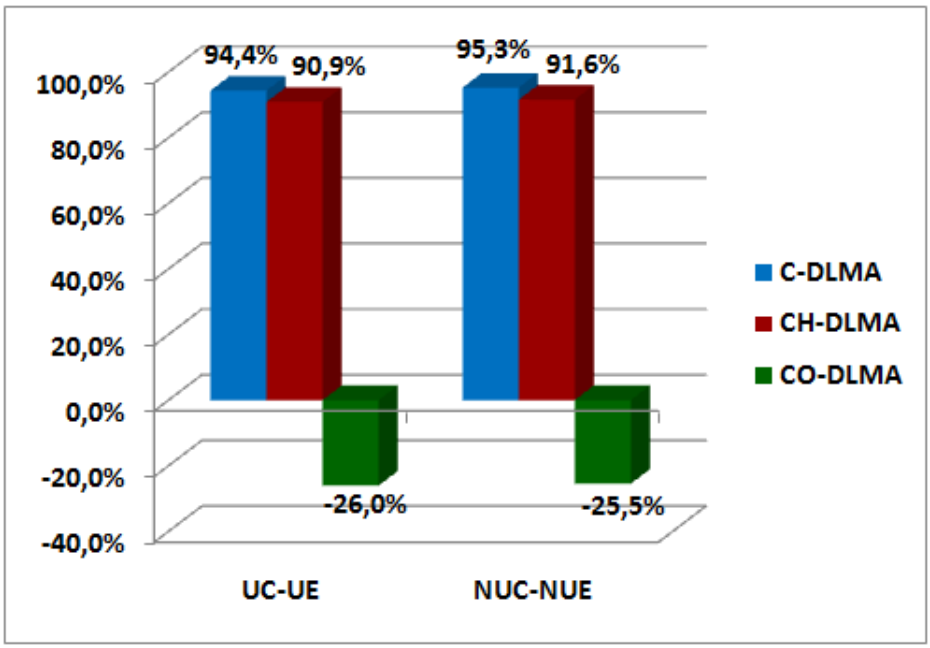
The average improvements for comparisons DLMA-S and DLMA-CH is 91.5%, while the average decrement for comparison DLMA-CO is 29.2%. With respect to the previous scenario, a significant improvement of the proposed algorithm when compared with the the CH mechanism is observed.

Table 4.4: Percentage values of lifetime conservation deviation for Scenario B

	Deterministic node selection		
	DLMA-S	**DLMA-CH**	**DLMA-CO**
UC-UE	0.7%	1.3%	14.3%
NUC-NUE	1.2%	1.8%	11.1%
	Stochastic node selection		
	DLMA-S	**DLMA-CH**	**DLMA-CO**
UC-UE	1.2%	2.3%	15.3%
NUC-NUE	2.4%	3.5%	17.7%



(a) Deterministic node selection



(b) Stochastic node selection

Figure 4.7: Percentage values of mean lifetime conservation using DLMA in Scenario B, considering deterministic (a) and stochastic (b) node selection

This is because in Scenario A nodes are placed randomly, and thus it is more likely that they form clusters that are quite close to each other. In Scenario B, the only nodes that can be considered cluster heads are those with greater computational power (empty markers in Figure 4.3). They are far from each other, and since the burden of processing is all upon them, their lifetime decreases quite fast.

Also in this scenario, the node selection procedure doesn't heavily influence the final per-

formance. However, again, the stochastic node selection slightly outperforms the deterministic one.

4.4.3 Considerations about MLE and DLMA Algorithms

In this Section, some additional considerations about MLE and DLMA algorithms behavior for both Scenarios A and B are drawn.

Table 4.5 shows the average number of steps for DLMA to come to a solution, which is the mean number of times each node has to perform DLMA. As foreseen, the deterministic node selection leads to a result in a lower number of steps than the stochastic node selection. Furthermore, although the application for Scenario B is much more complex than the application for Scenario A, the average numbers of steps are quite close. Of course, since the optimization complexity depends on the number of variables, and therefore it depends on the number of nodes involved in the optimization and, most importantly, on the number of tasks that those node are able to perform, each step is more complex for Scenario B than for Scenario A.

This is demonstrated by results in Table 4.6, where the average times t_{MLE} and t_{DLMA} to perform algorithms MLE and DLMA, and their summation t_{TOT}, are compared to the related average time t_{CO} to perform centralized algorithm CO. These times are computed taking into account not only execution time, but also signalling transmission and reception time due to the algorithms. As expected, execution time to perform DLMA is, on average, 95% higher for Scenario B than for Scenario A. Nevertheless, the total amount of time for the algorithms to converge is still low, especially if compared with centralized algorithm CO: since the number of nodes involved in each optimization is much lower in DLMA than in CO, and the number of tasks that can be performed by adjacent nodes is low, CO execution weights much more than DLMA on the network, particularly for complex application scenarios.

Table 4.5: Average number of steps for DLMA

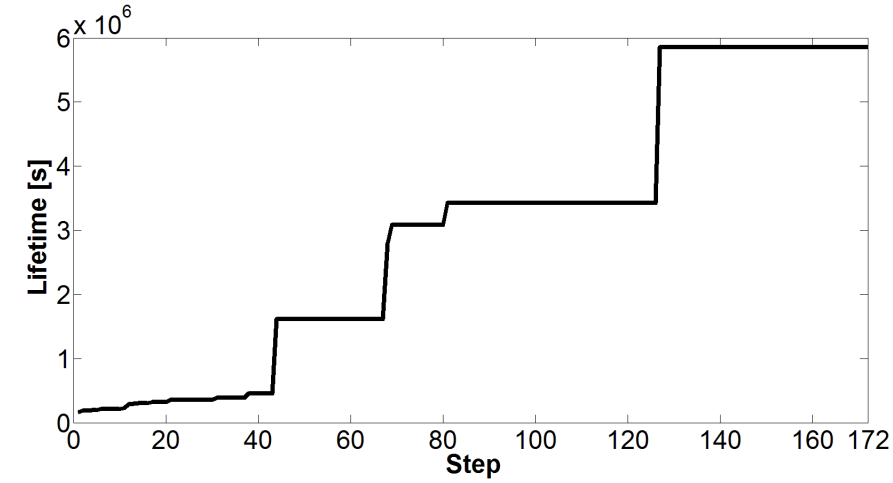
	Scenario A	
	Deterministic selection	**Stochastic selection**
UC-UE	3.6	4.5
NUC-NUE	3.5	4.6
	Scenario B	
	Deterministic selection	**Stochastic selection**
UC-UE	3.8	4.1
NUC-NUE	3.7	4.3

Table 4.6: Average time (in seconds) to perform MLE and DLMA

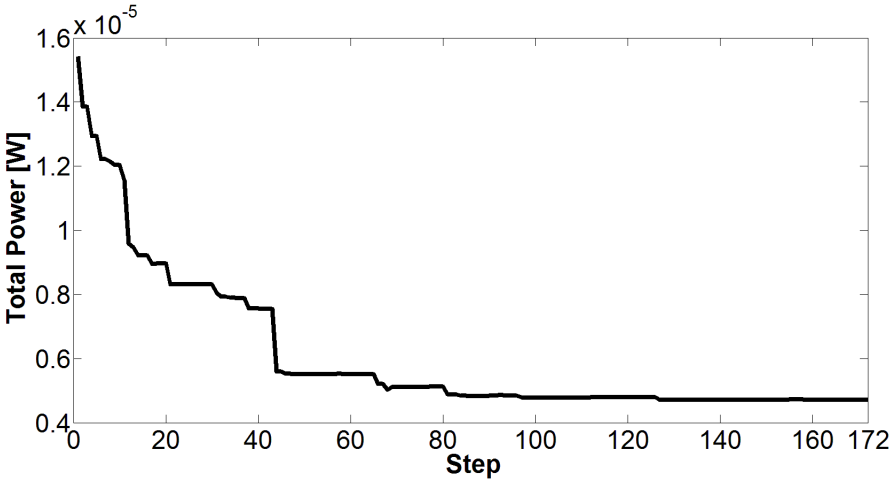
		Scenario A	
		Deterministic selection	**Stochastic selection**
UC-UE	t_{MLE}[s]	0.0017	0.0017
	t_{DLMA}[s]	0.020	0.043
	t_{TOT}[s]	0.022	0.045
	t_{CO}[s]	0.089	
NUC-NUE	t_{MLE}[s]	0.0017	0.0017
	t_{DLMA}[s]	0.019	0.044
	t_{TOT}[s]	0.021	0.046
	t_{CO}[s]	0.089	
		Scenario B	
		Deterministic selection	**Stochastic selection**
UC-UE	t_{MLE}[s]	0.0017	0.0017
	t_{DLMA}[s]	0.40	0.79
	t_{TOT}[s]	0.40	0.80
	t_{CO}[s]	397	
NUC-NUE	t_{MLE}[s]	0.0017	0.0017
	t_{DLMA}[s]	0.36	0.80
	t_{TOT}[s]	0.37	0.81
	t_{CO}[s]	433	

It needs to be noticed that conditions of Equations 4.13a and 4.13b ensure that the local optimization algorithm is never started if the expected lifetime decrement due to its execution is higher than the current node's lifetime. This means that the local optimization does not run for high numbers of nodes involved and tasks that can be performed. For this reason, it is unlikely that a local optimization takes a long time to come to a solution, and therefore it is unlikely that DLMA takes a long time to converge.

At the same time, conditions of Equations 4.13a and 4.13b guarantee that, at each DLMA step, the network lifetime cannot decrease with a change in status. This monotonic increase is attested by Figure 4.8(a), where an example of the network lifetime improvement for each DLMA step is shown. Each step corresponds to the local optimization computed on one node according to the local state update rule 4.3.2: at each status change, the expected network lifetime changes accordingly. The same reference DLMA execution is used to compute the expected total power consumption changes shown in Figure 4.8(b): although this curve is not monotonic, the expected total power consumption decreases while DLMA converges.



(a) Network lifetime



(b) Total Power Consumption

Figure 4.8: Example of network lifetime and total power consumption improvement during DLMA execution

Chapter 5

Task Allocation Negotiation Algorithm

Due to the drawbacks of a centralized solution such as the one described in Chapter 3, a distributed algorithm, the DLMA, has been proposed in Chapter 4 to perform task allocation in WSNs. DLMA reduces the problem complexity, as only local areas are considered rather than the whole network. The communication overhead caused by packet exchanges between network nodes and the sink is also avoided. Nevertheless, it does not take into account the execution time of the application assigned to the network. This might lead to an inconvenient task assignment, where the application deadline is reached before the application is executed.

In this Chapter, a new distributed algorithm for the allocation of application tasks among WSN nodes is proposed. This algorithm, named *Task Allocation Negotiation* (TAN) [87], aims not only to improve the existing algorithms performance reducing computational complexity, but also to reduce both network energy consumption and application execution time. The major contribution is the adoption of the rules of non-cooperative game theory [88]. Sensor nodes negotiate among each other while setting the application configuration. In doing so, each node aims at maximizing a *node utility function* under the constraints set by neighboring nodes. The scenario under consideration is then proved to be a potential game, which means that every improvement of the node utility functions corresponds to the same improvement in the utility perceived by the whole network, which implies that the problem has a unique outcome that is reachable in a finite time.

This Chapter is organized as follows. Section 5.1 introduces the problem and the adopted approach. Section 5.2 describes the task assignment model. Section 5.3.2 presents the TAN algorithm. Section 5.4 shows and discuss simulation results.

5.1 Problem Formulation

The reference scenario considered in this paper is that of a hierarchical heterogeneous WSN. This WSN needs to run a required application, which can be divided into smaller tasks that can be assigned to network nodes to be executed. Due to the heterogeneous node characteristics, some nodes can perform the same task faster than others, and even spend less energy. The objective of the proposed algorithm, TAN, is to assign the tasks to suitable nodes, such that the processing time of the application is as short as possible and the network lifetime is as long as possible. TAN includes negotiations among nodes to determine task assignments. Whenever a node receives some data, along with the information on which particular tasks have to be performed on the data, it 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.

5.1.1 Network Model

The reference network model has already been described in Section 2.2. The scenario under consideration is that of a hierarchical heterogeneous WSN. The network is organized into clusters of sensor nodes. Each cluster is controlled by a single cluster head, which collects sensory data from nodes in its cluster. There is a single hop between a sensor node and its corresponding cluster head. At the top of the hierarchy a a sink node, which collects data received from the cluster heads. Sensors transmit their data to the cluster heads, which then deliver the collected data to the gateway through a path of cluster heads. Routing paths over the cluster heads is determined by conventional routing protocols [89], such as Self-Organizing Protocol, Location-Based Routing Protocol, etc. In this model, Hierarchical Power-Aware Routing protocol is used [90], due to the fact that it is based on a trade-off between minimizing power consumption and maximizing the minimal residual power of the network.

Negotiations are performed among neighbor nodes; multihop negotiations do not take place, and this significantly reduces the number of message exchanges. No communication is allowed between sensors belonging to different clusters. This means that, given two nodes i and j, $e_{ij}^X \in E_X$ if and only if i and j are in the same cluster. Such an architecture helps reduce the overhead caused by the negotiations. Figure 5.1 shows the reference architecture of the network under examination, where SN stands for sensor node, CH stands for cluster head, and GW is the gateway.

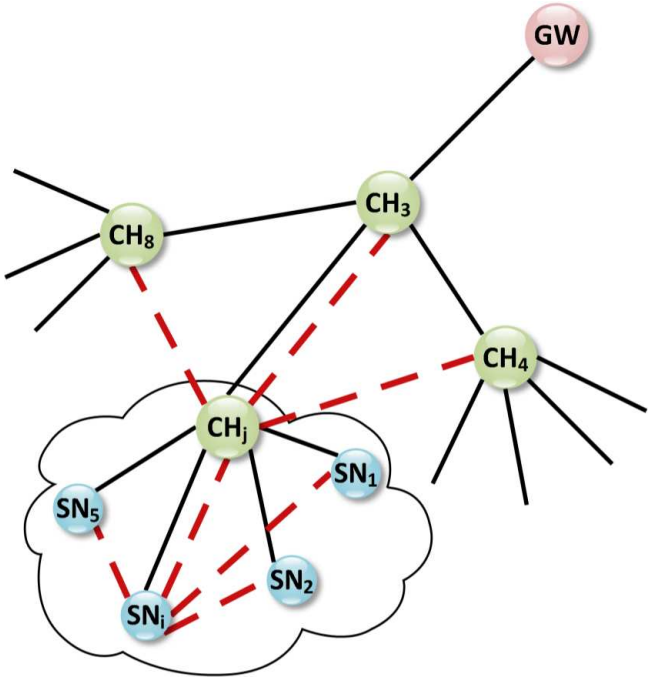


Figure 5.1: Architecture of the network. Solid lines represent connections formed by the routing protocol; dashed lines represent connections between negotiation nodes

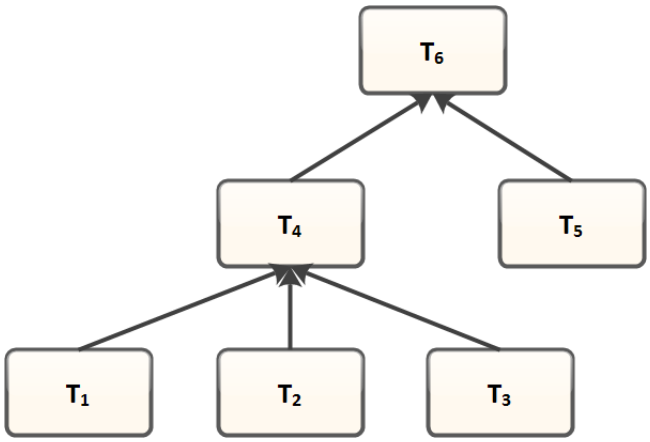


Figure 5.2: Example of a task DAG.

5.1.2 Task Model

A single large application is considered to be assigned to the network for execution. The application is decomposed into a set of tasks, and can be described as a DAG of tasks $\mathcal{G}_T = (T, E_T)$, where $T = \{1, \ldots, \lambda, \ldots \Lambda\}$ is the set of tasks, and $E_T = (e_{vw}^T)$ is the set of edges, with each edge e_{vw}^T representing a unidirectional data transfer from task v to task w. An example of the task DAG is depicted in Figure 5.2. The sink initially sends the application graph $\mathcal{G}_T$ to nodes, so that each node learns the relations and dependencies between the tasks.

A binary vector $\boldsymbol{s}(i) = (s(i, \lambda))$, for $\lambda \in T$, can be assigned to each node i in the network

where $s(i,\lambda)$ is a boolean value representing the current state of node i corresponding to task λ, i.e. $s(i,\lambda) = 1$ when node i is performing task λ. The vector $\boldsymbol{s}(i)$ is called the *task assignment strategy* of node i. The state $s(i,\lambda)$ can only be set to 1 if and only if all its predecessor tasks have been assigned to a node, i.e. $s(i,\lambda) = 1$ if $s(j,l) = 1,\ \forall l \in T_{in}(\lambda)$, where $T_{in}(\lambda)$ is the set of input tasks for task λ. With reference to Figure 5.2, $T_{in}(T_1) = T_{in}(T_2) = T_{in}(T_3) = T_{in}(T_5) = \{\}$, hence they are *source tasks*, $T_{in}(T_4) = \{T_1, T_2, T_3\}$, and $T_{in}(T_6) = \{T_4, T_5\}$.

Since not every node may be able to perform each and every task, a binary vector $\boldsymbol{d}(i) = (d(i,\lambda))$ is defined, where $d(i,\lambda) = 1$ if node i is able to perform task λ. Note that $s(i,\lambda) = 1$ is possible only if $d(i,\lambda) = 1$, which means that $d(i,\lambda) \geq s(i,\lambda)$.

During the execution of the proposed algorithm, two sets of tasks are formed: the set of tasks $T_{prev} = \{1,\ldots,h,\ldots,H\}$ that have been already assigned to nodes and the set $T_{next} = \{1,\ldots,k,\ldots,K\}$ of those tasks that are yet to be assigned. This means that, for each task $h \in T_{prev}$, there is a node i for which its state $s(i,h)$ is equal to 1. Source tasks are assumed to be already assigned to the nodes by the sink according to the output required by the application. For example, if the application requires the temperature measurement for a certain area, source tasks correspond to the temperature sensing tasks for that area. In this case, the sink assigns these source tasks to the sensor nodes that are located in that area. Hence, initially, the set T_{prev} is only populated with source tasks, i.e. with reference to Figure 5.2, $T_{prev} = \{T_1, T_2, T_3, T_5\}$ and $T_{next} = \{T_4, T_6\}$.

5.2 The Task Assignment Model

Finding the trade-off which best fits the requirements of both minimizing the application completion time and maximizing the network lifetime could be exceedingly time and energy consuming since this is an NP-hard problem [67],[68]. For this reason, in this Section, a non-cooperative game model [91] is used for the task allocation problem in WSNs. Neighboring nodes negotiate in order for each node i to choose a task assignment strategy $\boldsymbol{s}(i)$ that maximizes its own utility function.

5.2.1 Definitions

A non-cooperative game is defined by the tuple $\Gamma = \langle X, \{\boldsymbol{s}(i), u_i\}_{i\in X}\rangle$, where a utility function u_i is assigned to node $i \in X$ for the given strategy vector $\boldsymbol{s}(i)$. The goal of each node is to maximize its own utility in a rational way. Therefore, a strategy $\boldsymbol{s}^*(i)$ is preferred to a strategy $\boldsymbol{s}(i)$ if and only if $u_i(\boldsymbol{s}^*(i)) > u_i(\boldsymbol{s}(i))$. For simplicity, $\boldsymbol{S} = \bigcup_{i\in X} \boldsymbol{s}(i)$ is denoted as the strategy of all the nodes in the network.

5.2.2 Task Utility Function

The TAN algorithm decides which particular node should execute a given task k by maximizing a *task utility function* over the set T_{next} of unassigned nodes, given by

$$u_k(\boldsymbol{S}) = \max_{i \in X} \left\{ \left[\Omega_t(i,k) + \frac{\alpha}{NF(k)} \times \Omega_\tau(i,k,\boldsymbol{S}) \right] \times s(i,k) \right\} \tag{5.1}$$

where $\Omega_t(i,k)$ is the *task completion time component* of task k when it is performed by node i, $\Omega_\tau(i,k,\boldsymbol{S})$ is the *network lifetime component* when task k is performed by node i according to the strategy $\boldsymbol{S}$, $NF(k)$ is a normalization factor that eliminates the difference in magnitude between the task completion time and the network lifetime values, and $\alpha > 0$ is a weighting factor.

The maximum operation ensures that the task can only be assigned to the node that maximizes the utility function $u_k(\boldsymbol{S})$: since the goal of the game is to maximize the utility function only the node that ensures the best outcome will be chosen to perform task k.

Task Completion Time Component

As defined in Section 5.1, the set T_{next} is formed of those tasks that are to be assigned to nodes for execution. Each task $k \in T_{next}$ is characterized by two parameters:

1. the deadline for successfully completing a task, $t_d(k)$;
2. the number of required instructions, $I(k)$.

The task completion time component is expressed as

$$\Omega_t(i,k) = \frac{t_d(k) - t_c(i,k)}{t_d(k)} \tag{5.2}$$

where $t_c(i,k)$ is the completion time if task k is performed by node i, which is defined as

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

with $t_{instr}(i)$ time needed by node i to perform a single instruction.

Network Lifetime Component

This component is defined as follows

$$\Omega_\tau(i,k,\boldsymbol{S}) = F_p(i,k) + F_{tx}(i,k,\boldsymbol{S}) \tag{5.3}$$

where $F_p(i,k)$ is the component related to the change in network lifetime due to the processing cost needed to perform task k in node i. This component is defined as

$$F_p(i,k) = -I(k) \times \frac{e_i^{ins}}{e_i^{res}} \tag{5.4}$$

Note that, since data processing entails a certain amount of energy consumption, $F_p(i,k)$ cannot increase task k's utility, and thus it needs to be negative in order to decrement the utility of node i when task k is executed at node i.

The term $F_{tx}(i,k,\boldsymbol{S})$ in Equation 5.3 is related to the change in network lifetime due to the transmission (and reception) of the necessary data for task k to node i, i.e. it represents communication costs in the task utility function. Let $\boldsymbol{p}_{i\to j} = \{(i,a),(a,b),\ldots,(e,f),(f,j)\}$ be the routing path that connects node i to node j, and HL be a hierarchically higher-level node for nodes i and j. The transmission term is then

$$F_{tx}(i,k,\boldsymbol{S}) = -C_{tx}(\boldsymbol{p}_{i\to HL},k) + \sum_{l\in T_{in}(k)} \sum_{j\in X} \Big\{ C_{tx}(\boldsymbol{p}_{j\to HL},l) - C_{tx}(\boldsymbol{p}_{j\to i},l) \Big\} \times s(j,l) \tag{5.5}$$

where

$$C_{tx}(\boldsymbol{p}_{i\to j},k) = \sum_{(x,y)\in \boldsymbol{p}_{i\to j}} \left(\frac{e_y^{rx}}{e_y^{res}} + \frac{e_{xy}^{tx}}{e_x^{res}} \right) \times n(k) \tag{5.6}$$

In Equation 5.6, $C_{tx}(\boldsymbol{p}_{i\to j},k)$ is the cost to transmit (and receive) data from node i to node j, and $n(k)$ is the number of output bits for task k. The difference operation is due to the fact that the difference in lifetime between two cases is considered: in the first case, task k is not performed, and input data for task k are sent directly to the higher-level node. In the second case, input data for task k are sent to node i where task k is performed, and then the output is sent to the hierarchically higher-level node. Note that, contrary to the processing component $F_p(i,k)$ (Equation 5.4) that is always negative, the transmission component $F_{tx}(i,k,\boldsymbol{S})$ (Equation 5.5) may also be positive, increasing the task utility function (Equation 5.1) and making it more convenient to perform task k at node i rather than delegating the processing job to the higher-level node.

It may be noted that processing and transmission are not the only mechanisms that affect the network lifetime. The proposed solution does not take into account other mechanisms such as sensing or actuation because they are supposed to be already assigned to the nodes. Hence, they are considered static reasons of lifetime reduction that cannot be changed by a strategic change, so their contribution to the task utility function is null.

Normalization factor $NF(k)$

In Equation 5.1, the normalization factor $NF(k)$ is introduced so as to make the magnitudes of the network lifetime and task completion time components comparable to each other. Since $NF(k)$ is independent from the task assignment strategy, its value can be computed offline before the negotiation starts. The normalization factor is computed by

$$NF(k) = \frac{\overline{\Omega}_\tau(k)}{\overline{\Omega}_t(k)} \tag{5.7}$$

where $\overline{\Omega}_\tau(k)$ and $\overline{\Omega}_t(k)$ are the mean values of $\Omega_\tau(i, k, \boldsymbol{S})$ and $\Omega_t(i, k)$, computed over all nodes $i \in X$.

Weighting factor α

The parameter α introduced in Equation 5.1 is a coefficient of the network lifetime component in the task utility function. If α tends to 0, the utility function is mostly influenced by the task completion time component; the higher its value is, the more the changes in the network lifetime component are reflected by the utility function. Clearly, if α tends to 1, the task completion time component $\Omega_t(i, k)$ and the network lifetime component $\Omega_\tau(i, k, \boldsymbol{S})$ have comparable magnitudes, and therefore comparable influences on the utility function.

Fig. 5.3 shows how different values of α affect the task utility function $u_k(\boldsymbol{S})$. In particular, it is interesting to note that when $\alpha = 0.5$, if $\Omega_t(i, k)$ shrinks by 10% an increase of $\Omega_\tau(i, k, \boldsymbol{S})$ of 5% is necessary in order to keep the same value of $u_k(\boldsymbol{S})$; on the other hand, when $\alpha = 2$, if $\Omega_t(i, k)$ shrinks by 10% the same value of $u_k(\boldsymbol{S})$ could only be reached with an increase of 20% of $\Omega_\tau(i, k, \boldsymbol{S})$.

5.2.3 Network Utility Function

Given the task utility function defined in Equation 5.1, the network utility function is defined as the sum of all task utility functions for those tasks that are not yet assigned to nodes

$$u_g(\boldsymbol{S}) = \sum_{k \in T_{next}} u_k(\boldsymbol{S}) \tag{5.8}$$

This Equation implies that the network utility function is maximized when the sum is maximized. However, this is only 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 maximization itself, particularly for large networks. For this reason, the TAN algorithm lets each node negotiate only

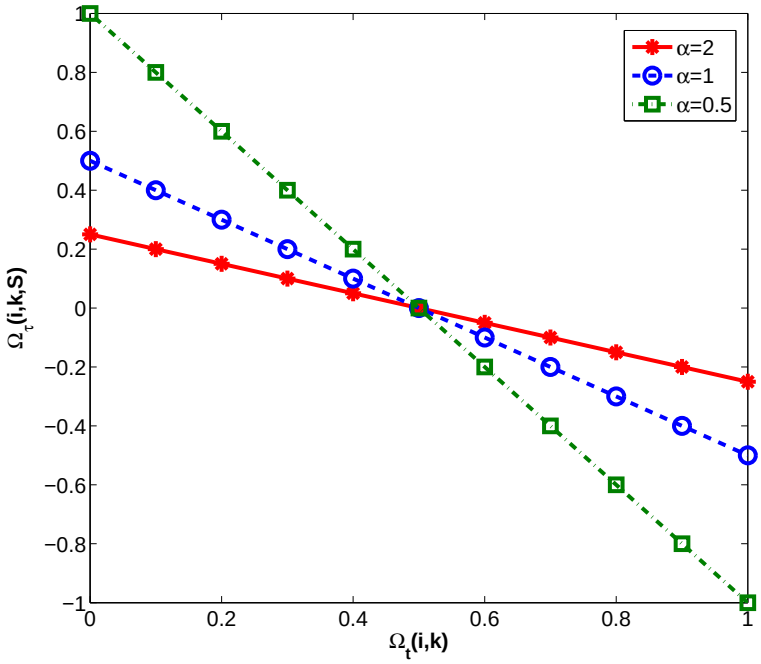


Figure 5.3: Network lifetime component $\Omega_\tau(i,k,\boldsymbol{S})$ with respect to task completion time component $\Omega_t(i,k)$, supposing $s(i,k) = 1$, for task utility function value of $u_k(\boldsymbol{S}) = 0.5$ and different values of weighting factor α

with its neighbors, achieving a sub-optimal but computationally more efficient solution. Hence, the definition of a *node utility function* is required.

5.2.4 Node Utility Function

The node utility function must be defined in such a way that any increment in its value must correspond to an equivalent increment in the network utility function of Equation 5.8. In doing so, an approximation based on local negotiations can be obtained, without the need for network-wide negotiations. The node utility function u_i can then be written as an aggregation of the *marginal* contributions $u_k^{mar}(\boldsymbol{s}(i))$ of node i to each task k (and therefore to the network utility function) given by

$$u_i(\boldsymbol{s}(i)) = \sum_{k \in T_{next}} u_k^{mar}(\boldsymbol{s}(i)) \tag{5.9}$$

The marginal contribution is defined as a *Wonderful Life Utility* (WLU) in [92]. WLU is the difference between the task utility for a given node strategy $\boldsymbol{s}(i)$ and the task utility for the null strategy $\boldsymbol{s}_0(i)$, in which all the elements are equal to 0, i.e. the node is not contributing to any task. The marginal utility u_k^{mar} of node i to task k is then computed by

$$u_k^{mar}(\boldsymbol{s}(i)) = u_k(\boldsymbol{s}(i)) - u_k(\boldsymbol{s}_0(i)) \tag{5.10}$$

Note that marginal utility is null when the node is not contributing to the task as a part of its strategy.

From Equation 5.10 it is possible to infer that the marginal contribution of node i to task k is not null only if the node is contributing to the task. Furthermore, since the network utility function in Equation 5.8 is a summation of task utility functions (Equation 5.1), a node contributes to the network utility when it contributes to at least one task. Therefore, a change in node i's strategy $\boldsymbol{s}(i)$ that increases its utility entails the same increment in the network utility. Let $\boldsymbol{s}(i)$ be the strategy of node i at time t and let $\boldsymbol{s}^*(i)$ be the strategy of node i at time $t^* > t$

$$u_i(\boldsymbol{s}^*(i)) - u_i(\boldsymbol{s}(i)) = \sum_{k \in T_{next}} mu_k(\boldsymbol{s}^*(i)) - \sum_{k \in T_{next}} mu_k(\boldsymbol{s}(i)) =$$
$$= \sum_{k \in T_{next}} \Big(u_k(\boldsymbol{s}^*(i)) - u_k(\boldsymbol{s}(i)) \Big) = u_g(\boldsymbol{s}^*(i)) - u_g(\boldsymbol{s}(i))$$

This property is particularly desirable because it implies that the game under consideration is a *potential* game [93], where the *potential function* is given by the network utility function. A consequence is that this game has at least one pure Nash equilibrium [93][1]. Furthermore, potential games have the *Finite Improvement Property*: every sequence of changes in the strategy that improves the network utility converges to a Nash equilibrium in finite time. This property ensures that many simple adaptive processes, such as the Distributed Stochastic Algorithm (DSA) [94], converge to Nash equilibria.

5.3 Task Allocation Negotiation Algorithm

The aim of the TAN algorithm is to maximize the network utility function given by Equation 5.8. To achieve this purpose, individual node utility functions are maximized by means of a negotiation accomplished by neighboring nodes. The negotiation is based on a greedy search algorithm, called DSA [94], which is proved [95] to provide a solution more quickly than existing algorithms such as Distributed Breakout Algorithm (DBA) [94] and Maximal Gain Messaging (MGM) [96]. Before introducing TAN in detail, it is essential to explain DSA, which is next.

5.3.1 Distributed Stochastic Algorithm

DSA is a synchronous algorithm. At each time step t, each node that is involved in the negotiation and changed its strategy in the previous time step $t-1$, sends to all the other involved nodes a *strategy update* message (SUM) which contains its new strategy $s^*(i)$. If a node receives

[1] Pure Nash equilibria are characterized by a unique outcome, contrary to mixed Nash equilibria where the outcome is stochastically variable.

Algorithm 1 DSA for node i

```
1:  while counter < t_wait do
2:      if s*(i) was found at t − 1 < t then
3:          broadcast a SUM to all neighbors
4:      end if
5:      if a SUM has been received then
6:          counter ← 0
7:          v ← rand()
8:          if v > p then
9:              find s(i) that maximizes u_i(s(i))
10:             if u_i(s(i)) > u_i(s*(i)) then
11:                 s*(i) ← s(i)
12:             end if
13:         end if
14:     else
15:         counter ← counter + t_step
16:     end if
17: end while
```

a *strategy update* message, and if the value v assigned by its random number generator $rand()$ is higher than a given probability p, then a new strategy $\boldsymbol{s}(i)$ that maximizes its own utility $u_i(\boldsymbol{s}(i))$ (Equation 5.9) is computed. If the utility is already maximized by the current strategy, no changes occur. DSA terminates when no new *strategy update* messages are received within the last t_{wait} seconds. The pseudocode for DSA is given in Algorithm 1.

The probability value p is known as the *degree of parallel executions*. The value of p affects the performance of DSA: the higher it is, the higher the probability that all the nodes simultaneously change their strategy, thus increasing the communication costs, and the algorithm's convergence time. On the other hand, low values may lead to a large convergence time, due to less frequent changes in node strategies.

5.3.2 Task Allocation Negotiation

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 $u_g(\boldsymbol{S})$. At each step of the TAN algorithm, some nodes in the network hold some data, on which some tasks in T_{next} need to be performed. These nodes first exchange the information required for DSA with their neighboring nodes; then, all the nodes in the involved neighborhoods run the DSA until a strategy is found. Finally, the tasks are performed according to the strategy determined with DSA, and processed data are sent to the higher-level node. The TAN algorithm will be now described in detail.

The pseudo-code of TAN is provided in Algorithm 2. The algorithm starts as soon as source

Algorithm 2 TAN

```
Source tasks are performed
while T_next ≠ {} do
    for all i ∈ X_data do
        send INFO to all nodes in n(i)
        for all j ∈ n(i) do
            send INFO to all nodes in {{i, n(i)}\{j}}
        end for
        for all nodes in {i, n(i)} do
            execute DSA
        end for
        for all x^k_proc ∈ X_proc do
            perform t^k_proc
            T_next ← {T_next\{t^k_proc}}
            T_prev ← {T_prev, {t^k_proc}}
            if x^{k+1}_proc ≠ x^k_proc then
                send DATA to x^{k+1}_proc
            end if
        end for
        send DATA to the higher-level node
        update X_data
    end for
end while
```

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, these are the 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, $n(i)$. An INFO message includes:

1. $\boldsymbol{s}(i)$, e_i^{ins}, $\boldsymbol{e}_i^{tx}$, e_i^{res} (see Table 2.2);
2. the subset $T'_{prev} \subseteq T_{prev}$ of tasks that are already performed on the data that node i currently holds.

Since initially the only tasks already performed are source tasks, T'_{prev} is made of the source tasks that are assigned to i.

All the nodes in $n(i)$ reply sending an INFO message with their own information to their neighbours.

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 5.8).

Let $X_{proc} = \{x^1_{proc}, \dots, x^k_{proc}, \dots\}$ be the set of nodes in $n(i)$, for which the strategy of

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 set of processing tasks to be performed. 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 the node x^{k+1}_{proc} for which $t^{k+1}_{proc} : e^T_{k(k+1)} \in E_T$. 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.

5.3.3 Computational Complexity of the Node Utility Function Maximization

The maximization of the node utility function $u_i(\boldsymbol{s}(i))$ defined in Equation 5.9 is a MILP problem [74]. It is well known that MILP problems are optimally solved using branch-and-bound algorithms [97], which, in the worst case, have a complexity that grows exponentially with respect to the number of variables.

The complexity of the node utility function is related with the number of variables a node uses to compute the function. Note that the tasks in $\boldsymbol{d}(i)$ are node i's variables used for executing TAN. Hence, the number of variables $N_{vars}(i)$ of node i is equal to $N_{task}(i)$. A node i can only perform the tasks that are set in its vector $\boldsymbol{d}(i)$, but not all tasks in T_{prev}. The number of node i's tasks is found by

$$N_{vars}(i) = \sum_{\lambda \in T_{next}} d(i, \lambda)$$

In the worst case, the complexity of maximizing $u_i(\boldsymbol{s}(i))$ is equal to the complexity of performing an exhaustive search among all the possible combinations of processing assignments, i.e. $2^{N_{vars}} \times N_{operations}$, where $N_{operations}$ is the number of operations needed to evaluate one node utility function $u_i(\boldsymbol{s}(i))$. Note that computational complexity can be considerably reduced using sub-optimal heuristic algorithms, such as genetic or tabu-search algorithms.

5.3.4 Message Complexity of the Node Utility Function Maximization

INFO, SUM and DATA messages are sent during the TAN algorithm execution.

The INFO message is broadcast within the cluster once by each node. Let $N_{cluster}(i)$ be the number of nodes within the cluster that node i belongs to, and let $n(\text{INFO})$ be the number of bits required for an INFO message. Recalling the INFO message structure, $n(\text{INFO})$ consists of: $|\boldsymbol{s}(i)|$ boolean values, $|\{e^{ins}(i), \boldsymbol{e}^{tx}_i, e^{res}_i\}|$ double values, and $|T'_{prev}|$ identification values (which

can be, for example, unsigned chars). The number of transmitted bits due to the exchange of INFO messages for each negotiation is therefore $N_{cluster} \times n(\text{INFO})$.

During the negotiation, a SUM message consisting of $|\boldsymbol{s}(i)|$ boolean values is broadcast within the cluster whenever node i changes its strategy. Let N_{steps} be the number of steps taken until the negotiation converges, and let $n(\text{SUM})$ be the number of bits of a SUM message. In the worst case, at every time step, each node in the cluster changes its strategy. Hence, the worst case number of transmitted bits for SUM messages during each negotiation is $N_{steps} \times N_{cluster} \times n(\text{SUM})$.

After the negotiation process has converged, a DATA message of $n(\text{DATA}, k)$ bits is sent to each node $x^k_{proc} \in X_{proc}$ (see Section 5.3.2), and then to the higher-level node. The dependence of $n(\text{DATA}, k)$ on k is due to the fact that the number of bits of the output data varies based on which task t^k_{proc} is executed. $n(\text{DATA}, k)$ is proportional both to the amount of output data generated by the processed tasks and to the number $|T_{prev}|$ of identification values (which can be, as for T'_{prev}, unsigned char). In the worst case, each node $x^k_{proc} \in X_{proc}$ is different from the following one x^{k+1}_{proc}. Therefore, considering that the last DATA message is sent to the higher-level node, in the worst case, the number of DATA messages is $|X_{proc}| + 1$. Thus, the worst case number of transmitted bits due to DATA messages for each negotiation is $(|X_{proc}| + 1) \times \sum_{k \in X_{proc}} n(\text{DATA}, k)$.

As a result, the worst case number of transmitted bits sent during a single negotiation is

$$\begin{aligned} n(negotiation) = & N_{cluster} \times n(\text{INFO}) + N_{steps} \times N_{cluster} \times n(\text{SUM}) + \\ & |X_{proc}| \times \sum_{k \in X_{proc}} n(\text{DATA}, k) \end{aligned} \tag{5.11}$$

5.4 Performance Analysis

The proposed algorithm has been tested on realistic heterogeneous WSN scenarios, by means of simulations carried out in a MatLab environment. The main setup parameters of the nodes in the modelled WSNs are listed in Table 3.3. The heterogeneity has been introduced not only using sensor nodes with different characteristics from cluster head ones, but also setting node parameters so that they have:

- a non uniform energy consumption: energy consumption parameters are selected randomly in the range from 60% to 140% of the mean values;
- a non uniform initial energy at each node: battery charge is assigned randomly from 20% to 100% of the total charge;

- a non uniform processing speed at each node: processing speed is set randomly from 60% to 100% of the maximum processing speed.

In the following, the performance of the TAN algorithm for two different scenarios will be presented.

5.4.1 Smart City Scenario

The scenario under examination is that of the urban environment described in Section 4.4.2 and depicted in Figure 4.3. Every stretch of street is a cluster, where cluster heads are represented by empty markers. Cluster heads 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.

The application is the same as the one described in Section 4.4.2. 89 different *speed sensing* tasks ($\lambda_1 \div \lambda_{89}$) are identified, one for each sensing node. Since the mean traveling time for each stretch of street is required in order to compare them to each other, 78 *mean speed computing* tasks ($\lambda_{90} \div \lambda_{167}$), and 11 *mean travelling time computing* tasks ($\lambda_{168} \div \lambda_{178}$) are defined, one for each stretch of street. To find the best path, i.e. the best combination of stretches of streets that leads from the Start point to Destination, the sum of the mean traveling times of all the combination of stretches that can be driven one after the other is needed in order to compare 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 4.3, 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). Table 5.1 summarizes the tasks for this scenario.

The described scenario has been simulated, and the performance of the proposed algorithm have been compared with three alternative approaches:

- all the data sent to the sink and processed only by the sink, that is the Start

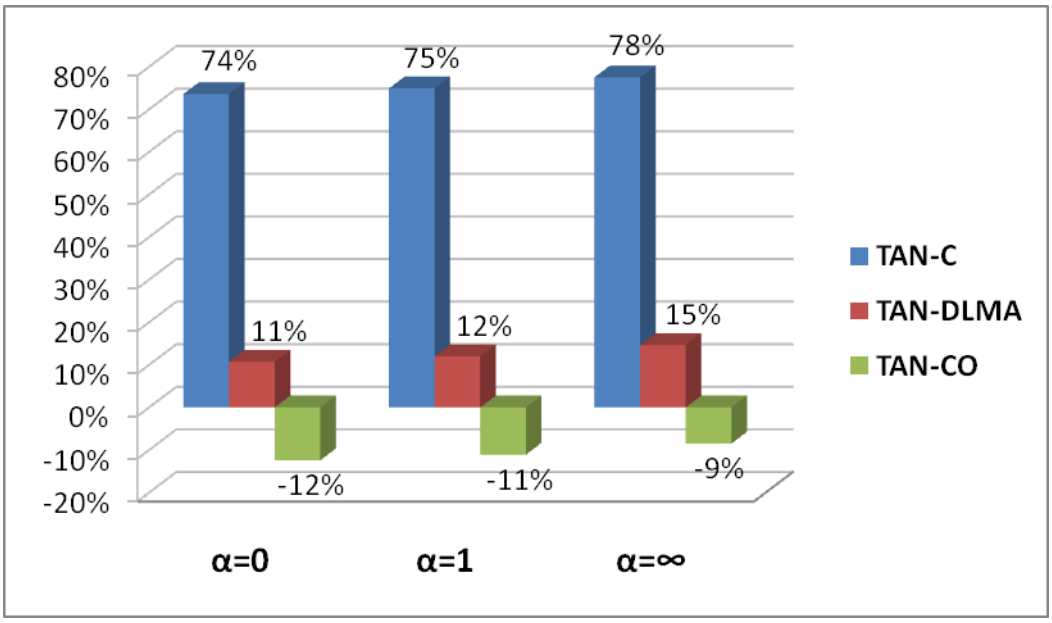
Table 5.1: Tasks for Smart City Scenario

Task	**Description**
$\lambda_1 \div \lambda_{89}$	Speed sensing
$\lambda_{90} \div \lambda_{167}$	Mean speed computing
$\lambda_{l68} \div \lambda_{178}$	Mean travelling time computing
$\lambda_{179} \div \lambda_{186}$	Summation of mean travelling times for different stretches
$\lambda_{187} \div \lambda_{189}$	Choice of the best path

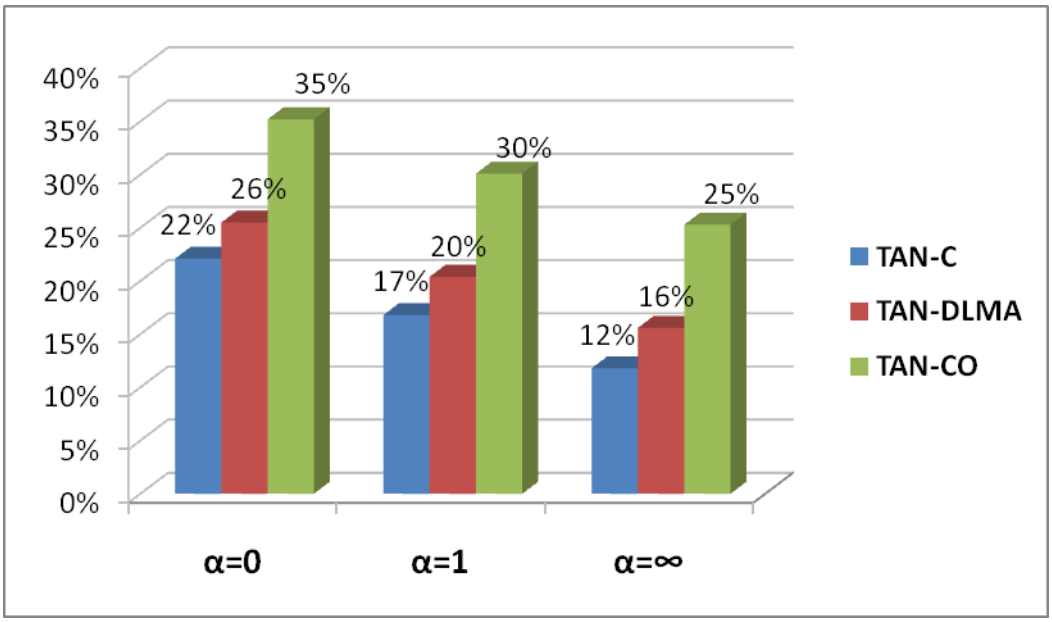
node (mechanism S);

- data processed according to the centralized optimization algorithm described in Chapter 3 (mechanism CO);
- data processed according to DLMA algorithm described in Chapter 4 (mechanism DLMA).

The obtained results for energy consumption and completion time have been compared to those obtained using TAN algorithm.



(a) Energy conservation



(b) Completion time gain

Figure 5.4: Percentage values of mean energy conservation and completion time gain using TAN

Figure 5.4 shows the percentage of energy conservation and completion time gained when using TAN with respect to the alternative approaches. These comparisons are referred to as TAN-S, TAN-DLMA and TAN-CO.

Simulations have been run for:

- $\alpha = 0$: null network lifetime component;
- $\alpha = 1$: comparable Ω_t and Ω_τ;
- $\alpha = \infty$: null task completion time component.

A marked improvement of energy conservation and a good improvement in completion time gain is observed with respect to the static S mechanism. Good results are also observable both for energy conservation and completion time gain with respect to TAN-DLMA. Of course, good results could not be expected for energy conservation in comparison TAN-CO, but a marked improvement in completion time gain is observed, mainly due to the fact that centralized algorithm is more complex and needs more time to be accomplished. It has to be noted that results for $\alpha = 0$ and $\alpha = 1$ have been reported just for completeness, but all the algorithms do not take into account completion time, therefore the most significant results are those obtained for $\alpha = \infty$.

As expected, when α increases, which means that the utility function is more and more over-balanced in favor of its network lifetime component, the energy conservation percentage increases, while the completion time gain decreases; on the contrary, when α decreases, the utility function is over-balanced in favor of its task completion time component: the energy conservation percentage decreases, while the completion time gain increases.

5.4.2 Realistic Random Scenarios

Starting from the results obtained for the Smart City Scenario, it has been investigated how TAN algorithm would work for realistic scenarios with different characteristics. A rectangular-shaped outdoor environment was studied (e.g., a vineyard, a seaport, a tourist plaza), where nodes have been positioned randomly following a uniform distribution. The random network is created taking into account two parameters: the nodes density, i.e. the number of nodes per square meter, and the cluster numerosity, which is the mean number of nodes inside each cluster. The application assigned to each scenario is described by a random DAG. This DAG is obtained by starting with a set of tasks and adding edges among them at random, provided that this configuration represents a DAG and the number of processing tasks is equal to a fixed value. The number of single instructions required to perform each task and the number of output bits are set randomly according to a uniform distribution from 270000 to 330000, and from 720 bits

Table 5.2: Characteristic Parameters Values for Realistic Random Scenarios

Parameter	**Min**	**Default**	**Max**
Node density [nodes/m^2]	0.2	0.3	0.4
Cluster numerosity [nodes/cluster]	5	15	25
Number of processing tasks	10	20	30
Distribution of processing tasks	10%	50%	100%

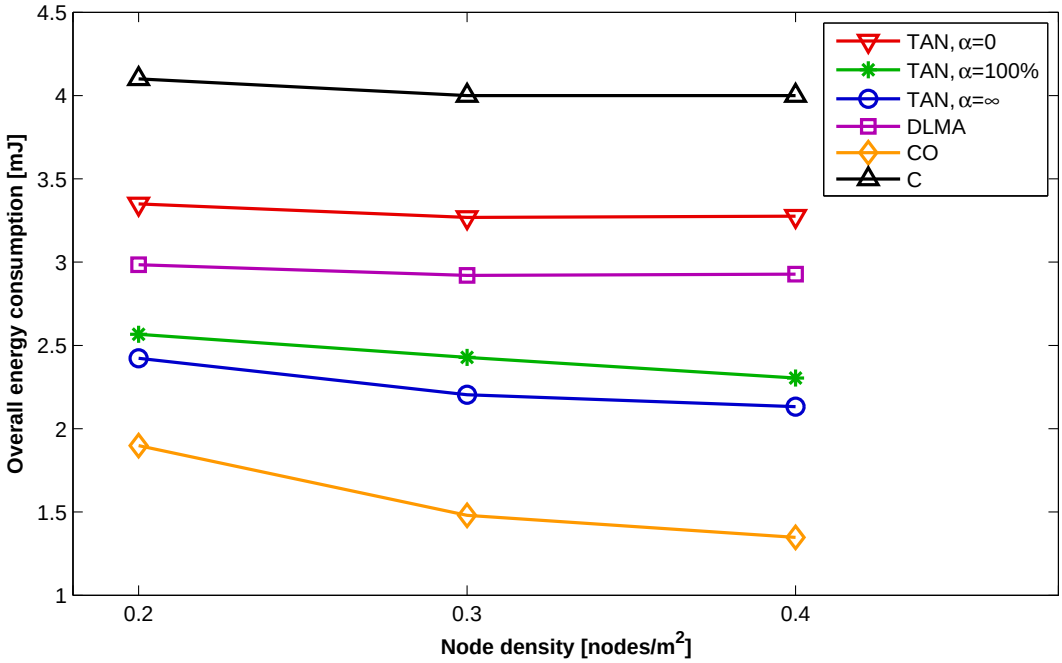
to 880 bits respectively [98]. The application deadline is fixed and set to 80 ms. The ability of each node to perform a processing task, that is the values of the elements corresponding to the processing tasks of each vector $\boldsymbol{d}(i)$ described in Section 5.1, is assigned according to the processing task distribution parameter, which defines the probability of each node to be able to perform a given processing task. Table 5.2 shows the different parameter values used to simulate the described scenario.

Note that the Smart City Scenario studied previously corresponds to a network with a low nodes density of about 0.15 nodes/m^2, a low cluster numerosity of about 8 nodes/cluster, a high number of processing tasks equal to 100 and a very low task distribution of about 6.9%. Although this task distribution is extremely low if compared with those of the Random Scenarios, it needs to be noted that, contrary to Random Scenarios, where the ability of each node to perform a task is assigned randomly according to the tasks distribution parameter, in the Smart City Scenario the ability of each node to perform a task is strictly related to that task, i.e. only the nodes that are more suitable to perform one given task have the ability to execute it. For this reason, worse results are expected in the Random Scenarios with respect to the Smart City Scenario, when focusing on the task distribution parameter.

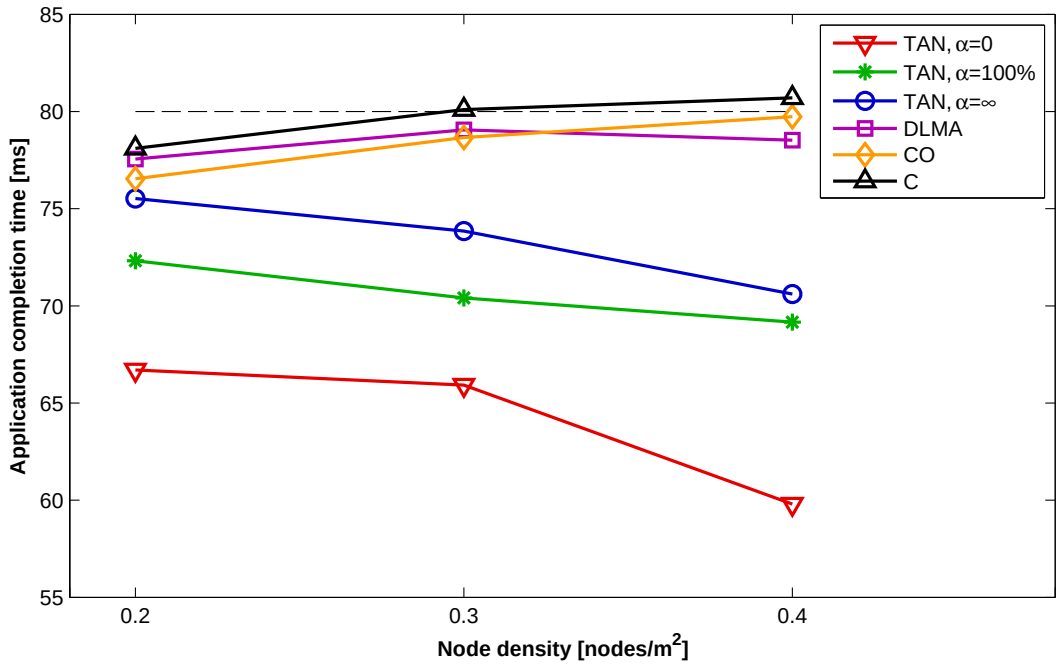
In the following, the performance of TAN algorithm when these parameters change will be discussed.

Performance Changing Node Density

The first run of simulations regards different node densities, with the aim of observing how different distances among nodes affect the algorithm performance. Figure 5.5 shows how the mean overall energy consumption and application completion time change when node density increases, for different values of α.Since the network lifetime component (Equation 5.3) depends on the hop distance, an improvement in energy consumption is observed using TAN when node density increases, particularly when α increases. Of course, when $\alpha = 0$ the network lifetime component is not taken into account for choosing the best strategy, and this is reflected in the flatness of the related curve. Still, there is an energy conservation even in this case. This is

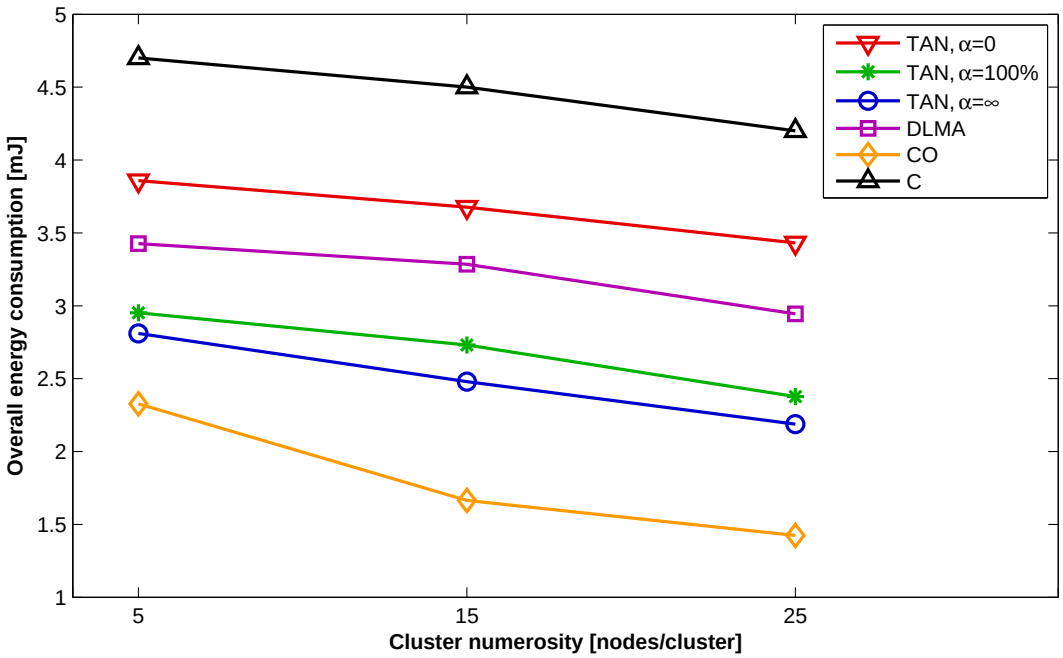


(a) Overall energy consumption

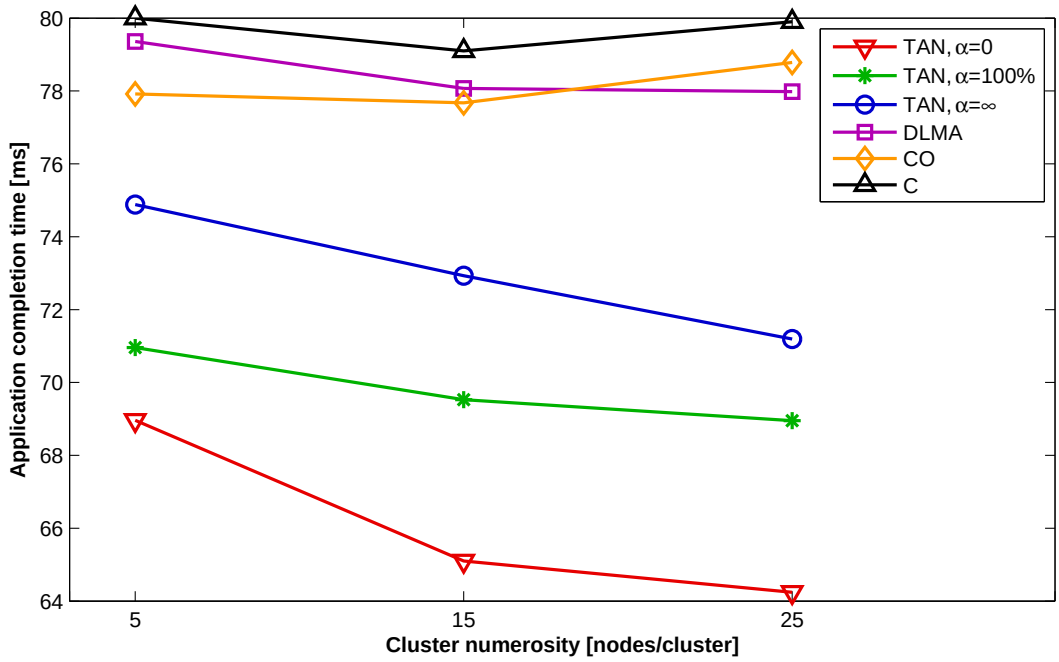


(b) Application completion time

Figure 5.5: Mean overall energy consumption and application completion time for TAN, DLMA, CO and S task assignment mechanisms, for different node densities



(a) Overall energy consumption



(b) Application completion time

Figure 5.6: Mean overall energy consumption and application completion time for TAN, DLMA, CO and S task assignment mechanisms, for different cluster numerosity parameters

mainly due to the fact that, even though TAN's goal is not improving the network lifetime component when $\alpha = 0$, improving the application completion time component entails that tasks are more likely executed before data arrives to the sink. Since executing tasks, in most cases, reduces the amount of data to be sent, a reduction in the completion time leads to a reduction in the transmission cost.

An improvement in the application completion time is noticed when node density increases. The main reason of this is that completion time is influenced both by tasks completion time and TAN completion time: when node density increases, negotiations among nodes are quicker, hence TAN completion time decreases.

Performance Changing Cluster Numerosity

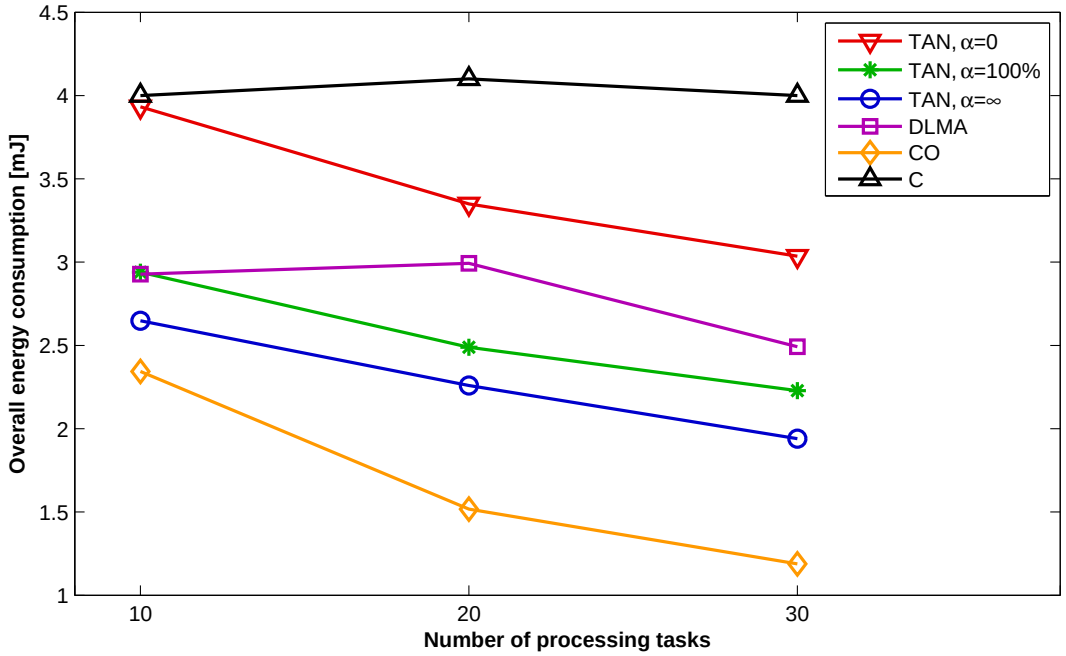
The second set of simulations was run changing the cluster numerosity parameter. In Figure 5.6, the overall energy consumption and application completion time variations are shown with respect to different values of α and different number of nodes per cluster. Note that results are similar to the previous case, but for different reasons. In fact, when node numerosity increases, the likelihood that the strategy is chosen in a few negotiations increases, reducing both energy consumption and completion time.

Performance Changing the Number of Processing Tasks

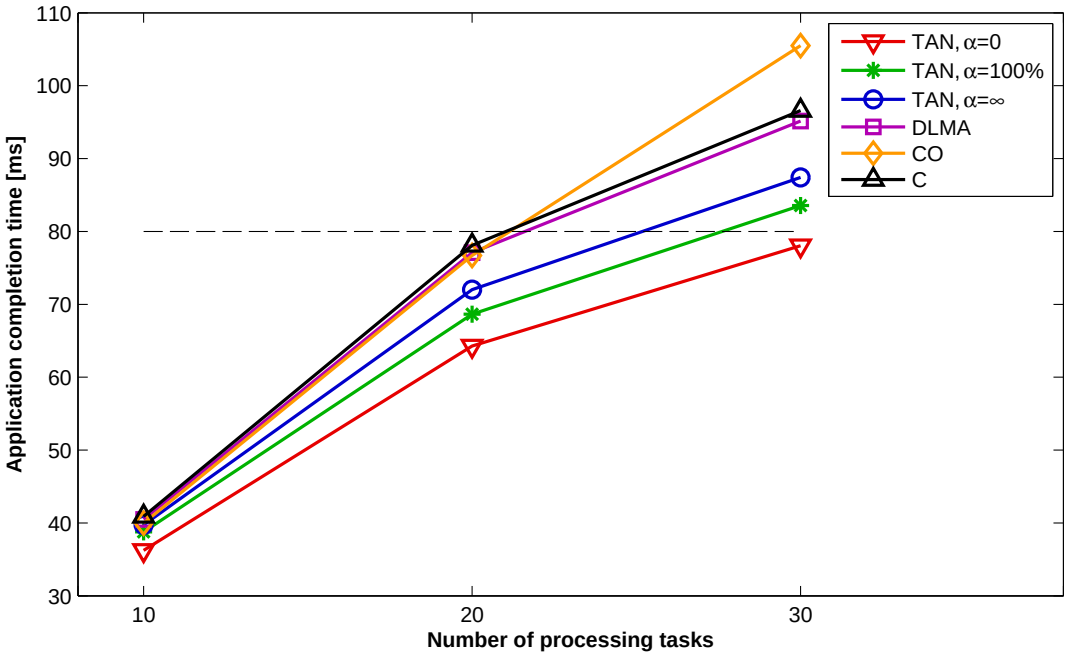
By changing the number of processing tasks, TAN's performance has been studied for more complex applications. Since the application deadline is fixed, increasing the number of tasks means decreasing the available time for each task, and therefore affecting the task completion time component. The results for these scenarios are shown in Figure 5.7. A marked improvement of TAN algorithm is observed in both energy consumption and completion time when the number of processing tasks increases from 10 to 20. This is explainable considering that, for a low number of processing tasks such as 10, the results for an optimized strategy can not be much different than the case where all the processing tasks are assigned to the sink. On the contrary, the difference is significant when the number of processing tasks starts to increase. Note that TAN with $\alpha = 0$ is the only algorithm for which, even with 30 tasks, the deadline is never missed.

Performance Changing Processing Task Distribution

With the last set of simulations, how the processing tasks distribution parameter affects TAN behavior has been studied. It needs to be noted that the ability of each node to perform a given

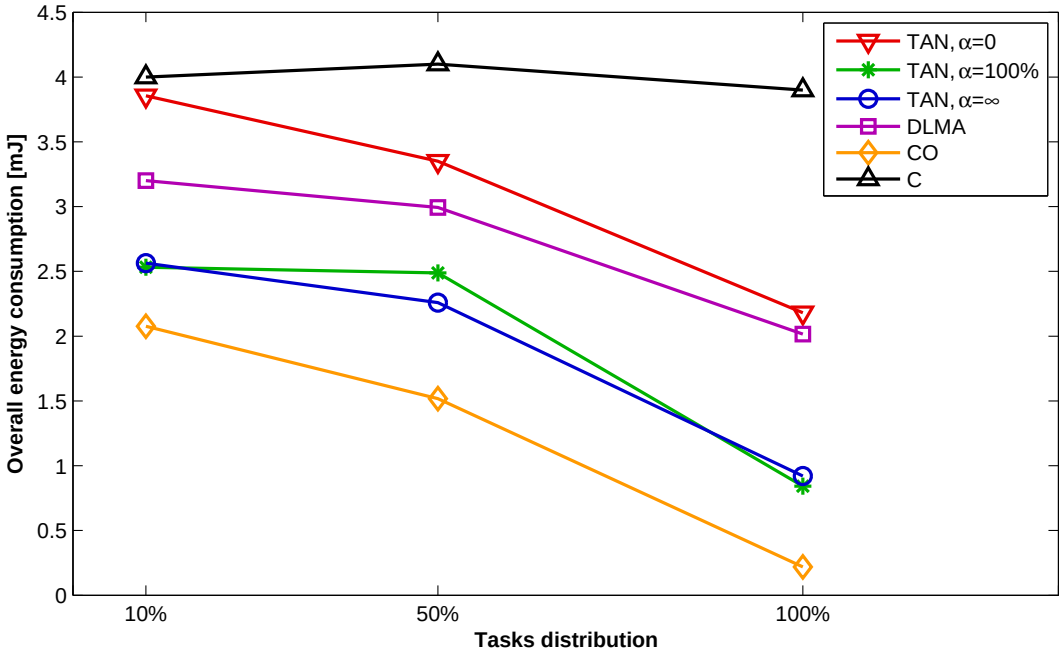


(a) Overall energy consumption

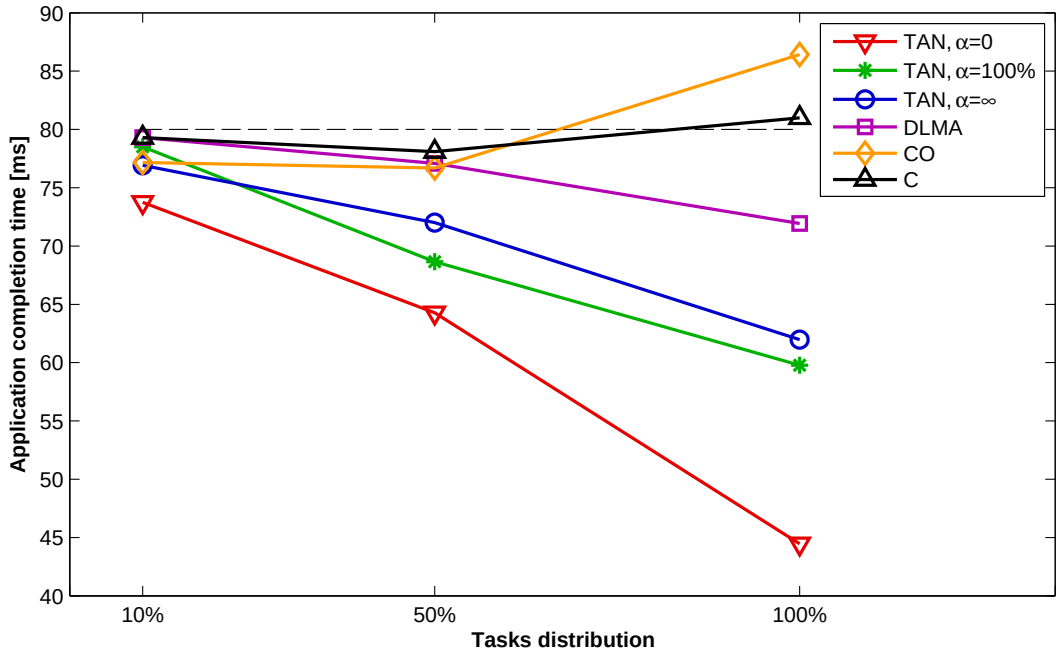


(b) Application completion time

Figure 5.7: Mean overall energy consumption and application completion time for TAN, DLMA, CO and S task assignment mechanisms, for different numbers of processing tasks



(a) Overall energy consumption



(b) Application completion time

Figure 5.8: Mean overall energy consumption and application completion time for TAN, DLMA, CO and S task assignment mechanisms, for different processing task distributions

processing task is assigned randomly. This means that in most real cases, where these abilities are assigned more suitably based on the application to be performed, results will be better than those obtained in these scenarios. Figure 5.8 shows TAN performance with respect to alternative algorithms for different values of α and different tasks distribution values.

A marked improvement both in overall energy consumption and in application completion time using TAN is noticed when every node in the network is able to perform every processing task. In fact, when the processing tasks distribution parameter is equal to 100% TAN algorithm can actually choose the best nodes in the cluster to perform each task, and not just the best among a small number of them. On the opposite, results for processing tasks distribution parameters lower than 50% are not much different from each other.

Chapter 6

Conclusions and future works

In this thesis, the problem of efficient dynamic deployment of application tasks into WSNs with the aim of extending the network lifetime has been studied. The scenario of a multi-hop network has been analyzed, along with the main components that govern the network dynamics in terms of energy consumption. Accordingly, the related models have been defined. The three algorithms proposed are intended to dynamically adapt to the behavior of the network, offering an energy-efficient task allocation which minimizes the impact of the assigned application on the network lifetime.

In Chapter 3, a framework based on a centralized algorithm for the reduction of the overall energy consumption have been presented. Significant improvements were observed in terms of energy saving. In particular, the framework resulted to be particularly energy conserving for networks with high node densities, for complex applications, and for heterogeneous networks. However, centralized algorithms suffer from computational complexity, especially for large WSNs.

In Chapter 4, a decentralized algorithm for the maximization of the network lifetime, the DLMA, have been described. Contrary to centralized solutions, distributed solutions adapt quickly to network changes, reduces the control message exchange overhead, and scales well with the number of nodes. The DLMA is based on the gossip communication paradigm, which allows neighboring nodes to iteratively and asynchronously perform a local optimization to increase network lifetime. This solution presents good results when compared with other static task assignment mechanisms. It performs well, even though it is outperformed, when compared to the centralized solution in Chapter 3. Nevertheless, its computational complexity makes it suitable even in large scale networks, where a centralized algorithm would not be practically implemented. Even though DLMA is a better solution, it does not take into account the execution time of the application assigned to the network. This might lead to an inconvenient task assignment, where the application deadline is reached before the application is executed.

The TAN algorithm, proposed in Chapter 5, is a distributed task assignment algorithm which aims to reduce the overall network energy consumption and the application execution time contemporarily. This algorithm, implemented for a hierarchical WSN, is based on the rules of non-cooperative game theory. Its performance resulted in a reduction of both overall energy consumption and application completion time, particularly for complex scenarios with high node density and cluster numerosity, for more complex applications, and for more versatile nodes.

The study done so far has led to the acquisition of the expertise required to widen the research from WSNs to the IoT scenarios. Future work will be focused on the specification of an interoperability middleware that enables the implementation of multi-objective optimization methods into heterogeneous networks. Interoperability among different devices will be ensured by the use of ontologies. Ontologies are needed to describe network devices, their capabilities, available resources, and requirements of the services that the network must be able to supply. A meta-model of the architecture will then be defined in order for devices to be able to autonomously cooperate among each other. Hence, the critical challenge of allocation and management of available resources such as electrical energy, memory, processing, and object capability to perform a given task, will be analyzed. In this scenario, all nodes need to interoperate in order to reason and allocate the available resources in a distributed way, with the aim of executing the application assigned to the network. Most of these decisions should be taken autonomously to avoid centralized solutions, which usually limit the flexibility of the systems and requires intense control data exchanges. The vision is that of a multi-technology scenario where the adaptive network components cooperate dynamically in order to achieve optimal performance not only from the energy consumption point of view, but also with regards to other requirements such as time consumption, Quality of Service, and Quality of Information.

Appendix A

Notation used throughout the paper

A.1 Energy Consumption Model

e_i^{sens}	Sensing energy consumption for node i
e_{ih}^{proc}	Processing energy consumption for node i and task h
I_h	Number of instrauction needed to perform task h
e_i^{ins}	Average energy consumption per instruction related to node i
P_{ij}^T	Radio frequency power consumption for transmitting from node i to node j
P_j^R	Radio frequency power consumption for receiving in node j
P_i^{T0}	Power consumption related to the transmitting circuitry
P_j^{R0}	Power consumption related to the receiving circuitry
P_i^A	Power consumption related to the PA of node i
δ_{ij}	Distance between node i and node j
P_i^{Tx}	Output power at node i antenna
η_i	Drain efficiency for node i
α_{PL}	Path loss exponent
φ_{ij}	Coefficient proportional to the reception power and the characteristic parameters of the antennas of the transmitting node i and the receiving node j
R	Data rate transmission
e_{ij}^{tx}	Per-bit energy to transmit from node i to node j
e_j^{rx}	Per-bit energy to receive in node j

A.2 Network Model

$\mathcal{G}_X = (X, E_X)$	DAG representing the network model
$X = \{1, \ldots, i, \ldots, N\}$	Set representing the nodes of the network model
$E_X = (e_{ij}^X)$	Set representing the link of the network, where e_{ij}^X represents a connection from nome i to node j
$\mathbf{\Delta} = (\delta_{ij})$	Matrix of the pairwise distances (in meters) between adjacent nodes i and j
$\mathbf{\Phi} = (\varphi_{ij})$	Matrix of pairwise parameters φ_{ij} between adjacent nodes i and j (see Section 2.1)
e_i^{sens}	Average energy spent by node i to perform a sensing task
$e^{ins}(i)$	Average energy spent by node i to perform a single instruction
$\boldsymbol{e}_{ij}^{tx} = (e_{ij}^{tx})$	Vector of the pairwise per-bit energies to transmit from node i to node j
e_i^{rx}	Per-bit reception energy at node i
e_i^{res}	Residual energy of node i

A.3 Centralized Task Allocation Algorithm

P^{tot}	Total cost value for the network, related to the overall power consumption executing a given application
$\mathcal{G}_T = \{T, E_T\}$	DAG representing the application model
$T = \{t_1, \ldots, t_l, \ldots, t_L\}$	Set representing the tasks in the application model
$E_T = (e^T_{uv})$	Set representing the edges of the application model, where e^T_{uv} represents a unidirectional data transfer from task u to task v
$D_i = \{d_{i1}, \ldots, d_{im}, \ldots, d_{il_i}\}$	Set of the tasks the node i is able to perform
s_i	Status of node i that defines which task t_l is assigned to node i
$\boldsymbol{S} = \{s_1, \ldots, s_i, \ldots, s_N\}$	Set of statuses for the whole network
$\boldsymbol{\Theta}^{out}_i = \{\theta^{out}_{i1}, \ldots, \theta^{out}_{ih}, \ldots, \theta^{out}_{iH}\}$	Output traffic for node i
$\boldsymbol{\Theta}^{in}_i = \{\theta^{in}_{i1}, \ldots, \theta^{in}_{ih}, \ldots, \theta^{in}_{iH}\}$	Input traffic for node i
$\theta^{out}_{ih} = \{k^{out}_{ih}, f^{out}_{ih}\}$	Output traffic flow for node i, related to task h, where k^{out}_{ih} and f^{out}_{ih} are the number of transmitted bits and the transmitting frequency for that flow
$\theta^{in}_{ih} = \{k^{in}_{ih}, f^{in}_{ih}\}$	Input traffic flow for node i, related to task h, where k^{in}_{ih} and f^{in}_{ih} are the number of received bits and the receiving frequency for that flow
γ_i	Coefficient in inverse proportion to e^{res}_i
P^{sens}_i	Sensing cost function for node i
P^{proc}_i	Processing cost function for node i
P^{tx}_i	Communication cost function for node i

A.4 Decentralized Lifetime Maximization Algorithm

$\mathcal{N}_i$	Neighbors for node i
$\mathcal{N}_{out,i}$	Out-neighbors for node i, i.e. nodes that receive information from node i
$\mathcal{N}_{in,i}$	In-neighbors for node i, i.e. nodes that send information to node i
$T^s = \{t_1^s, \ldots, t_W^s\}$	Sequence of sensing tasks that have to be executed by the network
$T^p = \{t_1^p, \ldots, t_L^p\}$	Sequence of processing tasks that have to be executed by the network
$\mathcal{G}_T = (\{T^s, T^p\}, E_T)$	DAG representing the application model
$E_T = (e_{uv}^T)$	Set representing the edges of the application model, where e_{uv}^T represents a unidirectional data transfer from task u to task v
$\boldsymbol{m}_i = (m_{iw})$	Binary state of the sensing tasks assigned to node i, where $m_{iw} = 1$ if node i performs sensing task w
$\boldsymbol{s}_i = (s_{il})$	Binary state of the processing tasks assigned to node i, where $s_{il} = 1$ if node i performs processing task l
$\boldsymbol{S} = (\boldsymbol{s}_i)$	Matrix of the states of all nodes
$\boldsymbol{d}_i = (d_{il})$	Binary vector representing the processing tasks that node i is able to perform, where $d_{il} = 1$ if node i is able to perform task l
$\boldsymbol{D} = (\boldsymbol{d}_i)$	Matrix of all the vectors of the processing tasks that each node is able to perform
τ	Network lifetime
$\boldsymbol{\Theta}_i^{out} = \{\theta_{i1}^{out}, \ldots, \theta_{ih}^{out}, \ldots, \theta_{iH}^{out}\}$	Output traffic for node i
$\boldsymbol{\Theta}_i^{in} = \{\theta_{i1}^{in}, \ldots, \theta_{ih}^{in}, \ldots, \theta_{iH}^{in}\}$	Input traffic for node i
$\theta_{ih}^{out} = \{k_{ih}^{out}, f_{ih}^{out}\}$	Output traffic flow for node i, related to task h, where k_{ih}^{out} and f_{ih}^{out} are the number of transmitted bits and the transmitting frequency for that flow
$\theta_{ih}^{in} = \{k_{ih}^{in}, f_{ih}^{in}\}$	Input traffic flow for node i, related to task h, where k_{ih}^{in} and f_{ih}^{in} are the number of received bits and the receiving frequency for that flow
P_i^{sens}	Sensing cost function for node i
P_i^{proc}	Processing cost function for node i
P_i^{tx}	Communication cost function for node i
P_i	Overall cost function for node i
$\widehat{E}_i$	Upper bound to the energy spent by solving a local optimization in node i

A.5 Task Allocation Negotiation Algorithm

$\mathcal{G}_T = (T, E_T)$	DAG representing the application model
$T = \{1, \ldots, \lambda, \ldots, \Lambda\}$	Set representing the tasks in the application model
$E_T = (e_{uv}^T)$	Set representing the edges of the application model, where e_{uv}^T is a unidirectional data transfer from task u to task v
$\boldsymbol{s}(i) = (s(i, \lambda))$	Binary task assignment strategy vector of node i, where $s(i, \lambda)$ is the state of node i for task λ, such that $s(i, \lambda) = 1$ if task λ is assigned to node i
$T_{in}(\lambda)$	Set of the input tasks for task λ
$\boldsymbol{d}(i) = (d(i, \lambda))$	Binary vector of the tasks that node i is able to perform, where $d(i, \lambda) = 1$ if node i is able to perform task λ
$T_{prev} = \{1, \ldots, h, \ldots, H\}$	Set of already assigned tasks
$T_{next} = \{1, \ldots, k, \ldots, K\}$	Set of tasks that are yet to be assigned
$\boldsymbol{S} = (\boldsymbol{s}(i))$	Strategy of all the nodes in the network
$u_k(\boldsymbol{S})$	Task utility function for task k related to the network strategy $\boldsymbol{S}$
$\Omega_t(i, k)$	Task completion time component of task k when it is performed by node i
$\Omega_{tau}(i, k, \boldsymbol{S})$	Network lifetime component when task k is performed by node i according to the strategy $\boldsymbol{S}$
$NF(k)$	Task utility function normalization factor for task k
α	Weighting factor of the task utility function
$t_d(k)$	Deadline for successfully completing task k
$I(k)$	Number of required instructions for task k
$t_c(i, k)$	Completion time of task k if it is performed by node i
$t_{instr}(i)$	Time needed by node i to perform a single instruction
$F_p(i, k)$	Term of the network lifetime component related to the change in network lifetime due to the processing cost needed to perform task k in node i
$F_{tx}(i, k, \boldsymbol{S})$	Term of the network lifetime component related to the change in network lifetime due to the transmission (and reception) of the necessary data for task k to node i
$C_{tx}(\boldsymbol{p}_{i \to j}, k)$	Cost to transmit (and receive) data from node i to node j
$n(k)$	Number of output bits for task k
$u_g(\boldsymbol{S})$	Network utility function
$u_i(\boldsymbol{s}(i))$	Node utility function
$u_k^{mar}(\boldsymbol{s}(i))$	Marginal utility function of node i for task k
p	Degree of parallel executions for DSA

Acronyms

μAMPS	Micro-Adaptive Multi-domain Power-aware Sensors
ADS	Advanced Deployable System
ARPANET	Advanced Research Projects Agency Network
BWRC	Berkely Wireless Research Center
CCA	Cooperative Collision Avoidance
CEC	Cooperative Engagement Capability
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DAG	Directed Acyclic Graph
DARPA	Defense Advanced Research Projects Agency
DBA	Distributed Breakout Algorithm
DG	Directed Graph
DLMA	Decentralized Lifetime Maximization Algorithm
DSA	Distributed Stochastic Algorithm
DSN	Distributed Sensor Network
EOFS	Environmental Observation and Forecasting System
FDS	Fixed Distributed System
FI	Future Internet
HR-WPAN	High-Rate Wireless Personal Area Network
HVAC	Heating, Ventilation and Air-Conditioning
IoT	Internet of Things
ITS	Intelligent Transport System
LR-WPAN	Low-Rate Wireless Personal Area Network
LWIM	Low Power Wireless Integrated Microsensor
MAC	Medium Access Control
MEMS	Micro-Eletro-Mechanical Systems
MGM	Maximal Gain Messaging
MILP	Mixed Integer Linear Programming
MIT	Massachussets Insitute of Technology
MLE	Minimum Lifetime Estimation
NFC	Near Field Communications
NOAA	National Oceanographic and Atmospheric Administration
PA	Power Amplifier

QoS	Quality of Service
REMBASS	Remote Battlefield Sensor System
RFID	Radio-Frequency IDentification
SDAC	Sense, Decide, Act, Communicate
SOSUS	Sound Surveillance System
TAN	Task Allocation Negotiation
TDMA	Time Division Multiple Access
TRSS	Tactical REmote Sensor System
UCLA	University of California at Los Angeles
UGS	Unattended Ground Sensor
USN	Ubiquitous Sensor Network
VLSI	Very Large Scale Integration
WAMPS	Wireless Sensor Network Air Pollution Monitoring System
WLU	Wonderful Life Utility
WSAN	Wireless Sensor and Actuator Network
WSN	Wireless Sensor Network

Bibliography

[1] W. Dargie and C. Poellabauer, *Fundamentals of Wireless Sensor Networks: Theory and Practice*. Wiley, 2010.

[2] C.-Y. Chong and S. P. Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, 2003.

[3] C. Nishimura and D. Conlon, "IUSS dual use: Monitoring whales and earthquakes using sosus," *Marine Technology Society Journal*, vol. 27, pp. 13–13, 1994.

[4] M. O'Driscoll and J. Krill, "Cooperative engagement capability," *Naval engineers journal*, vol. 109, no. 2, pp. 43–57, 1997.

[5] H. Marandola, J. Mollo, and P. Walter, "Rembass-ii: the status and evolution of the army's unattended ground sensor system," in *Proceedings of SPIE*, vol. 4743, no. 99-107, 2002, p. 148.

[6] D. Steere, A. Baptista, D. McNamee, C. Pu, and J. Walpole, "Research challenges in environmental observation and forecasting systems," in *Proceedings of the 6th annual International ACM Conference on Mobile Computing and Networking*, 2000, pp. 292–299.

[7] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh, "Deploying a wireless sensor network on an active volcano," *Internet Computing, IEEE*, vol. 10, no. 2, pp. 18–25, 2006.

[8] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "Sensorscope: Application-specific sensor network for environmental monitoring," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 2, p. 17, 2010.

[9] K. Khedo, R. Perseedoss, and A. Mungur, "A wireless sensor network air pollution monitoring system," *International Journal Of Wireless & Mobile Network(IJWMN) 2010*, vol. 2, no. 2, pp. 31–45, 2010.

[10] N. Berry, J. Davis, T. Ko, R. Kyker, R. Pate, D. Stark, R. Stinnett, J. Baker, A. Cushner, C. Van Dyke *et al.*, "Sense, decide, act, communicate (SDAC): Next generation of smart sensor systems," in *Proceedings of SPIE*, vol. 5417, 2004, p. 371.

[11] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Wireless sensor networks for structural health monitoring," in *Proceedings of the 4th International ACM Conference on Embedded Networked Sensor Systems*, 2006, pp. 427–428.

[12] V. Gungor and G. Hancke, "Industrial wireless sensor networks: Challenges, design principles, and technical approaches," *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 10, pp. 4258–4265, 2009.

[13] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis, "Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*. ACM, 2005, pp. 64–75.

[14] S. Coleri, S. Cheung, and P. Varaiya, "Sensor networks for monitoring traffic," in *Allerton conference on communication, control and computing*, 2004, pp. 32–40.

[15] S. Biswas, R. Tatchikou, and F. Dion, "Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety," *Communications Magazine, IEEE*, vol. 44, no. 1, pp. 74–82, 2006.

[16] V. Tang, Y. Zheng, and J. Cao, "An intelligent car park management system based on wireless sensor networks," in *Pervasive Computing and Applications, 2006 1st International Symposium on*, 2006, pp. 65–70.

[17] D. Tacconi, D. Miorandi, I. Carreras, F. Chiti, and R. Fantacci, "Using wireless sensor networks to support intelligent transportation systems," *Ad Hoc Networks*, vol. 8, no. 5, pp. 462–473, 2010.

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

[19] I. Howitt and J. Wang, "Energy balanced chain in distributed sensor networks," in *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, vol. 3, 2004, pp. 1721–1726.

[20] 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.

[21] S. Sengupta, S. Das, M. Nasir, and B. Panigrahi, "Multi-objective node deployment in wsns: In search of an optimal trade-off among coverage, lifetime, energy consumption, and connectivity," *Engineering Applications of Artificial Intelligence*, 2012.

[22] S. Singh, M. Woo, and C. S. Raghavendra, "Power-aware routing in mobile ad hoc networks," in *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, ser. MobiCom '98, 1998, pp. 181–190.

[23] J.-H. Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, 2000, pp. 22 –31.

[24] 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.

[25] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, 2000, p. 10pp.

[26] D. Wei, Y. Jin, S. Vural, K. Moessner, and R. Tafazolli, "An energy-efficient clustering solution for wireless sensor networks," *Wireless Communications, IEEE Transactions on*, vol. 10, no. 11, pp. 3973–3983, 2011.

[27] S. Bandyopadhyay and E. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3, 2003, pp. 1713–1723.

[28] M. Zorzi, A. Gluhak, S. Lange, and A. Bassi, "From today's INTRAnet of things to a future INTERnet of things: a wireless- and mobility-related view," *Wireless Communications, IEEE*, vol. 17, no. 6, pp. 44 –51, 2010.

[29] J. Kahn, R. Katz, and K. Pister, "Next century challenges: mobile networking for smart dust," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, 1999, pp. 271–278.

[30] G. Pottie and W. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[31] F. McQuiston, J. Parker, and J. Spitler, *Heating, ventilating, and air conditioning: analysis and design*. Wiley, 2010, vol. 6.

[32] N. Bressan, L. Bazzaco, N. Bui, P. Casari, L. Vangelista, and M. Zorzi, "The deployment of a smart monitoring system using wireless sensor and actuator networks," in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, 2010, pp. 49–54.

[33] C. Persson, G. Picard, and F. Ramparany, "A multi-agent organization for the governance of machine-to-machine systems," in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on*, vol. 2, 2011, pp. 421–424.

[34] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *Communications Magazine, IEEE*, vol. 40, no. 8, pp. 102–114, 2002.

[35] IEEE Standards Association, "IEEE 802.11™-2012." [Online]. Available: standards.ieee.org/about/get/802/802.11.html

[36] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[37] E. Callaway, P. Gorday, L. Hester, J. Gutierrez, M. Naeve, B. Heile, and V. Bahl, "Home networking with ieee 802.15. 4: a developing standard for low-rate wireless personal area networks," *Communications Magazine, IEEE*, vol. 40, no. 8, pp. 70–77, 2002.

[38] IEEE Standards Association, "Ieee 802.15.3-2003." [Online]. Available: http://standards.ieee.org/findstds/standard/802.15.3-2003.html

[39] Z. Alliance, "Zigbee specification," *ZigBee document 053474r06, version*, vol. 1, p. 378, 2006.

[40] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, and M. Nixon, "Wirelesshart: Applying wireless technology in real-time industrial process control," in *Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS'08. IEEE*, 2008, pp. 377–386.

[41] HART Communication Foundation, "WirelessHART." [Online]. Available: http://www.hartcomm.org/protocol/wihart/wireless_overview.html

[42] ISA100, Wireless Systems for Automation, "ISA100.11a." [Online]. Available: http://www.isa.org/MSTemplate.cfm?MicrositeID=1134&CommitteeID=6891

[43] Z. Shelby and C. Bormann, *6LoWPAN: the wireless embedded internet*. Wiley, 2011, vol. 43.

[44] 6LoWPAN Working Group, "6LoWPAN." [Online]. Available: http://tools.ietf.org/wg/6lowpan/

[45] Nokia, "WiBree." [Online]. Available: http://www.developer.nokia.com/Community/Wiki/Wibree

[46] *Proceedings of the Distributed Sensor Networks Workshop*. Pittsburgh, USA: Department of Computer Science, Carnegy Mellon University, 1978.

[47] R. Sproull and D. Cohen, "High-level protocols," *Proceedings of the IEEE*, vol. 66, no. 11, pp. 1371–1386, 1978.

[48] C. Myers, A. Oppenheim, R. Davis, and W. Dove, "Knowledge based speech analysis and enhancement," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84.*, vol. 9, 1984, pp. 162–165.

[49] Y. Bar-Shalom, "Multitarget-multisensor tracking: advanced applications," *Norwood, MA, Artech House, 1990, 391 p.*, vol. 1, 1990.

[50] C. on Distributed Remote Sensing for Naval Undersea Warfare and C. o. D. R. S. f. N. U. W. S. National Research Council (US), *Distributed Remote Sensing for Naval Undersea Warfare: Abbreviated Version*. National Academy Press, 2007.

[51] K. Bult, A. Burstein, D. Chang, M. Dong, M. Fielding, E. Kruglick, J. Ho, F. Lin, T. Lin, W. Kaiser *et al.*, "Low power systems for wireless microsensors," in *Low Power Electronics and Design, 1996., International Symposium on*, 1996, pp. 17–21.

[52] R. Shah and J. Rabaey, "Energy aware routing for low energy ad hoc sensor networks," in *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, vol. 1, 2002, pp. 350–355.

[53] E. Shih, S. Cho, F. Lee, B. Calhoun, and A. Chandrakasan, "Design considerations for energy-efficient radios in wireless microsensor networks," *The Journal of VLSI Signal Processing*, vol. 37, no. 1, pp. 77–94, 2004.

[54] M. Presser and A. Gluhak, "The internet of things: Connecting the real world with the digital world," *EURESCOM mess@ ge–The Magazine for Telecom Insiders*, vol. 2, 2009.

[55] J. Bernat Vercher, S. Perez Marin, A. Gonzalez Lucas, R. Sorribas Mollon, L. Villarrubia Grande, L. Campoy Cervera, and L. Hernández Gómez, "Ubiquitous sensor networks in ims: an ambient intelligence telco platform," 2008.

[56] Z. Tafa, "Ubiquitous sensor networks," *Application and Multidisciplinary Aspects of Wireless Sensor Networks*, pp. 267–268, 2011.

[57] H. Schaffers, N. Komninos, M. Pallot, B. Trousse, M. Nilsson, and A. Oliveira, "Smart cities and the future internet: Towards cooperation frameworks for open innovation," *The future internet*, pp. 431–446, 2011.

[58] A. Tabar, A. Keshavarz, and H. Aghajan, "Smart home care network using sensor fusion and distributed vision-based reasoning," in *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*. ACM, 2006, pp. 145–154.

[59] D. Han and J. Lim, "Design and implementation of smart home energy management systems based on zigbee," *Consumer Electronics, IEEE Transactions on*, vol. 56, no. 3, pp. 1417–1425, 2010.

[60] M. Skubic, G. Alexander, M. Popescu, M. Rantz, and J. Keller, "A smart home application to eldercare: Current status and lessons learned," *Technology and Health Care*, vol. 17, no. 3, pp. 183–201, 2009.

[61] M. Erol-Kantarci and H. Mouftah, "Wireless sensor networks for cost-efficient residential energy management in the smart grid," *Smart Grid, IEEE Transactions on*, vol. 2, no. 2, pp. 314–325, 2011.

[62] J. Park and S. Sahni, "An online heuristic for maximum lifetime routing in wireless sensor networks," *Computers, IEEE Transactions on*, vol. 55, no. 8, pp. 1048 –1056, 2006.

[63] A. Chamam and S. Pierre, "On the planning of wireless sensor networks: energy-efficient clustering under the joint routing and coverage constraint," *Mobile Computing, IEEE Transactions on*, vol. 8, no. 8, pp. 1077–1086, 2009.

[64] F. Liu, C. Tsui, and Y. Zhang, "Joint routing and sleep scheduling for lifetime maximization of wireless sensor networks," *Wireless Communications, IEEE Transactions on*, vol. 9, no. 7, pp. 2258–2267, 2010.

[65] 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.

[66] 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.

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

[68] 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.

[69] J. Zhu, J. Li, and H. Gao, "Tasks allocation for real-time applications in heterogeneous sensor networks for energy minimization," in *Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, vol. 2, 2007, pp. 20–25.

[70] S. Abdelhak, C. Gurram, S. Ghosh, and M. Bayoumi, "Energy-balancing task allocation on wireless sensor networks for extending the lifetime," in *Circuits and Systems (MWSCAS), 2010 53rd IEEE International Midwest Symposium on*, 2010, pp. 781 –784.

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

[72] Y. Shen and H. Ju, "Energy-efficient task assignment based on entropy theory and particle swarm optimization algorithm for wireless sensor networks," in *Green Computing and Communications (GreenCom), 2011 IEEE/ACM International Conference on*, 2011, pp. 120 –123.

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

[74] M. Conforti, G. Cornuéjols, and G. Zambelli, "Polyhedral approaches to mixed integer linear programming," *50 Years of Integer Programming 1958-2008*, pp. 343–385, 2010.

[75] ILOG, "Cplex user's manual," 2008. [Online]. Available: http://yalma.fime.uanl.mx/cplex11-manual/Content/Optimization/Documentation/CPLEX/_pubskel/XPlatform/data.html

[76] FICO, "Xpress optimization suite." [Online]. Available: http://www.fico.com/en/Products/DMTools/Pages/FICO-Xpress-Optimization-Suite.aspx

[77] 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, 2002.

[78] Chipcon, "Smartrf cc2420 datasheet," 2.4GHz IEEE 802.15.4/ZigBee-ready RF transceiver.

[79] D. Shah, "Gossip algorithms," *Foundations and Trends® in Networking*, vol. 3, no. 1, pp. 1–125, 2009.

[80] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2508–2530, 2006.

[81] M. Franceschelli and A. Gasparri, "On agreement problems with gossip algorithms in absence of common reference frames," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 4481–4486.

[82] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Trans. on Computer Systems*, vol. 23 (3), pp. 219–252, 2005.

[83] M. Franceschelli, A. Giua, and C. Seatzu, "Load balancing over heterogeneous networks with gossip-based algorithms," in *2009 American Control Conference*, 2009.

[84] J. Lu, C. Tang, P. Regier, and T. Bow, "A gossip algorithm for convex consensus optimization over networks," in *American Control Conference (ACC), 2010*, 2010, pp. 301–308.

[85] R. Van Renesse, Y. Minsky, and M. Hayden, "A gossip-style failure detection service," in *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, 2009, pp. 55–70.

[86] Z. Cheng, M. Perillo, and W. Heinzelman, "General network lifetime and cost models for evaluating sensor network deployment strategies," *IEEE Transactions on Mobile Computing*, vol. 7, no. 4, pp. 484–497, April 2008.

[87] V. Pilloni, P. Navaratnam, S. Vural, L. Atzori, and R. Tafazolli, "Cooperative task assignment for distributed deployment of applications in wsns," in *Communications (ICC), 2013 IEEE International Conference on*, 2013, Accepted.

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

[89] 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.

[90] Q. Li, J. Aslam, and D. Rus, "Hierarchical power-aware routing in sensor networks," in *In Proceedings of the DIMACS Workshop on Pervasive Networking*, 2001.

[91] 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.

[92] 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.

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

[94] W. Zhang, G. Wang, Z. Xing, and L. Wittenburg, "Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks," *Artificial Intelligence*, vol. 161, no. 1-2, pp. 55 – 87, 2005.

[95] 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.

[96] J. P. Pearce and M. Tambe, "Quality guarantees on k-optimal solutions for distributed constraint optimization problems," in *Proceedings of the 20th international joint conference on Artifical intelligence*, 2007, pp. 1446–1451.

[97] E. Lawler and D. Wood, "Branch-and-bound methods: A survey," *Operations research*, vol. 14, no. 4, pp. 699–719, 1966.

[98] Y. Tian and E. Ekici, "Cross-layer collaborative in-network processing in multihop wireless sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 6, no. 3, pp. 297–310, 2007.