POLITECNICO DI MILANO
Dipartimento di Elettronica e Informazione
DOTTORATO DI RICERCA IN INGEGNERIA DELL'INFORMAZIONE

ADVANCED METHODOLOGIES AND TECHNIQUES FOR THE OPTIMAL DESIGN OF WIRELESS SENSOR NETWORKS

Doctoral Dissertation of:
Paolo Roberto Grassi

Advisor:
Prof. Donatella Sciuto

Tutor:
Prof. Cesare Alippi

Supervisor of the Doctoral Program:
Prof. Carlo Ettore Fiorini

2012 – XXV

POLITECNICO DI MILANO
Dipartimento di Elettronica e Informazione
Piazza Leonardo da Vinci, 32 I-20133 — Milano

Ringraziamenti

Questo dottorato è stato una sfida, una prova su cui solo una come *Donatella Sciuto* poteva scommerterci; sono e le sarò eternamente grato della fiducia e della grande opportunità che mi ha dato. La sua competenza, la sua capacità di guardare oltre l'apparente aspetto delle cose mi ha non solo insegnato un mestiere, ma una importante lezione di vita. Grazie anche per aver coronato un sogno che avevo nel cassetto: insegnare ad un corso al Politecnico di Milano; questo significa moltissimo per me.

Il secondo sentito ringraziamento è per *David Atienza*, che mi ha ospitato all'*ESL* presso l'*École Polytechnique Fédérale de Lausanne*; Losanna è una città bellissima e le esperienze che ho avuto mi rimarranno nel cuore per sempre, in particolare *MatraCAM*, che non c'entra nulla con questo lavoro di tesi, non mi ha fatto pubblicare nulla, ma il suo sviluppo mi ha insegnato dove finisce la pazienza ed inizia la passione.

Un ringraziamento speciale va a mia moglie *Sara*, per aver sopportato amorevolmente sia le trasferte, che le lunghe nottate di lavoro in ufficio, e per aver sempre scommesso su di me, anche quando le cose non andavano per il verso giusto. Un dottorato che ha anche segnato l'inizio della nostra vita assieme, cominciata con poche cose e terminato con un bellissimo matrimonio che durerà per sempre, come il mio amore per lei.

Tantissima gratitudine va poi a tutte le persone che hanno condiviso con me i momenti di gioia e (pura) disperazione: *Ivan*, per aver sempre tenuto duro anche davanti ad insormontabili difficoltà, al *Nacci*, per aver sempre trovato il modo di farmi sorridere, a *Vins*, per i preziosi consigli sugli articoli, a *Christian*, per (forse) una birra di troppo, a *Daniele*, per tutte le stimolanti discussioni a pranzo, a *Marco*, di cui ho sempre ammirato dedizione, attenzione e passione, ad *Antonio*, il cui naturale pessimismo si è sempre scontrato con il mio (irragionevole) costante ottimismo, a *Chiara*, che mi ha insegnato cosa vuol dire *vero* divertimento, a *Luigi*, che mi ha fatto capire che un buon colpo di fortuna val più di due dottorati, a tutto il dipartimento e all'*ESL* di Losanna per avermi regalato momenti indimenticabili.

Contents

List of Figures

List of Tables

1 Introduction

In the last two decades, technology advances in chip miniaturization, energy consumption and wireless communication allow the development of revolutionary applications in fields like wearable and ubiquitous computing. The term ***ubiquitous computing*** refers to distributed technologies and applications designed to disappear in the environment, allowing the user to unconsciously interact with it. Ubiquitous computing requires small and smart devices deployed in the field of interest with the purpose of sensing valuable physical variables and interact with the users. A large amount of applications has been envisioned to this future in the field of tele-medicine [141], child care [145], environmental monitoring [104], etc.

In 2001, the IST Advisory Group (ISTAG) published a white paper that describes what living with ***Ambient Intelligence*** **(AmI)** might be like for ordinary people in 2010 [53]. This document includes four user-centric scenarios that envision what technology could do in the future and what could be the role of the user with respect to the Information Technology. Although the paper does not accurately describe the ordinary life of current days, the paper has been used as **guideline for research** and development of new devices, methodologies and techniques.

In these days, there is a considerable interest in making our city *smarter* under several point of view such as urban monitoring [93], pollution [152] and various social services [114]. ***Smart Cities*** are a long-term project and the research community is working hard toward the development of technologies that will enable *Smart Cities* to become reality in a near future.

All these scenarios do not define any specific technology rather they are focused on user-machine interaction and applications. Very often they overlap in many aspects and thus technologies required for AmI could be useful for *Smart Cities* and so on. In particular, an element is **heterogeneity**, since different systems will interact each other to provide the user the required service. In *AmI*, devices like television, washing machine, heating system, security cameras, etc. will actively collaborate. In *Smart Cities*, a traffic light could need to communicate with a meteorological station. In *Ubiquitous Computing*, remarkably different devices with substantial hardware differences must communicate each other.

One of the **enabling technology** is **Wireless Sensor Networks** (WSNs), networks of tiny devices that cooperatively sense and act the environment in which they are

deployed. Next Section provides a short introduction on this topic.

1.1 Wireless Sensor Networks

A WSN is an ad-hoc network composed of tiny devices with limited energy and computational resources equipped with sensors to gather physical measures from the environment. In 1999, Wireless Sensor Networks (WSNs) have been considered as one of the **most important** technologies for the twenty-first century [17]. A decade of research and applications proved the truth of such statement and their potential in next generation digital systems.

In the last decade, a lot of research effort has been spent on Wireless Sensor Networks (WSNs), and many architectures [164], protocols [24], programming techniques [115] have been developed. Thanks to this research, today, complex and innovative applications can be developed in challenging application fields like medical [75] [127] or environmental monitoring [104] [108].

The increasing complexity of Wireless Sensor Networks (WSNs) is leading towards the deployment of **sophisticated networked systems**, and the optimal design of WSNs can be a very difficult task in case constraints and requirements are strong. A WSN is composed of several nodes that communicate among each other through a wireless channel: these nodes are typically battery-powered, and equipped with low-performance processors and small memories in order to reduce the power requirements. A common WSN node comprises five main components [130]: a **processing unit** (microcontroller, processor, FPGA, ...), **memories** (DRAM, SRAM, Flash, ...), **sensors** and **actuators**, **multiple communication layers** (physical radio, MAC, Routing, ...) and a **power supply** (external power supply, batteries, solar cells, ...).

During the design phase, the cooperation of all these components must be combined to identify the configuration that best fits the **design objectives**. The rising complexity of WSNs design is also due to the combination of general-purpose architectures (which offer flexibility, but require an optimal configuration in order to behave in an energy-efficient way) with *ad-hoc* radio, MAC and routing layers. The combination of different layers and the large number of configurable hardware and software parameters often generates an **extremely large design space**, which requires a powerful CAD algorithm to carry out the exploration.

In the last years, wireless sensor networks (WSNs) are becoming a well-established reality in many different domains, including military applications, environment control, industrial supervision, health monitoring [164] [28] and environmental monitoring [104]. The transmission range of wireless devices can vary from few meters to several kilometers, according to application's requirements and energy availability. Section 2.2 provides a deeper analysis of the application fields in WSNs.

Once retrieved, measures are elaborated and sent over the wireless channel to a *sink*, where data is stored and used to monitor activities of the area of interest. WSN nodes usually operate in **hostile environments** with **limited energy resources** ([104, 108]), constrained by the battery capacity, and thus the problem of achieving low power consumption has become one of the main research focuses of research over the last years [164]. WSN nodes are interconnected with low-power wireless radio devices

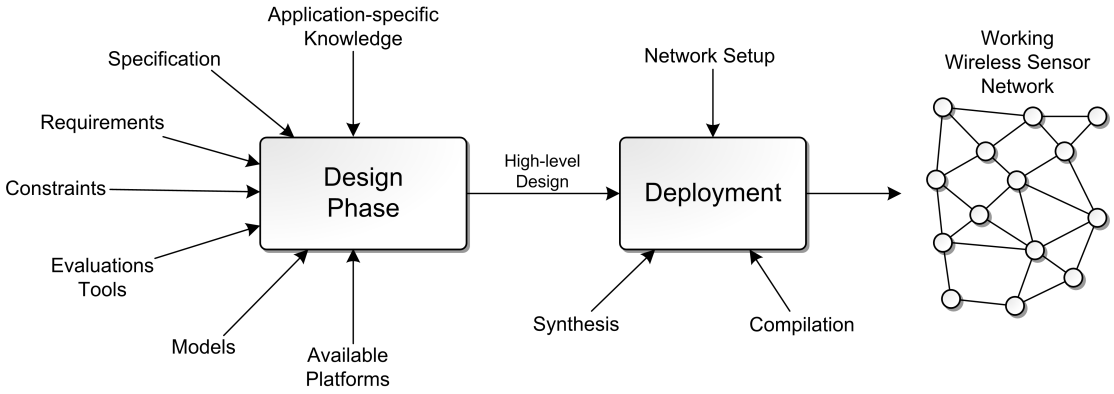


Figure 1.1: *Design Process of Wireless Sensor Network*

which create an ad-hoc network infrastructure able to route data from sensing devices to the sink.

In order to deal with the specific requirements of a given application domain, a WSN has to meet certain performance requirements as well as to guarantee a sufficient lifetime, which are often conflicting goals. The right tradeoff between these two objectives, as well as the prevention of undesired behaviors such as unbalanced performance among the different nodes of the WSN, can be guaranteed by accurately evaluating the network configurations during the design phase. In order to help the designer during the energy-performance tradeoff analysis, many **Design Space Exploration (DSE)** techniques for WSNs have been proposed in the literature [163] [118], and most of the classic optimization algorithms can also be adapted to WSNs with a low effort. However, providing such algorithms with an accurate system-level estimation of the WSN performance is still an open problem, and it is necessary to guide the DSE algorithm to the detection of the Pareto-optimal network configurations.

1.2 Motivations and Rationale

Nowadays, Wireless Sensor Network's design requires experts from **several application fields** such as computer science, electronics, telecommunication, digital signal processing and application-specific competences (medicine, geology, biology, etc.). A collaboration between experts in these fields is required to guarantee an **optimal design** that respects given constraints and meet desired requirements.

This thesis presents a comprehensive study on the design of Wireless Sensor Networks including hardware platform design, network optimization and software partition. Moreover, adaptive techniques are proposed to deal with online real-time events such as interferences.

The **main contribution** of this thesis is a **general-purpose design flow for WSN** that defines the set and the sequence of processes to follow to obtain the specified design. The proposed design flow is a guideline for the development of **automated design tools** and **design frameworks**. An overview of the design process of WSNs is given in Figure 1.1; the design phase takes several inputs such as the application specification, requirements, constraints, etc. The output is an high level definition of the final design;

it specifies the position of the nodes, their hardware and network configuration and the software that will run on the nodes. The output is at high-level since cannot be directly deployed on the field, thus final operations like compilation, synthesis and network setup are still to be done. Software, for instance, is defined in high-level languages, that will be converted into machine-readable specification during the compilation process. The objective of this thesis is to define how the design process should be done, what can be done automatically and which information are required at each step of the design flow.

The applicability and effectiveness of the proposed design flow and optimization techniques has been verified and tested through the implementation of a **design framework**. The vision of the proposed framework is to give the designer a **powerful tool** to design WSNs. It includes:

- Automated optimization of the given design;
- Pareto-frontier analysis to detect desired trade-off in a multi-objective scenario;
- Intuitive interface that allow manual design and test of WSNs.

Although the proposed design framework is still in an early development phase, it is able to provide high-level information to the designer in order to speedup the design process, reducing costs and time-to-market.

In addition to design-time (offline) analysis and optimization, this thesis presents two techniques for **online real-time adaptivity** (Chapter 5). This Chapter does not aim to provide a comprehensive study on online optimization, but show the reader the limits of design-time optimizations and the advantages of online real-time approaches. Please note that design-time optimization are usually more effective than online optimization for two aspects: the ability to define the design, in case of non optimality, in any aspect (protocols, devices, etc.), and the amount of information during evaluation. However, online real-time optimizations are able to deal with stochastic processes such as faults or interferences effectively.

Design-time and online optimizations must be both considered during the design of a WSN. **Reliable and cost-effective design solutions** are important factors to ensure success and diffusion of WSN, thus efficient design tools to support the designer in this phase is of extreme important.

1.3 Thesis Organization

This thesis work is organized in five core Chapters before the concluding remarks as shown in Figure 1.2. Chapter 2 presents the background and preliminary information required to understand the work. It includes a general model for WSN nodes, an overview of WSN applications, considerations about design methodologies and evaluation techniques. Chapter 3 illustrates a set of commercial and open source hardware platforms for WSNs, and includes a proposed sensing/processing architecture equipped with low-power FPGAs. This is intended to provide the user a clear understanding about kind of target platforms. Chapter 4 introduces and details the proposed design flow. This is the core chapter and includes the main contributions of this thesis such

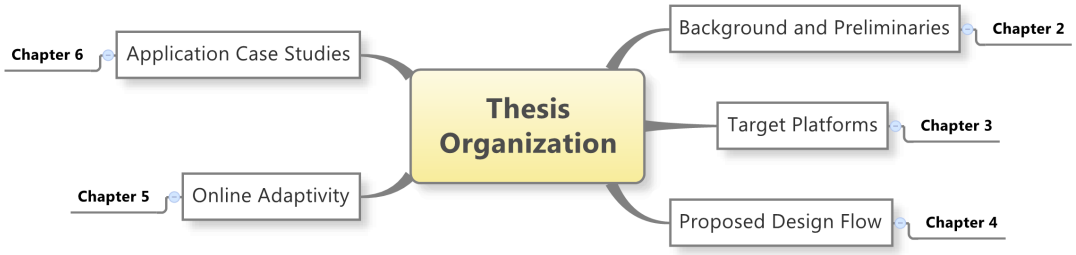


Figure 1.2: *Thesis Organization*

as the proposed design flow, two models and a novel technique for the design space exploration. Chapter 5 copes with online adaptivity in WSNs and is focused on the network layer. Considering the complexity of this topic, two examples related to online adaptivity in network communications are provided. Chapter 6 presents several experimental results based on various case studies and scenarios. These application case studies show how the proposed design flow is used in the specific field. Chapter 7 concludes the thesis providing some hint for future development.

1.4 Publications

The various aspects of the research proposed in this thesis have been published and presented at international conferences. The list of papers is the following:

- **P. R. Grassi**, I. Beretta, V. Rana, D. Atienza, D. Sciuto: *Knowledge-Based Design Space Exploration of Wireless Sensor Networks*. In Proc. of International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2012), 7 - 12 October 2012, Tampere, Finland
  - This work presents a novel technique to perform design space exploration of WSNs using both models and heuristics on the top of a Markov Decision Process to perform the exploration
  - Presented in Section 4.4.3.1
- **P. R. Grassi**, D. Sciuto: *Energy-Aware FPGA-Based Architecture for Wireless Sensor Networks.* In Proc. of 15th Euromicro Conference on Digital System Design *(DSD 2012)*, 5 - 8 September 2012, Izmir, Turkey
  - An FPGA-based architecture for WSNs is presented here. The proposed architecture uses a Flash-based FPGA to allow the system to control the energy consumption of the device dynamically
  - Presented in Section 3.2
- **P. R. Grassi**, I. Beretta, V. Rana, D. Sciuto: *Tacit Consent: a Technique to Reduce Redundant Transmissions from Spatially Correlated Nodes in Wireless Sensor Networks.* In Proc. of 15th Euromicro Conference on Digital System Design *(DSD 2012)*, 5 - 8 September 2012, Izmir, Turkey

  - This paper introduces the *Tacit Consent*, a technique to reduce redundant transmissions in cluster-based sensor networks exploiting the spatial correlation of sensed data
  - Presented in Section 5.2

- F. Rincon, **P. R. Grassi**, N. Khaled, D. Atienza, D. Sciuto: *Automated Real-Time Atrial Fibrillation Detection on a Wearable Wireless Sensor Platform*. In Proc. of 34th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), 28 August - 1 September 2012, San Diego, USA
  - This paper presents an innovative solution to accurately and timely detect atrial fibrillations using resource constrained wireless sensor nodes

- I. Beretta, F. Rincon, N. Khaled, **P. R. Grassi**, V. Rana, D. Atienza: *Design Exploration of Energy-Performance Trade-Offs for Wireless Sensor Networks*. In Proc. of 49th Asia and South Pacific Design Automation Conference (ASP-DAC), 3 - 7 June 2012, San Francisco, USA
  - This work illustrates a model-based optimization framework able to identify the optimal energy-performance trade-off in WBSN systems
  - Presented in Section 4.4.2.2

- **P. R. Grassi**, V. Rana, I. Beretta, D. Sciuto: *B^2IRS: a Technique to Reduce BAN-BAN Interferences in Wireless Sensor Networks*. In Proc. of 9th International Conference on Wearable and Implantable Body Sensor Networks (BSN 2012), 9 - 12 May 2012 London, United Kingdom
  - It introduces a technique to reduce BAN-BAN interferences in IEEE 802.15.4 based networks in presence of up-to 4 co-located networks operating on the same channel
  - Presented in Section 5.1

- **P. R. Grassi**, A. Ceppi, F. Cancaré, G. Ravazzani, M. Mancini, D. Sciuto: *Automatic Identification and Placement of Measurement Stations for Hydrological Discharge Simulations at Basins Scale*, in Proc. of European Geosciences Union General Assembly (EGU 2012), 22 - 27 April 2012, Vien, Austria
  - It illustrates a technique to identify the optimal position of sensor network nodes for hydrological monitoring and forecasting

- I. Beretta, F. Rincon, N. Khaled, **P. R. Grassi**, V. Rana, D. Atienza, D. Sciuto: *Model-Based Design for Wireless Body Sensor Network Nodes*. in Proc. of 13th Latin American Test Workshop (LATW 2012), 10-13 April 2012, Quito, Ecuador
  - This paper presents a model-based optimization framework and a multi-objective exploration algorithm for Wireless Body Sensor Networks.
  - Presented in Section 4.4.2.1

2

Background and Preliminaries

A Wireless Sensor Network (WSN) is designed to gather physical measures from the environment where it has been deployed. These information can be collected in a central node (a *sink*) for further analysis, or used by the network to actuate the environment. Transmission ranges of WSNs can vary from few meters to some kilometers, according to the application's needs and power requirements. The typical communication medium is the air, but other mediums have been successfully used (water [25], skin [65], etc...). In all of these cases, a WSN is composed of several nodes whose communications are wireless and that are able to sense (and actuate) the environment of interest.

The type of sensed variable depends on the application needs and can vary in both sampling throughput and accuracy. A WSN must be tailored to the specific application to deliver the correct data effectively. It is not possible to design a general purpose system that works effectively for all kind of applications; to guarantee efficient architectures, problem-specific design approach must be adopted. If this concept is true even for other fields of computer science, in WSN field it must be taken into consideration: design choices for implantable body sensor networks are extremely different to the design choices of a WSN for structural monitoring.

The design problem of a WSN consists in identifying the position of the nodes in the space, their hardware/network configuration and the software definition. Only a perfect combination of the components ensures that the network will perform its tasks in a correct and efficient way. The identification of the optimal design choices is a very complex task that require experts in both electronic, telecommunication and computer science. Moreover, to implement efficient applications, experts on the target application field are required (medical, geosciences, etc...). Unfortunately, few designers have these skills, since they are more focused on the electronic or telecommunication components rather than computer science or vice versa, thus teams with heterogeneous and complementary expertises are usually required. In addition, even though the team members have the required expertises, the identification of the optimal design can be extremely complex. For such a reason, Computer Aided Design (CAD) automated tools for the design of WSN are required. To develop design methodologies and automated techniques for WSN design, it is crucial to define the WSN field and its design problem.

This Chapter provides some basic concepts about the design of WSNs that will help the reader to better appreciate the thesis work. Section 2.1 presents a general structure

of a WSN. Section 2.2 classifies the WSN applications according to their design requirements and constraints. In order to provide a wide view of the design problem a set of common parameters and metrics is introduced. Section 2.3 illustrates the theoretical complexity of the design of WSN. Section 2.4 presents the most common Hardware/-Software co-design methodologies for embedded systems. Design evaluation is an important part of the design flow; Section 2.5 shows three evaluation techniques for WSN highlighting pros and cons.

2.1 General Structure of a Wireless Sensor Network

A Wireless Sensor Network is composed of several nodes that communicate to accomplish a common task. Each device processes sensorial data, uses actuators, receives and transmits through the network interface. Most of the WSN nodes are battery powered in order to allow free movements or the deployment in hostile places. Moreover, energy harvesting helps the devices to extend their lifetime.

The architecture of a WSN node strictly depends on the target application, thus nodes can be significantly different according to their application's needs; they can differ in sensor types, processing unit, operating system used, etc. However, WSN nodes can be modeled in a generic way, in order to have a common mathematical structure that could help the designer to interpret experimental results and compare alternative designs. This Section introduces and present this mathematical abstraction. Dot notation $x.y$, widely used in this thesis, indicates that the feature y belongs to the component x. If f_{sam} indicates the sampling frequency of a sensor, $s_1.f_{sam}$ indicates the sampling frequency of sensor s_1.

2.1.1 Generic Node Models

A WSN node δ is a computing device composed by:

- Processing unit (CPU, ASIC, FPGA,...)
- Network interfaces and components (network layer, mac layer and physical layer [radio, optical, ...])
- Sensor(s) (temperature, vibration,...) *(optional)*
- Actuator(s) (motors, switches, ...) *(optional)*
- Power Unit (batteries, recharge circuit, harvesting, ...)

The proposed model, illustrated in Figure 2.1, is a general-purpose model able to describe a large variety of architectures. This model describes the relationships among various layers of a WSN node, from sensors and actuators to network management. From the top of the figure, sensed data, gathered by sensors ($s_i \in S$) and filtered by custom filters ($\phi_j \in \Phi$), are transmitted to the processing unit, which hosts applications and manages memories. The processing unit, that is the center of the node, elaborates both (filtered) sensorial data (F_k) and network packets (R_x^a). Then applications transmit data to the network (T_x^a) and to the actuators (C_k). Network layer is decomposed in three sub-layers: Network (NET), MAC and Physical (PHY). Network layer is responsible for routing and transport (end-to-end connectivity), MAC layer ensures

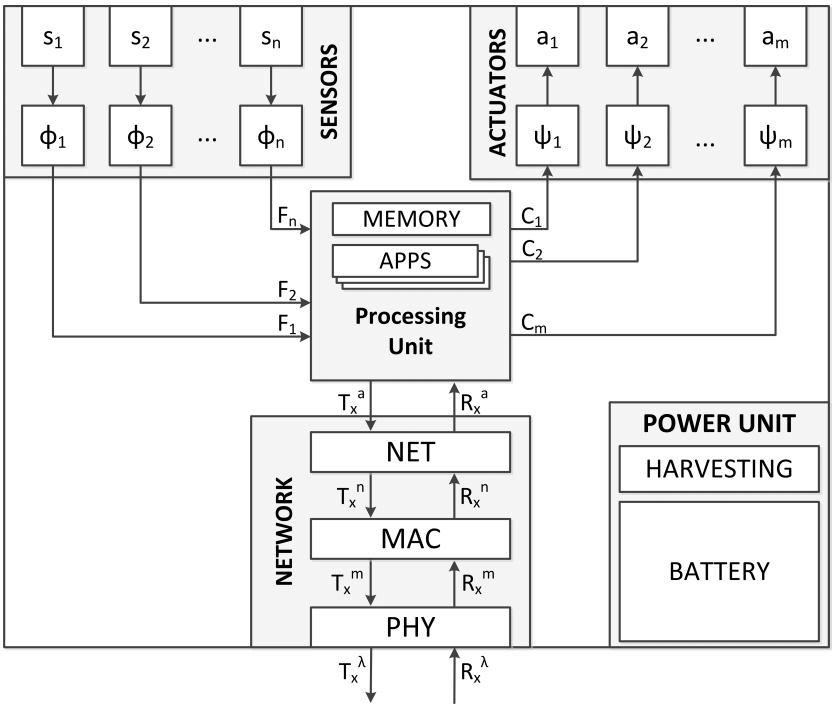


Figure 2.1: *Model of the Node*

synchronization and non-conflicting communications among co-located nodes, and the Physical layer provides bit-to-bit transmissions over the communication medium. The proposed model depicts only the data exchange among components, but it does not constrain their implementation to specific technologies or systems such as hardware, software, operating systems, etc. This leaves the designer with a degree of freedom to decide how to implement the functionalities, and provides a high level model of the node, that will be useful to compare different architectures.

Batteries and energy harvesting are modeled in a black box (power unit) that specifies the discharge model of the batteries, the harvesting capabilities, faults, etc. Similarly to other components, the power unit has been left as generic as possible to avoid imposing any constraint to its implementation.

PROCESSING

In a WSN node, processing is performed in three distinct components:

- **Sensor Filters** (Φ) **and Actuator Controllers** (Ψ): performed after the sensors operations and before the actuators activation, such processing is dedicated to the filtering and controlling of sensors and actuators. Can be either implemented with dedicated hardware (DSP, FPGA,...) or using software routines;
- **Network Layer**: the management of data in the network is one of the most important aspects of a WSN. The network layer routes packets through the network interface, checks and identifies the topology of the network and so on. Such operations usually require dedicated hardware or software units. The model proposed in this work decouples network from application data in order to effectively ex-

press the overhead of networking in WSN. In this model, all the communications among network layer have been explicited: application data (T_x^a,R_x^a), network packets (T_x^n,R_x^n), MAC packets (T_x^m,R_x^m) and radio transmissions (T_x^λ,R_x^λ). This set of parameters allow further analysis on transmitted and received packets such as, for instance, the overhead of the network protocol ($T - T_x^d$);

- **Processing Unit**: located in the conjunction of sensors, actuators and network interfaces, it completely controls the data in the node. It can or cannot host a tiny operating system, depending on both application needs and technical requirements. The Processing Unit handles application data coordinating sensor, actuator and network information. It can be either a micro-controller, a micro-processor, an FPGA, or any component able to cope with sensors, actuators and network data.

SENSORS AND ACTUATORS

The main purpose of a WSN is to sense the environment where it is deployed. A node $\delta \in \Delta$ is equipped with a set of sensors $\delta.S$ and actuators $\delta.A$. Each sensor ($s \in \delta.S$) is characterized by a sampling frequency ($s.f_{sam}$) and the number of bits used to represent the sampled data ($s.b_{sam}$). Thus, each sensor generates a throughput of $s.T = s.f_{sam} \cdot s.b_{sam}$, that is an important metric to estimate the filtering needs. Similarly, each actuator ($a \in \delta.A$) is defined by a refresh frequency ($a.f_{ref}$) and the number of bits used on each refresh ($a.b_{ref}$). The definition of the attributes of a sensor or an actuator may vary from model to model; what is important is the definition of the throughput the sensors/actuators generate/require.

Sensed data must be processed (filtered) before transmitting them to the network. Filtering is performed to reduce the amount of data to transmit (aggregation) or to extract relevant information (feature-extraction). Given data from a sensor at a given frequency and width, a sensor filter takes sensor data as input and gives filtered data as output. In this model, sensor filters ($\phi \in \delta.\Phi$) are described as black boxes that take sensor data as input and gives data as output. For instance, if $\phi \in \delta.\Phi$ is a filter applied to $s \in \delta.S$, it takes $s.b_{sam}$ data at a frequency of $s.f_{sam}$, and generates $\phi.b$ data at a frequency $\phi.f$. The throughput of the filters, represented with F_i, indicates the amount of information that should be processed by the application.

Similarly, actuator's data are managed by controllers (Ψ), which interpret application's data and convert it to actuator-compatible information. The more the controller is *intelligent* the less information it will require from the application. In the model, the throughput required by the controllers labeled with C_i.

Both filters and controller can be implemented in hardware or in software. Moreover, filters and controllers are modeled as independent entities; this allows the definition of a large variety of filtering architectures and techniques, such as DSPs or FPGAs (see Section 3.2). Typically, filters and controllers are performed by software tasks, but their design is extremely important to ensure power-efficient, performance-aware design.

POWER UNIT

One of the limitations and requirements of a WSN is the energy available on each node. Considering that each node is usually equipped with batteries, a model for energy consumption is needed. Many sensor nodes are able to recharge their batteries harvesting energy from the environment [26, 104]. Energy consumption and harvesting can be modeled as a time dependent function of the battery status.

Energy consumption and harvesting are defined as the derivative of the energy in time $\frac{E(t)}{\partial t}$. If $\frac{E(t)}{\partial t}$ is positive, the node is harvesting energy, while if $\frac{E(t)}{\partial t}$ is negative, the node is consuming energy. It is important to notice that if the derivative is positive it does not imply that the node is not consuming at all, but that the harvesting is more effective than consumption, and the node is recharging their batteries.

Such definitions are particularly useful to define the concept of node and network **lifetime**. For instance, the lifetime of a node ($\delta \in \Delta$) is defined as the time t_L where, $\forall t' > t_L$, $E(t') = 0$. Network lifetime has different definitions [37]: in applications that depend on every single node, the lifetime of a network can be defined as the time until the first node runs out of battery power. Alternatively, a network can be more or less fault tolerant and it can live as long as all live nodes are still connected to each other. Extracting such properties from the single-node definition is not difficult.

2.1.2 Network Models

The most natural way to define WSNs is using graphs. A graph is defined as a pair $G = (V, A)$ where V is a set whose elements are called *vertices* or nodes and A is a set of ordered pairs of vertices, called *arcs*. A graph can be either *Directed*, where arcs have directions or *Undirected*, if arcs are undirected. All the concepts related to graph theory are assumed to be known by the reader; an introduction to graph theory can be found in [160].

Due to the asymmetry and heterogeneity of nodes, WSNs are generally Directed Graphs. Previous works [144, 159] proposed different models for WSNs to define topology control, routing and connection properties. These works use the concepts of Quasi Unit Disk Graph (QUDG) and Bounded Independence Graph (BIG) to model the connectivity of sensor nodes. These models, given a set of nodes distributed in a two-dimensional space, express which node can receive a transmission from a node.

Connections In a WSN, nodes communicate through a network interface to exchange information; in such a way, connections are links between nodes. According to the definition of QUDG given by [144], nodes with Euclidean distance at most ρ for some given $\rho \in (0, 1]$ are adjacent. Pairs with a Euclidean distance greater than 1 are never in each other's transmission range. Finally, pairs with a distance between ρ and 1 may or may not be connected. Summarizing, given two nodes $\delta_1, \delta_2 \in \Delta$:

- if $d(\delta_1, \delta_2) \leq \rho$ nodes are adjacent
- if $d(\delta_1, \delta_2) > 1$ nodes are never adjacent
- if $\rho < d(\delta_1, \delta_2) \leq 1$ nodes can or cannot be adjacent according to some rules.

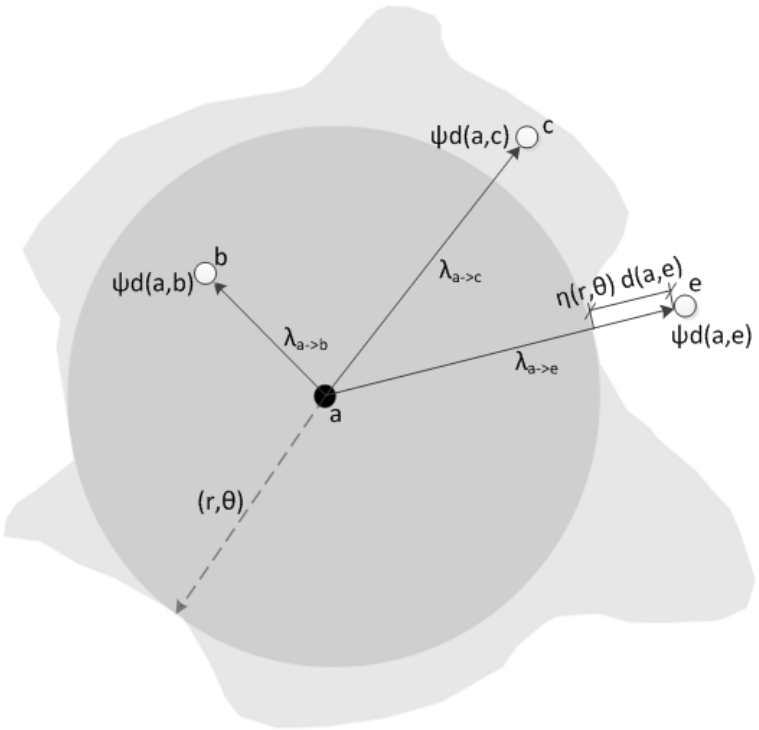


Figure 2.2: *Quasi Uniform Disk Graph*

Considering that QUDG is not useful to describe object interferences, the simplicity of the model is attractive. Independently from the function used to express the connections, considering two nodes $\delta_1, \delta_2 \in \Delta$, $\lambda_{1\to2} = (\delta_1, \delta_2)$ defines a unidirectional connection between δ_1 and δ_2 where δ_1 can communicate with δ_2 but δ_2 cannot communicate with δ_1. In most of the cases, network connections are bidirectional, implying a symmetry in the λ-relation. Figure 2.2 shows an example of QUDG. According to the definition, since $\lambda_{a\to b} < \rho$, these two nodes are adjacent, so they can communicate. $\rho < \lambda_{a\to c} < 1$, and the two nodes can communicate since c falls into the transmission area of a. As previously explained, the communication in the area $\rho < d(\delta_1, \delta_2) \leq 1$ can or cannot happen, according to some rules. One of the most common rule is to create a random shape, similarly to the one shown in the Figure. Using this concept, although $\rho < \lambda_{a\to e} < 1$, these two nodes cannot communicate.

QUDG defines only if two nodes have or not the opportunity to communicate. Within the communication links complex propagation models should be used in order to guarantee a good simulation/estimation accuracy. A good propagation model is the **Average Path Loss Model**, that estimates the average path loss between two nodes. It has been proven [170] that the lognormal shadowing model gives very accurate path loss estimates in case the distance between nodes is from few meters to one hundred meters. The following formula returns the path loss in dB as a function of the distance between the nodes and the characteristics of the channel:

$$PL(d) = PL(d_o) + 10\eta \log\left(\frac{d}{d_0}\right) + \chi_\sigma$$

Table 2.1: *Network characteristics WSNs*

Application Field	Number of Nodes	Sensed and Transmitted Variables	Network Organization
Structural Monitoring	10-1000	Vibration	Mesh
Body Sensor Networks	2-10	ECG, Temperature, EMG, EEG, ...	Star Networks
Multimedia Sensor Networks	10-50	Audio, Video	Mesh
Environmental Monitoring	30-3000	Temperature, Humidity, Brightness, ...	Push Networks
Battlefield Assistance	50-5000	Position, Speed, Audio, ...	Mesh
Underwater Monitoring	5-150	Temperature, Water Flow,	Push Networks
Domotics	10-300	Temperature, Humidity, Video, ...	Star Networks
Automotive	5-50	Temperature, Speed, Acceleration, ...	Star Networks

where $PL(d_0)$ is a measured path loss at the reference distance d_0, η is the path loss exponent and χ_σ a Gaussian noise with zero-mean and standard deviation equal to σ.

Position In WSNs, the position of the nodes defines the set of feasible topologies of the network and is also needed to define the right placement of sensors and actuators on the environment, to perform the required sensing and acting actions. Since the position is a property of nodes, we express the position of a node using the dot notation where $\delta.x$, $\delta.y$ and $\delta.z$ represent the coordinates in a 3-dimensional Euclidean Space.

2.2 Application Fields and Classification

WSNs provide useful technologies and architectures that allow several applications to be implemented. The analysis of the physical variables, sensed over the world, is extremely important to control events of interest for scientific purposes (cane toad monitoring [73], etc.), to prevent catastrophic events (such as structural health monitoring [79], landslides [27], etc.), or to monitor physiological aspects like cardiac diseases [138], posture problems [58], brain injuries [76].

Table 2.1 and 2.2 qualitatively illustrate some characteristics of WSNs with respect to their application field.

Network characteristics such as number of nodes, kind of sensed variable and network organization are described in Table 2.1. The amount of nodes in a network affects the design process in terms of complexity and optimality; the higher is the amount of nodes to be configured, the higher is the cardinality of the design space and, as a consequence, the difficulty to identify the optimal design. In Table 2.1, *network organization* has been classified as:

Table 2.2: *Power and Reliability Characteristics of WSNs*

Application Field	Power Supply	Harvesting	Fault Tolerance	Nodes Redundancy
Structural Monitoring	Battery	Solar	High	YES
Body Sensor Networks	Battery	Stress, Vibration	Critical	NO
Multimedia Sensor Networks	Cabled		Low	NO
Environmental Monitoring	Battery	Solar	Low	YES
Battlefield Assistance	Battery, Cabled	Solar, Stress, Vibration	Critical	YES
Underwater Monitoring	Battery	Water Flow	Low	YES
Domotics	Battery, Cabled		Medium	YES
Automotive	Cabled		Critical	YES

- ***Star Network***: one-hop networks. Data are gathered by a central node that receives sensorial information from all the nodes;
- ***Push Network***: network with a predefined direction of data. These networks are optimized to deliver data from sensors to sinks;
- ***Mesh***: a network of pairs with no preferred network flows.

Please note that a final design can use custom solutions for network; the listed organizations are just typical design choices.

Regarding the type of sensed variable, for the purpose of the design, what it is important is the throughput generated by each sensor node rather than its actual value. Some application fields are characterized by a large amount of nodes (*environmental monitoring* or *battlefield assistance*), but have low-frequency data such as temperatures or humidity, that get less than 1 sample per minute. On the other hand, other application fields like *multimedia sensor networks* have a lower amount of nodes (10-50), but require streaming of audio-video contents, that require broadband communications.

The identification of an optimal trade-off is extremely important in WSNs. Considering *body sensor networks*, for instance, even if the amount of nodes is considerably low (2-10), and the throughput is acceptable (ECG requires 125 samples-per-second at 24-bit precision, that is 3 kbits/sec), their optimization is very complex. In fact, power and computational limitation of these nodes makes the identification of the optimal trade-off hard to reach manually. A practical example is given in Chapter 6.

Regarding the power characteristics of WSN (See Table 2.2), the main difference concerns the use of batteries or cabled power. In case cabled power is available, it is strongly suggested to use it since it effectively reduces the faults due to battery depletion. However, in some cases, cabled power is not available. For instance, implantable *body sensor networks* require small batteries that cannot be recharged externally as well as *environmental monitoring,* where nodes are deployed in hostile places with no cabled sources available. In some cases, cabled power is available in some part of the network

(domotics and battlefield assistance), while the other nodes are powered by batteries. In domotics, for instance, nodes can be battery powered for economic (cost of cabling) and aesthetic needs (no visible cables).

In case batteries are used, an effective way to increase lifetime of the network is by harvesting energy from the environment. Energy harvesting is an open and very active research field [109] and promises to be an enabling technology for many applications. Nowadays, the most effective harvesting source is solar, that can be used for *structural monitoring*, *environmental monitoring* and in *battlefields*. In case sun is not present (underwater) or solar panels cannot be used (body sensor networks), other techniques such as stress, vibration or water current can be used instead.

Reliability of WSN varies according to the final application fields. Some of them are life-critical, while others just collect data from the environment. In the first case, reliability is critical, and must be tackled effectively. Redundancy is the most common way to increase the reliability of a system; if it is always possible to have redundancy on sensors, in some cases, nodes' redundancy cannot be applied for practical reasons. In *body sensor networks*, for instance, it is not always possible to place several nodes on the same spot to perform independent measurements.

Reliability in WSN means that the system must work in **standard scenarios**. Standard conditions are a set of possible scenarios that have been identified during design-time. Examples of standard scenarios are:

- Presence of up to three co-located networks operating on the same frequencies;
- Faults of up to 10 nodes in the network;
- White noise on the sensors with specified mean and standard deviation.

These scenarios have been tested at design-time, and the deployed network will be able to deal with these scenarios.

2.2.1 Design Considerations

An optimal design of a WSN should cope with several conflicting metrics while satisfying the given constraints. Optimizing a design to meet design requirements by maximizing or minimizing the metrics must take constraints into account. If hardware cost is a limitation, it is not a good idea to place several and/or expensive nodes, since, even if the final design is extremely optimized it will never be accepted since it violates the cost constraint. So, in the remaining of this thesis a *metric* is a measurable entity that must be **optimized** (minimize or maximize), while a constraint is a Boolean condition that must be **satisfied** (i.e., use at most 15 nodes in the network).

Regardless problem-specific requirements and constraints, each category is characterized by common design considerations. The lifetime of the system depends on both power consumption, battery capacity and the harvesting capabilities; the more power the application requires, the bigger the batteries and the more effective the harvesting should be. Vice versa, if the size of the batteries is fixed or the harvesting is not effective, architectures and applications must be power-aware.

Let us consider two distinct mathematical spaces: **Parameters** and **Metrics**. The *Parameters*' space (P) is a multi-dimensional space which defines all the feasible implementations of the design. Examples of the dimensions of this space are, i.e., the

position of nodes, the MAC layer, the routing protocol, the application's implementation, and so on. The *Metrics*' space (M) is a multi-dimensional space where all the feasible values of the required metrics (such as energy consumption, packet rate, etc.) are included.

2.2.1.1 PARAMETERS

During the design phase, a **parameter** is something which value can be defined by the designer. Every design choice is related to a specific parameter: CPU frequency, MAC protocol, radio TX power, etc. Every design project is characterized by its own specific design parameters according to: availability (i.e., there are 100 MSP430 processors available in stock), constraints (i.e., the law imposes the use of 2.4 GHz radio for this application), economy/license (i.e. the company already pays a license to use a specific standard and it will be very expensive to change the standard), etc.

For whatever reason, it is always a good idea to limit the amount of parameters and parameters' value such as the complexity of the design space is low enough to allow effective design space explorations. Section 2.3 illustrates the design complexity of WSN and will give the reader a better understanding of the problem.

2.2.1.2 METRICS

A specific combination of *parameters* results in a specific design that is characterized by specific values of the metrics. A **metric** is a measurable indicator that provides valuable information about the system. They differ from *parameters* since they cannot be arbitrarily set by the designer, rather their value is a result of the design choices. For example, *packet-receive-ratio* cannot be set at design time but depends on various factors like radio interferences, MAC configuration, routing algorithm, etc. For such reason, *parameters* are defined and *metrics* are evaluated (see Section 2.5).

2.3 On the Design Complexity

One of the most critical aspects of a WSN design is the cardinality of the design space which is too big to be explored exhaustively. This section aims at quantify the cardinality of the three spaces presented in this Section.

Let us consider Λ as a three-dimensional space whose size is equal to (X, Y, Z) meters and let discretize the space in squared cells of side of l meters. Assuming that more than one node can be placed in the same cell, if we would like to place a set of nodes Δ with $d_{min} \leq |\Delta| \leq d_{max}$, the number of possible solutions is equal to:

$$\sum_{i=d_{min}}^{d_{max}} \left(\frac{X * Y * Z}{l^3} \right)^i \tag{2.1}$$

For instance, if we would like to place from 20 to 40 nodes in an area of $100m^3$ with $l = 1m$, the amount of admissible solutions is greater than 10^{240}. Such cardinality discourages the use of semi-random algorithms to solve placement problems, thus custom algorithms are preferred.

Regarding the configurations' space, the amount of possible solutions depends on the kind of parameters. Considering the three kinds of configurations' parameters

(C_n,C_g and C_d), the size of the design space (upper bound) is:

$$\left(\prod_{c\in C_n}|c|\right)\left(\prod_{c\in C_g}|c|^G\right)\left(\prod_{c\in C_d}|c|^{|\Delta|}\right) \tag{2.2}$$

where $|c|$ corresponds to the amount of valid values of the parameter c, G is the amount of groups in the network and $|\Delta|$ is the number of nodes in the network. Assuming that a group includes at least one node, $G < |\Delta|$. From the equation is possible to see that node parameters' have a greater impact on the cardinality w.r.t. network parameters. It suggests that DSE on network parameters can be even performed manually, but refinements on node parameters requires an automated process.

Regarding cardinality of the applications' space A, since A includes all the implementations of all the possible applications, considering that there is no limit in the amount of nodes in a PDG, the cardinality of the PDG is infinite. Moreover, a feasible solution of a WSN design must have an application that implements the desired functionality. Although the set of applications that implements the desired functionality $A^F \in A$ is a small subset of ($|A^F| << |A|$), its cardinality is still infinite. In fact, $a \in A^F$ can be transformed in $a' \in A^F$, with $a \neq a'$, adding dummy tasks. Since this process can proceed to infinite, $|A^F|$ is infinite. For this reason, the automatic design of WSN applications, requires custom techniques.

In this section, we provide a characterization of the design space (denoted as S) of WSNs, in order to show the complexity of this kind of systems. The parameters space of a WSN is divided into two parts: a set of *node parameters* (P_n), which can assume different values on each node, and a set of *network parameters* (P_r), which assume one value throughout the whole network (or at least a part of it). For example, the memory size or the type of processor are specific for each node, hence they belong to P_n, whereas the network protocol must be the same among the nodes, therefore it belongs to P_r. Each parameter $p \in (P_n \cup P_r)$ is assigned to a discrete set of values in an interval $[p_{min}, p_{max}]$, whose cardinality is denoted as $|p|$.

Node parameters heavily affect the design space size $|S|$, since each parameter can assume an independent value on each node. Thus, when the network size (expressed in terms of the number of nodes, N) increases, then $|S|$ increases exponentially. More formally, we can express the size of the design space of a WSN:

$$|S| = \left(\prod_{p\in P_r}|p|\right)\cdot\left(\prod_{q\in P_n}|q|\right)^N . \tag{2.3}$$

To understand the order of magnitude of $|S|$, let us show a relatively small example for structural or environmental monitoring. Let us assume that 8 nodes are placed on a 2×2×2 tridimensional grid in the monitored area, and that the communication is regulated by the IEEE 802.15.4 MAC protocol [18]. Even without considering any hardware parameter, the configuration of the MAC protocol itself contains a wide set of possible parameters in P_r (e.g., the *frame* and the *beacon orders* [18]) and in P_n (e.g., the choice of using the contention active communication, or the number of requested *guaranteed time slots* [18]). Overall, the design space contains approximately 290 *billions* of solutions.

Thus, an extensive DSE would take an unacceptable amount of time when network simulation is the only viable way to evaluate a solution. As a consequence, a technique that is able to reduce the number of simulations is required.

2.4 Design Methodologies and Principles

The design problem consists in the identification of the optimal configuration of the parameters such as objective functions, defined as combination of metrics, are minimized. A specific configuration of the parameters is called **solution** and the process of identification of the best solution is called **Design Space Exploration** (DSE). In literature such problem is generally called **optimization problem** but, in order to disambiguate with respect to general optimization problems, the process of identification of optimal design solutions, in this thesis, will be always referred with the term Design Space Exploration or its acronym DSE.

As aforementioned, the design problem consists of two mathematical spaces: $\mathbb{P}$ and $\mathbb{M}$. The dimensions of $\mathbb{P}$ is the set of parameters, thus the set of configurable entities of the design, while $\mathbb{M}$ is the set of observable variables of the design. The main difference among these two spaces is that the values of the variables in $\mathbb{P}$ can be arbitrarily defined at design time, while the values of $\mathbb{M}$ are observed after the evaluation (see Section 2.5). Values of $\mathbb{M}$ are affected by evaluation inaccuracies, that depends on the chosen evaluation method.

This Section illustrates the common DSE loop, that is the basis of both manual and automated DSE.

2.4.1 Design Automation, Frameworks and Methodologies for WSNs

Design automation of WSNs is still an open research issue. An early analysis of the design space of WSNs has been proposed by Romer and Mattern in [139] where they present a comprehensive analysis of the typical requirements and characteristics of WSNs. The proposed analysis is focused on metrics and applications under a qualitative point of view and it does not provide practical tools for the development of tools for automated design space exploration.

Toward this direction, a high-level platform-based design methodology for WSNs is presented in [36]. To the best of our knowledge, this is the first design methodology specifically designed for WSNs able to consider concurrently hardware and software. The objectives were: first, raise the design abstraction level, second, ensure that final design's implementation will respect the initial requirements and, third, maximize component reuse. A recent work [130] proposes a complete system-level design flow for an alternative approach based on the concept of hardware microtasks. The authors show that hardware specialization and power gating are able to reach a power saving between one to two orders of magnitude w.r.t. MCU-based implementations.

Other than design methodologies, different design frameworks and tools have been proposed. A good framework for the rapid design and evaluation of WSNs is WISENES [89], which provides a high-level model for WSN. The target WSN is designed using the SDL language. The framework is able to simulate the WSN model and provides a backend for the code-generation for target platforms. Lately, McGibney et al. [110] propose a modeling and an optimization tool for WSNs focused on building wireless

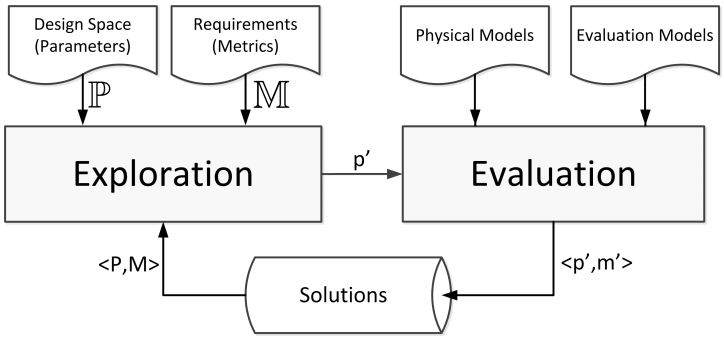


Figure 2.3: *The Design Space Exploration Loop*

application. The case study illustrates that the CAD tools effectively support the WSN designer during the design of the network. In the same year, Navarro et al. [123] presented a simulator based on SystemC language for the design space exploration of WSNs. Differently from the other approaches, main goal of the authors is to create a complete and generic design framework for the automated design space exploration of WSN rather than application specific tools.

Unfortunately, none of these approaches provide a generic and reusable formalization of the design space of WSN which is useful for the development of automated design space exploration algorithms for WSNs. To overcome this limitation, in this thesis, we provide a comprehensive formalization of the design space that can be used in any future design framework.

2.4.2 Design Space Exploration

The design of a WSN consists in the identification of a set of solutions that satisfy the constraints and are optimal with respect to the given requirements. As aforementioned, the process of the identification of this solution is generally known with the term **Design Space Exploration** (DSE) and, although DSE can be performed manually, the real challenge is the definition of automated algorithms that allow the identification of optimal solutions automatically since manual DSE requires expertise and sometimes is unfeasible due to the large number of design alternatives.

The DSE is an iterative process where several candidate solutions are evaluated, searching for the optimal solutions, according to given metrics. It is composed of two main components (Figure 2.3): **Exploration** and **Evaluation**. *Exploration* takes, as input, the (constrained) Parameters' space ($\mathbb{P}$), the Metrics' space ($\mathbb{M}$), the constraints (K) and a set of tuples $\langle p, m \rangle$ representing the current parameters' configuration ($p \in \mathbb{P}$) and the associated value of the metrics ($m \in \mathbb{M}$). It provides, as output, a set of new configurations ($p' \in \mathbb{P}$) to be tested (hopefully better than p). The *Evaluation* component takes the set of new configurations p' and applies an *evaluation function* on it ($m' = V(s')$). The obtained tuple $\langle p', m' \rangle$ is inserted in the *solution* database for next iterations. The objective of the DSE is to identify a set of configurations $P \subseteq \mathbb{P}$ such as $\forall p \in P$ are *Pareto-optimal*. Configurations in P are also defined as **optimal configurations** or **optimal solutions**.

CONSTRAINTS

Design constraints are used to pre-define unfeasible or undesired design solutions and are typically expressed with logical expressions over variables of the unconstrained parameters' space $\hat{\mathbb{P}}$. The resulting (constrained) space $\mathbb{P} \in \hat{\mathbb{P}}$ is used in the exploration, thus only the configurations which respect the constraints can be accepted at the end of the design phase.

There exist two kind of constraints: **strong** and **weak** constraints. *Strong* constraints define a strong relationship among variables which cannot be violated, otherwise the design cannot be accepted. *Weak* constraints define relationships among variables that suggest acceptable, but non-optimal configurations. An example of constraints in WSN design, related to IEEE 802.15.4 protocol, is defined on *Superframe Order (SO)* and *Beacon Order (BO)* parameters. According to the protocol, *SO* cannot be greater than *BO*, since *Superframe* period cannot be greater than *Beacon* period, thus a *strong* constraint is $SO \leq BO$. On the other hand, an example of *weak* constraint is related to the role of the nodes in such protocol. The structure of the *Superframe* is defined by the *network coordinator*, that is a specific node which coordinates communications among the nodes in its cluster. For such a reason non-coordinator nodes should not define such parameters since it is not required in the design. In conclusion, if *strong* constraints are used to avoid unfeasible solutions, *weak* constraints are introduced to reduce the cardinality of the design space, thus improve the DSE process.

EVALUATION

The **Evaluation** component computes metrics values for specific input configurations through an evaluation function. The **Evaluation Function** ($V(p) : \mathbb{P} \mapsto \mathbb{M}$) is a surjective function that maps $\mathbb{P}$ to $\mathbb{M}$. The objective of the evaluation function is to evaluate the quality of the given configuration with respect to the given metrics; these information will be used during the exploration to identify newer configurations. Evaluations can be performed either by using **models**, **simulations** or **testbeds** (see Section 2.5). Evaluated configurations are inserted into a shared database that will be used by the exploration component as a knowledge base to define which new configurations probably improve the known configurations.

EXPLORATION

The exploration component analyzes the set of configurations $P \subseteq \mathbb{P}$ in its database in order to identify a new set of configurations $P' \neq P$ such as $\exists p \in P' : \forall q \in P, V(p) > V(q)$. In other terms, the exploration component tries to discover new configurations with better objective functions with respect to known configurations. The exploration terminates when termination conditions have been reached (i.e. at least a configuration with metrics below a certain threshold), a certain amount of configurations have been evaluated, or the exploration converges. The algorithm converges when it is not able to improve the quality of the configurations and it can happen either because the Pareto set has been detected or for limits of the algorithm. It can be detected by monitoring if newer configurations do not improve the known configurations for a specific amount of time, i.e., a defined number of iterations.

Many techniques have been used here; they can be classified in three categories:

- **Model-Based Exploration**: custom techniques based on models of the problem to optimize. Optimal configurations are identified almost analytically from the model by inspection on equations;
- **Semi-Random Exploration**: when models do not exists or do not provide accurate results, generic semi-random techniques (such as genetic algorithms, simulated annealing, tabu-search, etc.) are preferred. These techniques require more evaluations with respect to Model-Based Explorations, thus exploration cost is higher. These techniques are also known with the term *meta-heuristics* [103];
- **Hybrid Exploration**: a mix of the previous two techniques. The idea is to use approximate models to guide the exploration to optimal configurations, and then use semi-random exploration where model accuracy is not enough. An example of this technique has been proposed by Beltrame et al. in [32].

2.5 Design Evaluation

The evaluation of the design's metrics is a fundamental part of the design process. As aforementioned, metrics cannot be defined, but derive from a conjunction of many components, thus the evaluation allows the designer to extract the effective quality of the design. The evaluation process answers to these questions: "*Does it (the design) meet the given requirements? Does it respect design criteria and constraints? Which design is better?*".

Design evaluation is an extremely important step in both manual or automatized design process, thus it must be defined accurately. In defining the evaluation method, two aspects should be considered: **accuracy** and **speed**. *Accuracy* means that the evaluation results adhere the reality of the phenomenon, thus they are a reliable source of information. Moreover, some method is able to perform accurate evaluations of very specific aspects, thus its overall accuracy is low. *Speed* indicates how fast results are computed; the faster are the evaluations, the more design choices can be evaluated and compared. Each evaluation technique is characterized by intrinsic values of *accuracy* and *speed*. Intuitively, the higher are these two aspects, the better is the evaluation method. Unfortunately, considering the design problem as a whole, none of the proposed techniques is both accurate and fast; some of them are fast but not accurate and vice versa.

A trade-off among *accuracy* and *speed* is required and should be examined before every design process in order to identify the technique that best fits economic requirements such as **time-to-market** and **product-quality**. Accurate but slow design evaluations allow the designer to better analyze the system and to optimize non-functional characteristics such as reliability, power consumption or efficiency that positively affect *product-quality* but increase *time-to-market*. Conversely, quick evaluation techniques enable a fast *time-to-market*, but do not provide any assurance about the full compliance to the design requirements.

2.5.1 Model

Often in engineering and science, models are used to analyze processes or systems, providing a useful tool for the investigation of such a system. Once created, a model allows quick analysis and forecasts and it is extremely useful to investigate pre-defined

scenarios, expecially when reproducibility of the phenomenon is difficult or expensive. A model can be used to evaluate the quality of a solution by computing the values of the metrics from the parameters.

Although models are good for quick analysis of the phenomenon, very often, to keep model-complexity as low as possible, model accuracy is not enough to ensure reliable results. When the process is complex, non-linear, or application-dependent, model's accuracy is below the tolerance and different evaluation techniques are preferred. On the other hand, models are usually defined as closed systems, thus their evaluation is extremely quick.

In the WSN field, a large amount of models have been presented in literature to cope with different aspects of such systems. Many of these models have been inherited from embedded system design (CPU energy consumption, sensing efficiency, ...) or from the telecommunication and networking field (queue theory,), but many of them refer to peculiar aspects of WSNs (protocols, sensing correlation, ...). Model-based evaluation has a long history, as many models have been proposed to describe the basic components of a node (e.g., memory, radio, etc. [81] [64] [142]). However, combining those components to form a model of the entire node is no easy task, as the model should include meaningful information of the specific node, while being reusable and not requiring a massive amount of experimental data to be constructed. In order to cope with the difficulty of building reliable node characterizations, a promising trend is to generate statistical models from a properly-selected set of experimental data [29]. The experimental data is used to estimate the parameters of a set of simple equations, which however do not provide an application-aware evaluation of the node.

ENERGY CONSUMPTION

Energy consumption is an important aspect in WSNs, although many models have been proposed in the state of the art, considering that energy consumption is a fundamental metric for sensor network design, it is quite difficult to find general purpose energy models for WSNs. Instead, energy consumption considerations and models are always included in analysis, protocol design and modeling. Moreover, power models of single components such as the microcontroller [64] and the memory [81] are available outside the scope of wireless sensor networks. Another analysis of energy consumption of WSNs have been analyzed in [52]. The authors analyzes both homogeneous and heterogeneous networks and estimate its energy consumption (and lifetime) in order to quantify the optimal number of clusters.

RADIO

Radio is a crucial component of the communication, since it is responsible in delivering each bit over the communication medium. The correct modulation-demodulation scheme allows the radio to work with better signal-to-noise ratios mitigating the interferences and transmitting more efficiently. A characterization of the radio has been proposed in [142], where the energy consumption is related to parameters like the bit error rate and the modulation. Similarly, [43] provides a model for an IEEE 802.15.4 transmitter, which is supported by a set of physical measurements. Received signal strength in radio communications are rarely uniform in the space and time, rather they

depend on the MAC layer and the direction. Using empirical data measures on the MICA2 platform, the author of [168] defines a radio model able to overcome the discrepancy among spherical radio models and the reality of radio signals.

MAC LAYER

The MAC layer is responsible to synchronize the nodes in the network avoiding collisions and keeping the energy consumption as low as possible. Moreover, the MAC layer is mainly responsible to decide sleeping and active periods of the nodes and their synchronization. The customization of the MAC layer is fundamental in the process of WSN design optimization, thus accurate models of this component are required.

Thanks to its standardization, the most common MAC layers in WSN belongs to the IEEE 802.15.4 family, thus a lot of models have been created for this standard. One of the first analysis of the IEEE 802.15.4 protocol has been conducted in [102], where the authors present a preliminary performance evaluation of such protocol. These results allowed the author of [151] to define the first performance model of IEEE 802.15.4; although the model is focused for medical applications with implanted sensors, the model is rather generic thus it works also in other scenarios. Lately, Kohvakka et al. [82] extended the analysis to large-scale sensor networks based on the IEEE 802.15.4 protocol. The analysis has been conducted on the CSMA-CA mechanisms on a ZigBee clustered network. The authors present a number of formulas to estimate the collision and retransmission probabilities, the power consumption and the goodput. The model has been verified and validated by simulating the ZigBee network in WISENES simulator [90]. More recently, a very accurate and complex model for the CSMA-CA mechanism in beacon-enabled networks has been proposed in [40] and [67]. On the other hand, a performance analysis of GTS allocation in beacon-enabled IEEE 802.15.4 has been presented in [129]. Current consumption modeling and measurement has been presented in [43]. The model describes the amount of current drained from the power source under different IEEE 802.15.4 communication operations. On the other hand, in [146] the authors propose an alternative MAC protocol for the *ZigBee* standard that introduces new power-saving policies. In [99], a model that relates the routing performed at the MAC level to the node lifetime is proposed.

The development of BAN applications for medical purposes lead toward the definition of new standards for IEEE 802.15.6. Although the standardization process is still not completed, various models of such protocols have been developed. In [156], Viittala et al. compare the model of the Ultra Wide Band (UWB) radio channel model with real measurements conducted in an hospital in order to prove the correspondence between the model proposed by the IEEE 802.15.6 sub-task group for WBAN with the reality. An analysis of energy consumption for scheduled access mode in IEEE 802.15.6 has been presented in [149] in order to provide an useful model to estimate the device lifetime. This analysis can be used for both system design offline analysis or online optimization, to estimate power consumption of the device. A numerical formulation of throughput and delay limits of IEEE 802.15.6 for an ideal channel with no transmission error has been proposed in [154]. These studies can help the designer to estimate the feasibility of its application in a very early development phase or, as suggested by the authors, to help the protocol designers to optimize the packet size and to determine the upper bound of IEEE 802.15.6 networks.

ROUTING PROTOCOL

Modeling a routing protocol is extremely difficult, due to its time characteristics and data dependencies. The behavior of routing protocols can be either address-centric (traditional routing), data-centric (based on routing content [84]) or probabilistic (packets are disseminated in the network based on probabilistic phenomenon [96]) and modeling all these behaviors can be difficult or extremely inaccurate. More frequently, models are used to show the scientific strength of the approach or to motivate its efficacy. In this thesis work we are interested in models to evaluate the design, not to design better routing protocols. According to these considerations, some aspects of routing protocols have been modeled such as energy consumption and performance. In [54], the author presents an energy consumption model of routing protocols for mobile ad-hoc networks. The analysis has been focused and validated on two routing protocols: Dynamic Source Routing (DSR) and Ad hoc On-demand Distance Vector (AODV). Another work on energy consumption modeling has been presented in [45], where the authors model the routing as linear programming problem. Network lifetime (defined as the time until node battery fails partition the network) is defined as objective function to be maximized. Simulations, performed with both uniform and arbitrary traffic patterns, show that the proposed modelization helps the identification of the optimal network lifetime.

The impact of data aggregation in sensor networks has been studied in [84,86] where theoretical results are presented. In particular, the authors show that data-centric routing algorithm substantially improve the performance with respect to address-centric routing. Moreover, they prove that, even if the complexity of optimal data aggregation is theoretically an NP-hard problem, polynomial-time solutions exist.

2.5.2 Simulation

Models provide a fast way to evaluate design metrics such as energy consumption or performance, but are unable to give information about temporal phenomenon such as critical runs or synchronization problems. Simulations are a powerful way to check functional correctness of applications, protocols and algorithms, and to verify the quality of the design with respect to given metrics. Simulation of WSNs is currently a very active field and many network simulators have been created. Unfortunately, an accurate simulation of a sensor network requires accurate low-level models for wireless channel, interferences, clock drifts, sensor noise, etc. as well as MAC and routing protocols, applications, and visual support. Moreover, a simulator must be efficient (simulation time) and scalable (simulation size). A good survey on simulation frameworks can be found in [148]; the following analysis is inspired to this document.

Summarizing, the main characteristics of sensor network simulators are:

- ***Abstraction Level***: a network simulation can be conducted at various levels of abstraction such as application level, network level or physical level. Usually, the higher is the abstraction level, the less accurate will be the simulation;
- ***Hardware Dependency***: a simulation is performed to check the functionality of the application before the deployment on a real hardware. Some network simulators have been developed for the testing of applications for specific hardware,

others to test the applications on generic sensor networks. The first ensure that the simulation will work on the target hardware, but are unusable to evaluate the algorithm on other hardware, the latter does not allow the direct deployment on a real hardware, but give a general view of the quality of the algorithm on a wide variety of nodes;

- ***Performance and Scalability***: evaluation time is a crucial aspect of evaluation methods. For simulators, performance and scalability typically depend on the language and framework used to write the application and the level of detail of the simulation; the more details are considered, the higher will be the computation time required, thus the higher is the simulation time. Typically, performant simulators are written in compiled languages like C/C++, while others prefer portability instead of performance. Scalability consist in the number of nodes and events that can be processed in a simulation, thus it deals with memory requirements rather than efficiency of language used by the simulator;
- ***Graphical and Debug Support***: writing complex and distributed applications, graphical and debug support is extremely important to discover and correct bugs. Moreover, graphical support can be useful to analyze and discover communication patterns or to simply observe how the network works. Debug support can be embedded into the simulation environment or carried with the technology used to write it;

The simulation framework should be identified according to the design needs. If the focus of the design is the definition of a powerful routing protocol, thus the simulator should allow the designer to easily define, test and visualize distributed protocols, otherwise, if the focus is on wireless interferences, so the simulator must accurately simulate low-level behaviors.

Some of the available (open) simulators are:

- **NS-2** [8]: very popular discrete event simulator for network research. It has a strong support of TCP, routing and multicast protocols over wired and wireless networks to simulate both LAN, mobile ad-hoc and wireless sensor networks. NS-2 is easily extendible, thus new protocols can be included and integrated with existing infrastructure. Although an high number of protocols is publicly available, few WSN-specific protocols have been implemented. Moreover, energy models and hardware-dependent aspects are substantially different with respect to actual hardware platforms and sensors;
- **TOSSIM** [94]: it allows the definition and simulation of entire TinyOS-based applications, thus it represents a good simulator in case the target platform support such operating system. The TinyOS stack can be simulated at the bit-level, allowing experiments with low-level protocols in addition to high-level network protocols and applications. Two plugins allow TOSSIM to simulate energy models (PowerTOSSIM) and to have graphical support (TinyViz). TOSSIM does not capture CPU time, thus critical runs and synchronization problems cannot be studied. It requires each node to execute the same piece of code making difficult to test heterogeneous applications;

- **UWSim** [13]: framework for Under Water Sensor Networks (UWSN) designed to simulate marine robotics research, which is a unique characteristics of such simulator. UWSim has been implemented in C++ and is distributed with a powerful 3D library; 3D scenes can be easily configured with third-party modeling software. It natively supports popular routing protocols such as AODV and DSR. Being developed for UWSN, as a drawback, it does not allow simulations of networks rather than UWSN;
- **Avrora** [1]: cycle-accurate simulator for AVR embedded platforms that allows to accurately simulate both microcontroller programs and radio communications. Rather than single-node, microcontroller-focused simulations, it is able to simulate complete networks. In addition, plugins enable Avrora to simulate the TinyOS network stack. It efficiently scales to networks of up to 10.000 nodes. As major limitation, Avrora does not support clock drift, and it is 50% slower than TOSSIM;
- **SENS** [10]: simulator with a modular, layered architecture with customizable components that allows realistic simulations of sensing data (that can be defined by the user on real measurements). Although it is a platform-independent simulator, WSN nodes can be characterized and customized to fit the real hardware. In addition, SENS provides the user various modeling and interaction mechanisms of the physical environment. However, SENS is less customizable than other simulators and the only physical phenomenon that can be detected by sensors is sound;
- **COOJA** [125]: simulator designed to test Contiki applications that, similarly to TOSSIM, allows the designer to test the same code that will be deployed. Unlike TOSSIM, nodes with different source code can coexist in the same network simulation. The Contiki code can be emulated by the COOJA framework or executed directly on MSP430 hardware. The main drawback of this simulator is the efficiency: the code-level simulation requires several calculations, thus long and complex simulations are very hard to perform;
- **Castalia** [9]: application-level simulator for WSNs, it allows to simulate fine-grained aspects such as wireless interferences, clock drifts, sensor noise and bias. Radio models are based on real measurements, that makes Castalia results extremely realistic. It is delivered with highly customizable radio and MAC components (included IEEE 802.15.4 and IEEE 802.15.6). Castalia is not a hardware-specific simulator, thus it should be used to evaluate feasibility and correctness of algorithms under realistic conditions;
- **Shawn** [55]: designed to support large-scale network simulations. Shawn has the highest level of abstraction among the simulator presented in this thesis, which explain its performances and scalability. However, detailed simulations of radio propagation or other low-layer issues are not well modeled, thus the overall accuracy is affected;
- **EmStar** [59]: it simulates iPAQ-class sensors running Linux, allowing a flexible environment to easily deploy simulated code. It can be used to test various execution platforms, combining simulation and emulation. However, EmStar uses an

extremely simple environmental and network medium model that affects the overall accuracy of the simulation. Moreover, it does not support parallel simulations and lacks algorithms that are reactive to sensed values;

- **VisualSense** [14]: implemented in the Ptolemy-II framework, it has the objective to accurately simulate sophisticated behaviors of wireless channels and physical processes such as acoustic channels. Although the component-oriented structure allow fast extendability, VisualSense does not provide any protocol above the physical/wireless medium or any physical phenomenon rather than sound;
- **(J)Prowler** [4]: event-driven deterministic and probabilistic wireless sensor network simulators developed on the top of MATLAB (Prowler) and Java (JProwler). Initially designed to target Berkley MICA motes running TinyOS applications it could be used to simulate more general systems. Simplified radio and MAC models allows the simulator to provide accurate results at a reasonable performance and scalability. However, (J)Prowler is delivered with only one MAC protocol;
- **MiXiM** [5]: OMNET++ modeling framework designed to simulate mobile and fixed wireless networks including wireless sensor networks, body area networks, ad-hoc networks and vehicular networks. Thanks to the wide spectrum of target applications, it includes several MAC and routing protocols. Although detailed models of radio wave propagation is included, other simulators (such as Castalia and VisualSense) have more realistic models;
- **WISENES** [15]: implemented in Specification and Description Language (SDL) and delivered with a powerful GUI, WISENES is a powerful and extensible simulation platform. All the components, including transmission medium and sensing channel, are implemented in SDL that provides a formal and clear graphical notation and give the designer an easy-to use way to implement and check new components. However, WISENES has been designed for high-level programming, thus low-level issues are not well modeled.

In addition to these aspects, another important characteristic of the simulators is the ability to be invoked by an external tool. This feature is extremely important during the automatized Design Space Exploration, since the evaluations and the whole loop are performed automatically. All of the simulators presented above can be invoked externally, thus can be used to perform automated Design Space Exploration. Table 2.4 presents a comparison between the simulators presented above.

Simulator	Programming Language or Platform	Abstraction Level	Hardware Dependency	Performance & Scalability	Graphical Support	Debug Support
NS-2	C++	LOW	-	MEDIUM	YES	Included
TOSSIM	nesC	HIGH	TinyOS devices	MEDIUM	YES	Included in GUI
UWSim	C++	VERY LOW (Physical Processes)	Underwater Sensor Networks (UWSN)	LOW	YES	Included in GUI, GDB
Avrora	Java	HIGH	AVR-based devices	MEDIUM	-	GDB
SENS	C++	LOW	-	MEDIUM	-	GDB
COOJA	Java (Simulation in C)	HIGH	Contiki	LOW	YES	CDT
Castalia	C++	MEDIUM/HIGH	-	LOW	-	GDB, OMNET++ tools
Shawn	Java	VERY HIGH	-	HIGH	-	JDB
EmStar	Linux	VERY LOW	-	LOW	-	GDB
VisualSense	Ptolemy-II	LOW	-	LOW	YES	Included in Ptolemy-II
(J)Prowler	Matlab & Java	LOW	-	MEDIUM	YES	Included in GUI, MATLAB
MiXiM	C++	MEDIUM/HIGH	-	LOW	YES	GDB, OMNET++ tools
WISENES	SDL	HIGH	-	HIGH	YES	Included in GUI

Table 2.4: *Comparison of Different Simulators*

2.5.3 Testbed

The most accurate technique to evaluate a design is through direct measurements on a real testbed. According to the application's type, a testbed can exactly represents the final system, or only be an approximation. For a Body Sensor Network, i.e., due to the cardinality of nodes, testbeds are accurate, instead, for structural monitoring, testing the final system can be extremely expensive and complex.

Due to these reasons, a testbed typically represents a general purpose network with a generic topology (uniform, grid), which is not the final topology. The objective of using a general-purpose testbed is to test the hardware and the implementation of the network protocols, in order to control the correctness of the implementation on the real platform.

Although these approximations, the accuracy of the experiments on the testbeds is extremely high since the measurements are performed directly on the hardware that will be used in the deployment and not on virtualization or modelization of it; if the design works correctly on the testbed, it will work correctly on the final deployment.

To overcome maintenance and setup costs of testbeds, and to provide standard conditions to compare protocols and algorithms, several publicly available testbeds have been proposed. Some of these testbeds are:

- **MoteLab** [7]: an experimental Wireless Sensor Network testbed deployed in Maxwell Dworkin Laboratory at Harvard University. It is publicly available for development and testing of sensor network applications and is composed by 190 nodes deployed over three floors. Deployed nodes are TMote Sky sensor nodes, which consist of an TI MSP430 running at 8MHz, 10KB of RAM, 1Mbit of Flash memory and a Chipcon CC2420 radio operating at 2.4 GHz with an indoor range of approximately 100 meters. Each node includes light, temperature and humidity sensors and run the TinyOS operating system. In addition, a web interface helps the designer to deploy and control the output of its applications.

- **TWIST** [12]: developed by the Telecommunication Networks Group (TKN) at the Technische Universität of Berlin, the TKN Wireless Indoor Sensor network Testbed (TWIST) is a scalable and flexible testbed architecture to test indoor sensor network applications. It allows the designer to extract and debug application data and test heterogeneous sensor networks. Moreover, the testbed allows the designer to actively monitor and control the energy consumption of nodes. The testbed is composed by 204 nodes (102 TMote Sky and 102 eyesIFX nodes) deployed on a regular grid which intra-node distance is 3m.

- **INDRIYA** [3]: a three-dimensional sensor network testbed deployed on three floors of the School of Computing at the National University of Singapore. It is a permanent and public framework to develop and test sensor network protocols and applications; a web-interface helps the user to interact with the testbed, upload executable and run protocols and applications. The testbed is composed of 139 TelosB nodes equipped with TI MSP430 processors, 10KB of RAM, 48KB of internal Flash, 1MB of external Flash and a Chipcon CC2420 radio operating at 2.4 GHz characterized by an indoor range of 20-30 meters. Nodes have different

sensors among passive and active infrared, accelerometer, magnetometer, light, temperature, and acoustic. All the nodes run the TinyOS operating system.

- **WUSTL** [16]: developed and deployed by the Washington University in St. Louis to measure communication characteristics of motes, protocols and applications. It is currently composed of 79 TelosB sensor nodes placed through several offices over two buildings. The testbed have several gateways directly connected with the nodes to allow fast, reliable and direct node programming and debugging. The nodes run TinyOS operating system and data are collected in a PostgreSQL database.
- **SensLAB** [11]: a very large testbed deployed over 4 research centers (INRIA Lille, INRIA Rennes, Strasburg/LSiiT and INRIA Grenoble) that compose a network of 1000 sensor nodes available to persons affiliated to corporations hosting SensLAB, or also any researchers for R&D on request. The main goal of SensLAB is to offer an accurate and efficient tool for scientific research, design, development and testing of real large-scale sensor network applications.

The main purpose of these testbeds is to offer a common platform to develop, test and compare protocols and applications on sensor networks. Due the large variety of platforms and technologies available in the sensor network field, use these testbed to compare protocols and approaches is extremely important to ensure fair comparisons and replicable results.

According to the purpose of this thesis work, testbeds provide the most accurate and reliable way to evaluate the designs, but both the setup and running time discourages the use of it in the automated DSE loop. However, testbeds are required to check the accuracy of simulation results among candidate final solutions before the final deployment.

2.5.4 Comparison Among Evaluation Techniques

In previous sections we presented three evaluation techniques, in this section we aim to compare them in order to define the characteristics of each approach. From the analysis performed above, we extract that models are the most efficient way to evaluate a design, while testbeds are the most time-consuming technique. On the other hand, testbeds allow the user to evaluate the design, the protocols and the application directly on the nodes that will be used in the deployment, thus the accuracy is extremely high. On the contrary, models define only specific aspects of the problem, and their accuracy strictly depends on the complexity of the problem, thus they are the less accurate way to evaluate a design. Simulators stands in the middle, giving a good compromise among evaluation accuracy and speed; time-dependent operations and synchronization problems, that are extremely complex to describe in a model, are easily implementable in a simulator.

In an automated DSE, the evaluation speed is generally preferred over accuracy since many solutions need to be compared with each other. Moreover, expecially during a first-order design, an average accuracy is enough to allow the system to discriminate good from bad solutions thus, in the following, we never use testbed for automated

DSE. So, accuracy-speed comparison is more critical considering model and simulation, where models can be used for first-order analysis and simulations only a better accuracy is desired; a deeper analysis of that will be provided in Section 4.4.3.1.

Figure 2.4 graphically illustrates a qualitative accuracy/speed comparison among models, simulations and testbed. As previously presented, the best practice during a design is using models for first-order analysis, or to identify a possible set of optimal solutions, then refine the choice through extensive simulations and, finally, verify the design on a real testbed.

In conclusion, considering the stochastic behavior of the wireless channel, communication protocols and applications, even using an extremely accurate evaluation method, two successive evaluations rarely give exactly the same results. More frequently, the observed values vary in an interval with specific mean and variance, representing the expected value and its uncertainty. The more the evaluation method is able to identify such statistical distribution, the more accurate the evaluation method is.

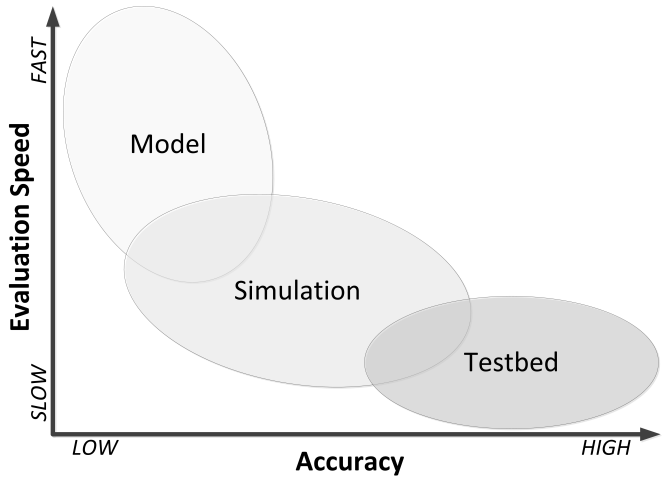


Figure 2.4: *Qualitative accuracy-speed comparison of different evaluation techniques*

2.6 IEEE 802.15.4/ZigBee

IEEE 802.15.4 has been designed for low power devices with low data rate Personal Area Networks (PAN). It specifies how the Physical Layer (PHY) and the Media Access Control (MAC) must work. Standardization process of IEEE 802.15.4 started in 2003 and ended in 2009 with the latest updates from IEEE. IEEE 802.15.4 became popular thanks to the diffusion of **ZigBee** devices that specify the standard in high-level communication protocols and extend some features to provide a better energy efficiency. For the sake of simplicity, from now on, the terms ZigBee and IEEE 802.15.4 will be considered as synonyms.

IEEE 802.15.4 based networks are composed of a central node, called **coordinator**, and a set of nodes, called **members**. The coordinator is the head of the network and determines the structure of the communication. The communication is divided into sequential frames delimited by specific packets called beacons (Figure 2.5). The coordinator sends periodic packets, named **beacons**, that define the **superframe** structure. A *superframe* is a portion of time, bounded by successive beacons, and it is used to

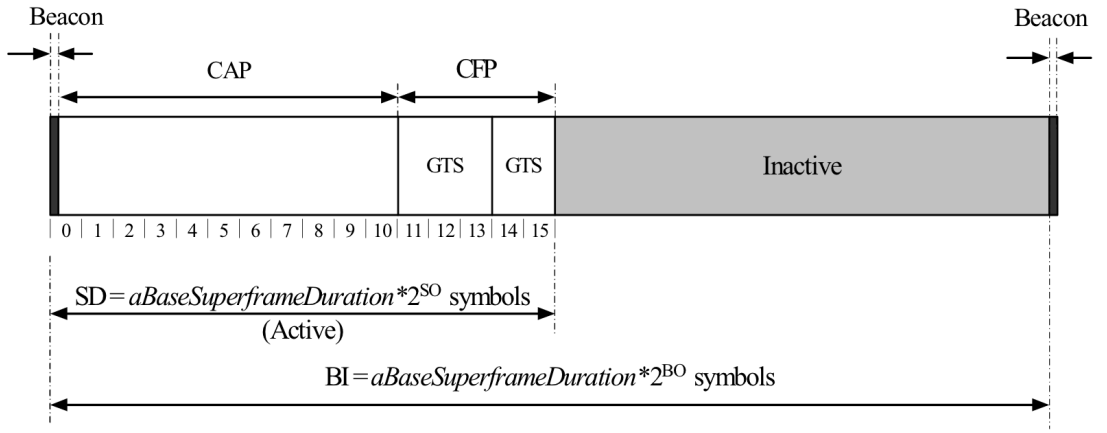


Figure 2.5: *An example of the superframe structure (from [18])*

define how the end-devices must communicate with the coordinator. The superframe is composed of an **Active Period** (SD) and an **Inactive Period** (BI-SD). The active period is further divided in two periods named **Contention Active Period (CAP)** and **Contention Free Period (CFP)**. CAP starts immediately after the reception of a *beacon* from the coordinator. During the CAP, nodes access the channel by using the CSMA/CA protocol. CFP starts at the end of the CAP and the end-devices uses *Guaranteed Time Slots* (GTS) to freely access the wireless channel. During the CFP, the nodes access the channel using a time division protocol which slots, namely Guarantee Time Slots, requested by each member node, are assigned by the coordinator by means of a policy *first come first first served* (FCFS) [18].

3 Target Platforms

In the last decade, a considerable amount of hardware platforms have been developed in order to test and develop applications and protocols for wireless sensor networks, enabling the Internet of Things. These platforms are mainly general purpose platforms equipped with low-power microcontrollers, radio interface and Flash memories. Sensors are either included into the platforms or can be connected through general purpose I/O or daughter boards.

The goals of this chapter are twofold: first provide the reader a comprehensive review of currently available commercial and open source platforms and, second, present a research study conducted on the use of FPGAs in WSN's nodes. Moreover, this chapter would like to give the reader a deeper comprehension on the difficulties of design and customization of WSN's platforms.

3.1 Microcontroller Based Platforms

MICA* Motes

Initially designed at Berkeley university, these nodes have been widely accepted in the research community, mainly because of their compatibility with TinyOS. This family includes MICA2 (Figure 3.1) and MICAz nodes. MICA2 are equipped with the MPR4*0CB processor (based on the ATmega 128L) which provides 128KB of program Flash memory, 512KB of Flash for measurements and 4KB of configuration EEPROM, delivered with several radios working at 315, 433, 868 and 916 MHz. MICAz are equipped with MPR2400CA processor with 128KB of program Flash memory, 512KB of Flash for measurements, 4KB of configuration EEPROM and a 2.4GHz ZigBee-compliant radio. Both nodes are powered with two AA batteries and are delivered with 51-pin expansion board to connect third-party components such as sensors or actuators.

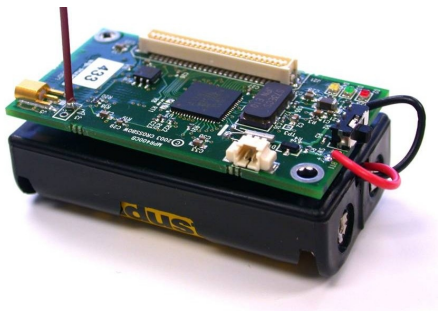


Figure 3.1: *MICA2*

TELOSB

Open-source platform designed to enable fast-experimentation of cutting-edge researches. TelosB (see Figure 3.2) has a IEEE 802.15.4 compliant integrated radio working at 250 kbps and the MSP430 Texas Instrument microcontroller with 10KB of RAM. The device is equipped with 48KB of program Flash memory, 1024KB of Measurement Flash and 16KB of configuration EEPROM. Similarly to MICA* platform, it is powered with two AA batteries and it natively supports TinyOS. 16 general purpose I/O pins allow the integration of third party components such as sensors or actuators.

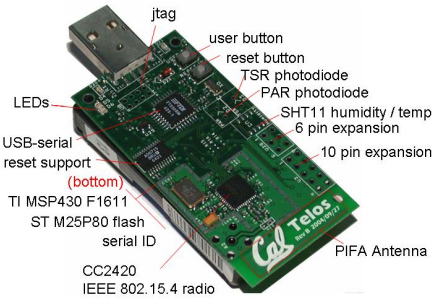


Figure 3.2: *TelosB*

TMOTESKY

Ultra low power device equipped with humidity, light and temperature sensors, a 2.4 GHz IEEE 802.15.4 compliant Chipcon radio working at 250 kpbs and a 8MHz MSP430 processor (see Figure 3.3). The integrated antenna allows communication with up to 50m indoors, and 125m outdoors. The device has been designed to be powered with two AA batteries. The board includes 16 general purpose I/O pins for the integration of external components.

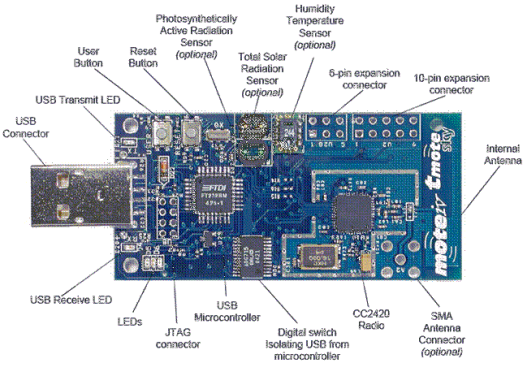


Figure 3.3: *TMoteSky*

LIBELIUM WASP MOTES

Libelium was a spin-off of the University of Zaragoza born in 2006. It produces hardware for development, integration and deployment of wireless sensor networks for Smart Cities and the Internet of Things. Their main product for WSN development is the WASP mote that is a low-power general-purpose, expandable platform (see Figure 3.4). It is equipped with the ATmega 1281 Microcontroller working at 8MHz, with 8KB of SRAM, 4KB of EEPROM, 128KB of Flash memory and a 2GB SD card. The board has several expansion sockets to sensors, ra-

Figure 3.4: *Libelium Waspmote*

dios or other components. The board supports ZigBee, GPRS, Bluetooth, WiFi, RFID and NFC modules and has been developed to natively support *Over the Air Programming*. Libelium produces several sensor boards compatible with this platform such as gases (CO, CO2, CH4, ...), temperature, liquid level, weight, pressure, humidity, luminosity, accelerometer, soil moisture, solar radiation and GPS. The platform supports energy harvesting through external solar panels.

SHIMMER PLATFORM

The SHIMMER [42] is a small, low-power commercial wireless sensor platform specifically designed for noninvasive biomedical research (see Figure 3.5). It is equipped with an ultra-low-power 16-bit microcontroller (TI MSP430), that runs at a maximum clock frequency of 8MHz and includes 10KB of RAM and 48KB of Flash, as well as some peripherals such as an 8-channel analog to digital (A/D) converter and a direct memory access unit (DMA). This platform has also two radios (Bluetooth and IEEE 802.15.4-compliant), a 3-axis accelerometer and an expansion port to connect a daughter board that can include additional sensors such as electrocardiogram (ECG), electromyogram (EMG), galvanic skin response (GSR), 3-axis accelerometers, gyroscope, magnetometer, temperature, pressure, strain gauges, GPS, tilt and vibration. SHIMMER platform has been used as target device in this thesis for Body Sensor Networks applications.

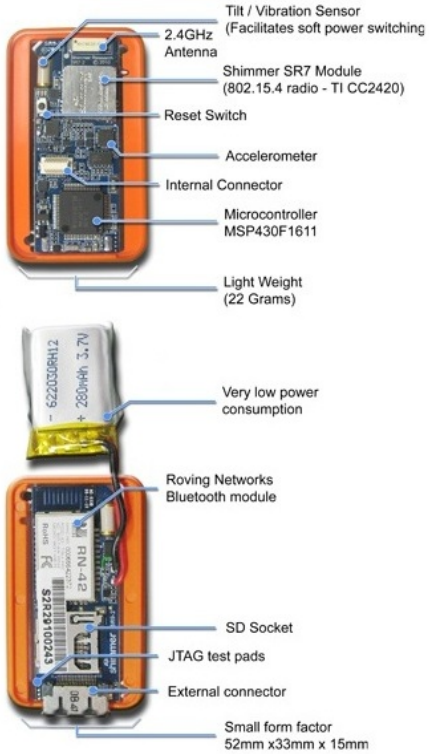


Figure 3.5: *The SHIMMER platform*

ORACLE SUN SPOT

Originally developed by Sun Microsystems and now owned by Oracle, Sun SPOTs are java enabled sensor networks designed to encourage and support the evolution of the internet network. This device is compose by two boards: processor and sensor board (see Figure 3.6). The 8th revision of the processor board includes an AVR AT91SAM9G20 processor working at 133 MHz with 125KB of RAM, 1MB of Flash memory and a

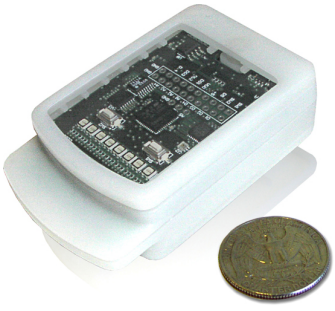


Figure 3.6: *Oracle Sun SPOT*

temperature sensor. The sensor board has
4 digital GPIO pins, 4 analog lines, a 3-axis accelerometer, lo-fi speaker, I2C connector and an IR receiver and transmitter. This board has not been designed for ultra low-power applications, but to allow high-level application development and to encourage the Java development for future internet of things.

ARDUINO

A very popular open-source prototyping platform delivered with an open-source easy-to-use programming environment. Arduino projects are entirely open-source and boards can be assembled or purchased from an authorized reseller. The designer developed several boards with different sizes, performance and end-purposes. In this thesis, the target platform is the *Arduino Funnel I/O (FIO)*, a small device designed for wireless applications (see Figure 3.7). It includes an ATmega328P processor running at 8MHz with 2KB of SRAM, 1KB of EEPROM and 32 KB of Flash memory. The board has 14 digital I/O pins (of which 6 provide PWM output), 8 analog I/O, a socket to connect XBee radios and a connector for Lithium Polymer batteries. The board natively supports the *Over the Air Programming* (OTA).

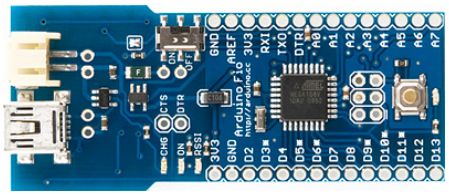


Figure 3.7: *Arduino Fio*

3.1.1 Comparison of Platforms

Heterogeneity of hardware boards and devices increases the amount of available design alternatives making the design process harder. Table 3.1 presents a brief comparison of hardware characteristics of these nodes. The nodes have been compared in terms of processor, memory, supported operating system and I/O capabilities. All these boards present some input/output pins to connect external sensors or actuators, according to project's needs, thus sensors are not specified.

All the boards have an 8MHz processor except for the *Oracle Sun SPOT* that is equipped with a 133MHz processor (motivated by the use of Java on the platform but definitely not energy-friendly). Memory is an important aspect in modern WSN nodes and conceptually, the higher is the amount of memory available the better it is. However, 10KB of RAM are enough for many applications; in [138], for example, the authors show how to implement an accurate atrial fibrillation detection algorithm using only 10KB of RAM. On the other hand, Flash memory are typically used to store historical data from sensors or configuration data; high flash memory capacity (i.e. 2GB in *Libelium Motes*) could be useful if data are not transmitted continuously to the sink. Concluding, EEPROM are use just to store boot program, thus its size is usually very limited.

The supported operating system is extremely important to guarantee a wide diffusion of the device on the market. The most popular OS for WSN is TinyOS, that is supported by many platforms. For the Oracle Sun SPOT, Java is not an operating system, but the technology that is supported. The other platforms can be programmed in C, C++ or

Table 3.1: *Comparison of Hardware Platforms*

	Processor	Memory			Supported OS	I/O
		RAM (KB)	Flash	EEPROM (KB)		
MICA* Motes	MPR4*0CB@8MHz		128+512	4	TinyOS	51-pins
TelosB	MSP430@8MHz	10	48+1024	16	TinyOS,FreeRTOS, Contiki	16-pins
TMoteSky	MSP430@8MHz	10			TinyOS, FreeRTOS, Contiki	16-pins
Libelium WASP Motes	ATmega1281@8MHz	8	128+2GB (SD)	4	FreeRTOS	Expansion Sockets
SHIMMER Platform	MSP430@8MHz	10	48		TinyOS, FreeRTOS, Contiki	Expansion Sockets
Oracle Sun SPOT	AT91SAM9G20@133MHz	125	1024		Java	Sensor Board
Arduino FIO	ATmega328P@8MHz	2	32	1	FreeRTOS	14 (digital) + 8 (analog)

derivated dialects like nesC (TinyOS); it provides a reasonable code density and a good execution efficiency.

As aforementioned, I/O characteristics are extremely important to ensure the success of a platform. The platform must be able to support any kind of external device like sensors, actuators or radios. In fact, almost all the platforms do not have an embedded radio interface on them in order to allow the user to buy the device that best fits its needs. For example, Arduino FIO has a dedicated slot to connect various kinds of radios and MAC layers.

All the platforms presented in this Section are similar in one characteristic: they all are classical processor-based platforms with a set of I/O directly connected to the processor. In these platforms the software and the OS are in charge of performing sampling, computation, filtering, communication and to manage power-aware policies. The next Section presents an alternative node, based on FPGAs, specifically designed to manage high-throughput data such as audio or video keeping the energy consumption lower than 4mW.

3.2 An FPGA-based Platform

Activity of WSN nodes depends on the kind of sensed data. In case of low sample rates, an higher duty cycle ensures a strong reduction of power consumption thanks to long sleeping periods, while high sample rates are more energy demanding. In case of high sample rates (above 100 Hz) if strong signal processing must be performed, intra-sample time cannot be enough to process information using low-power microcontrollers. When applications are characterized by high throughput data, i.e video or audio sensors, additional devices, like DSPs or FPGAs, are required. Recent works show that FPGAs are valuable candidates for data signal processing in Wireless Sensor Networks [144]. Flexibility and performance efficiency of FPGAs are interesting characteristics for future use in WSN nodes. Reprogrammability, reconfigurability, performance and effective hardware/software codesign are powerful features of FPGA-based systems which makes of FPGA-based WSNs energy and performance aware platforms.

With respect to SRAM FPGAs, recent Flash based FPGAs can be considered as the second generation of FPGAs; overcoming the limits of non-volatility of previous SRAM-based FPGAs, Flash-based FPGAs are promising in low-power, real-time applications [23]. The ability to preserve the LUTs configuration after a shutdown, known as *live at power-up*, perfectly fits battery powered systems, where the device can be entirely shutdown to preserve energy. A SRAM FPGA must be completely reconfigured after each *power-up*, due to volatility nature of memories, causing a waste of time and power. Considering that the FPGA reconfiguration is one of the most power consuming activities of a FPGA, and considering that the time needed to reconfigure the complete device takes several milliseconds, a reconfiguration on each start up is unfeasible when data are sampled at medium/high frequencies.

This Section provides an evaluation of Flash-based FPGAs technology for novel WSN's nodes. Differently from SRAM-based FPGAs, thanks to *Live at power-up*, Flash-based FPGAs allows the system to effectively control the power consumption of the system at runtime. For this purpose, this Section presents a controller to manage energy consumption in Flash-based FPGAs systems. Dynamic energy consumption,

evaluated on a real case study with a real testbed, shows an overall energy consumption on the FPGA lower than 4mW, which allows the system to work with batteries.

3.2.1 Related Works

Advanced applications in Wireless Sensor Networks require high-frequency sampled data [144] [111] [27] for complex, accurate and reliable analysis. Increasing the efficiency of filtering data from sensors on a wireless sensor node is fundamental in order increase the range of applicability of wireless sensor networks.

Structure health monitoring requires real-time analysis of sensed data for a continuous monitoring of structure's condition. Work proposed in [111] presents an high-frequency distributed sensing system for structure monitoring application using MICA2 motes and MICA sensor boards. The system is able to record approximately 90 seconds of continuous data at 250Hz.

Detection of shooters in urban environment [144] requires high tolerance of multiple sensor failures, high accuracy and the ability to overcome multipath effects. The proposed approach extends the MICA2 motes with a multi-purpose acoustic sensor board designed with three independent acoustic channels and a Xilinx Spartan II FPGA, used for signal processing purposes. The system is able to detect the shooter with an accuracy of 1.3 meters and an average latency under 2 seconds. The paper does not consider any kind of power management techniques and the power consumption of the system is not reported.

A different kind of acoustic sensor network application is presented in [73]. The paper presents a system for the monitoring of amphibian populations in the monsoonal woodlands of northern Australia. The application requires sampling frequencies of $10kHz$ to differentiate the calls of cane toad from other 8 native frogs. Since Mica motes provide only $200Hz$ of sampling frequency, the authors increase the clock rate of Analog to Digital Converter to achieve a sampling frequency of $10kHz$, but no power consumption results are presented.

From the analysis of acoustic data from sensors it is possible to detect rock collapses. In [27] the authors present an hybrid wireless-wired monitoring system that samples data at 1kHz using a dsPIC24 microcontroller. Considering the filters on the data, such microcontroller limits the maximum sampling frequency to few kHz. The system is powered by batteries and photovoltaic energy. This application could benefit from the use of low power FPGAs.

A recent work in the field of Wireless Multimedia Sensor Networks (WMSN) [147] takes advantage of FPGAs to implement an improved CSMA/CA mechanism for IEEE 802.15.4 protocol to allow reliable and timeliness transmission of voice data. The architecture has been tested on a Xilinx Spartan-3E FPGA, performances results are presented but energy consumption is not taken into account. The authors of [167] present an FPGA-based Wireless Vision Sensor nodes. The architecture includes a microcontroller and an Altera EP2C35 FPGA to provide low-power HW image compression.

An important feature of FPGAs is the reconfigurability, which allow the system to be dynamically adapted to different scenarios. In [116], the authors present a reconfigurable WSN node. The node is implemented on the Altera Cyclone II FPGA, tested in a real case study. Similarly, an architecture for dynamic reconfiguration of advanced WSN node is presented in [134]. The authors illustrates how dynamic reconfiguration

Table 3.2: *SRAM vs Flash FPGAs*

	PROS	CONS
SRAM	• Fast programmability • Small configuration bitstreams • High performance • High chip density	• Need to be programmed on each power up • High power consumption
Flash	• Live at power up • Low static power consumption • Efficient power control mechanisms	• Low chip density • High cost

can be achieved on Flash-FPGA devices.

3.2.2 SRAM vs Flash FPGAs

A Field Programmable Gate Array (FPGA) is composed of a dense array of programmable components (such as memories, logic gates, DSPs, etc.). Thanks to FPGAs reconfigurable, high-performance, general-purpose architectures for WSNs can be designed. The ability to reprogram an FPGA (statically and dynamically), coupled with its parallel architecture, makes FPGAs an interesting device for digital signal processing on WSNs nodes. Reduced power consumption of modern FPGAs offer the opportunity to use HW processing in a WSN node that, differently from microcontrollers, allows the implementation of dedicated hardware cores.

Memories, used to store data and configuration information, dispersed on the FPGA, can be created using SRAM or Flash technology. In SRAM FPGAs, data and configuration are stored into volatile memories whose content is lost when the device is powered off. On the other hand, Flash FPGAs store the information in non-volatile memories, thus the content is kept even if the device is not powered. As previously mentioned, this is called *live at power-up* and is one of the main characteristics that makes Flash FPGA an enabling technology for low-power applications such as WSNs.

Table 3.2 analyzes pros and cons of SRAM and Flash FPGAs. Let us consider the following three power modes:

- **Active**: FPGA is working at full speed. This mode provides maximum power consumption and performance
- **Sleep**: clock and I/O ports are turned off, the internal state is maintained but the device is on.
- **Shutdown**: the FPGA is powered-off. In this mode, the power consumption of the device is zero

Figure 3.8 shows an example of sleep-on-sleep-shutdown cycle which provides a qualitative comparison between power consumption in SRAM and Flash FPGAs. In a WSN,

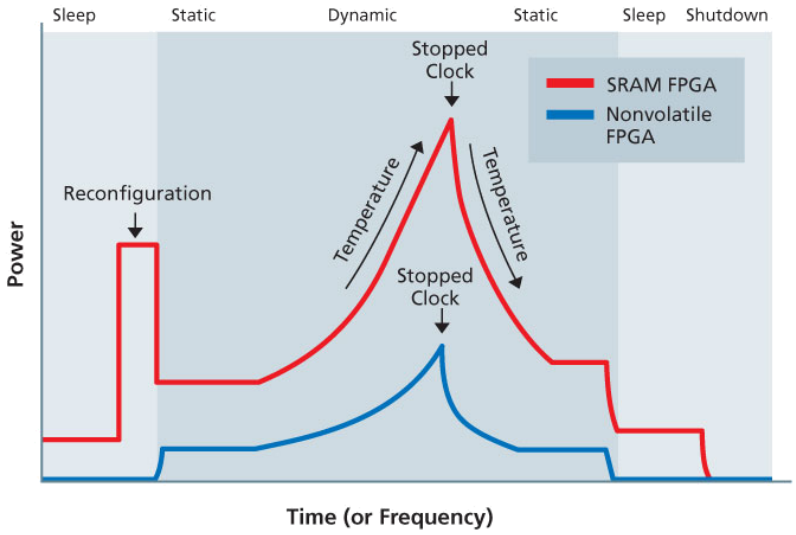


Figure 3.8: *A comparison between SRAM and Flash FPGAs in a sleep-on-sleep-shutdown cycle (From [31])*

this cycle is commonly repeated every sample received from sensors. Time (or frequency) depends on the sampled variable. At the beginning of the cycle, the FPGA is powered-on, thus it passes from shutdown-sleep to static. During this phase, SRAM FPGAs require energy and time to power-on and configure the device. In Flash-based FPGAs such transition is negligible in both power consumption and time, since it does not require any (re-)configuration. Static and dynamic power consumption are relatively higher in SRAM FPGAs since they are typically designed toward performance. As clock frequency increases, both power consumption and temperature increases accordingly in both cases. When the device has to be powered off (or it is required to enter in sleep mode), the clock is stopped, power consumption and temperature decreases. During the sleep phase, SRAM FPGAs require a certain amount of energy to keep data and configuration of the device, while Flash-FPGAs have a power consumption near to zero, thanks to the non-volatility of memories.

More formally:

- P_{on}: power consumption in *Active* mode;
- P_s: power consumption in *Sleep* mode;
- P_{off}: power consumption in *Shutdown* mode;
- $P_{A \to B}$: power required to switch from mode *A* to mode *B*, i.e. $P_{on \to s}$ is the power required to switch from *Active* to *Sleep* mode;
- t_{on}: average time in *Active* mode;
- t_s: average time in *Sleep* mode;
- t_{off}: average time in *Shutdown* mode;
- $t_{A \to B}$: time required to switch from mode *A* to mode *B*.

Considering an interval of

$$T = t_{on} + t_s + t_{off} + t_{on\to s} + t_{s\to off} + \sum_{A\in S}\sum_{B\in S} t_{A\to B}$$

the power consumption of the system is given by

$$P = \frac{P_{on}t_{on} + P_s t_s + P_{off}t_{off} + \sum_{A\in S}\sum_{B\in S} P_{A\to B}t_{A\to B}}{T} \quad (3.1)$$

where S indicates the set of available states (Active, Sleep and Shutdown). Under the same circumstances (same clock frequency, same design, etc.), overall power consumption is higher in SRAM FPGAs with respect to Flash FPGAs. In particular, power consumption of FPGAs during the sleep mode is extremely lower in Flash FPGAs with respect to SRAM FPGAs since SRAM technology requires energy to keep the configuration. Moreover,

$$t_{off\to on}^{SRAM} >> t_{off\to on}^{FLASH}$$

due to SRAM re-configuration. In WSNs, frequent sampling forces the system to switch from sleep to active mode (and viceversa) frequently, to gather and compute samples from sensors. If the sample frequency is defined with f_{sam} and the time required to compute a sample is t_c, the system works correctly if and only if:

$$\frac{1}{f_{sam}} < t_c + t_s + t_{on\to s} + t_{s\to on} \quad (3.2)$$

if sleep is used or

$$\frac{1}{f_{sam}} < t_c + t_s + t_{on\to off} + t_{off\to on} \quad (3.3)$$

if the device is shutdown instead of using sleep. In both cases, let us assume that $t_c = t_{on}$, thus the system is active if and only if it is computing data. Considering that

$$P_{on}^{SRAM} \simeq P_{on}^{FLASH}$$

and

$$P_s^{SRAM} \geq P_s^{FLASH}$$

due to SRAM technology, and considering the reconfiguration time

$$t_{off\to on}^{SRAM} >> t_{off\to on}^{FLASH}$$

SRAM FPGAs do not offer an effective solution for low-power application (w.r.t. Flash FPGAs), since in both sleep-on-sleep or shutdown-on-shutdown cycles, Flash FPGAs have a lower energy consumption.

As an example, please consider a configuration time of 100ms ($t_{off\to on}$), a negligible shutdown time ($t_{on\to off} = 0$) and a computing time of 1 ms (t_{on}), which corresponds to 100 thousand cycles at 10MHz of clock cycle, that is a great amount of time to process even complex data series. According to this information, an on-off cycle to save power with a SRAM FPGA is unfeasible for sampling rates higher than 10Hz, which throughput is too low to justify the use of a FPGA insead of using a microcontroller.

On the other side, in case the $t_{off \rightarrow on}$ period is reduced to few microseconds (i.e. 5 μs), the effective power consumption of the device is related to the activity of the device. In this configuration, the sampling frequency can rise up to 1 kHz.

In conclusion, SRAM FPGAs are not good candidates for low-power applications with periodical tasks, which is a typical situation in WSN systems.

3.2.3 FPGA-based WSN Node

Nowadays, not so many vendors produce Flash-based FPGAs. Microsemi produces different families of pure Flash-based FPGAs; in this Section only the IGLOO family is considered for the experiments. On the other hand, other vendors propose hybrid solutions, where SRAM FPGAs, equipped with Flash memory are able to guarantee *live at power-up* reconfiguring the FPGA at each start-up. Xilinx offers a Flash-based version of low-cost Spartan-3 FPGAs, called Spartan-3AN which couples high performance of leading-edge SRAM FPGAs with non-volatile memories. Similarly, LatticeXP FPGAs, from Lattice Semiconductor, use a combination of non-volatile FLASH cells and SRAM technology to guarantee *live at power-up*.

If *Active* and *Shutdown* modes are supported by every FPGA, the *Sleep* mode is supported only by recent Microsemi IGLOO FPGAs, some FPGAs of the Microsemi ProASIC3 family and LatticeXP FPGAs. Microsemi supports the *Sleep* mode with Flash*Freeze technology [22] and, similarly, LatticeXP FPGAs support it [92]. Although the presented architecture is focused and verified on Microsemi IGLOO FPGAs, the approach presented here can be easily extended to future Flash-based FPGAs supporting the *Sleep* mode.

Let us assume that the FPGA is able to switch from *Sleep* to *Active*, and vice versa, using a dedicated pin called **Sleep Pin**. *Sleep Pin* must be accessible from the system and from an external component (i.e. a microcontroller or an ADC). The FPGA switches from *Active* to *Sleep* when the pin goes high and switches back when the pin goes low. The transition to and from *Shutdown* mode needs an external circuitry that controls the power of the device but this is not considered here.

A WSN node is composed of sensors, actuators, a network interface, power supply unit (batteries, external, solar, etc.) and a processing unit. In case the digital signal processing requirements are demanding (such as for high throughput sensorial information), additional processing units are required. In a WSN node, FPGAs can be used to perform signal processing only (in this case an external microcontroller is required), or to manage both signal processing and operating system (a soft core on the FPGAs). Figure 3.9 shows the two aforementioned scenarios. On the top of the Figure, the microcontroller is external to the FPGA (like in [167]), while the second Figure shows the microcontroller implemented in the FPGA. The second solution offer more flexibility with respect to the first one, thus in the experimental results this architecture is used.

Thanks to this architecture, the FPGA can process the information from sensors (or the control for the actuator) completely in parallel. In fact, a dedicated IP core should be implemented for each sensor (or actuator) in order to exploit the performances of the FPGA. Using this strategy, an array of sensors for an event detector system (i.e. for landslide monitoring) can be easily implemented on the FPGA. In addition, as shown

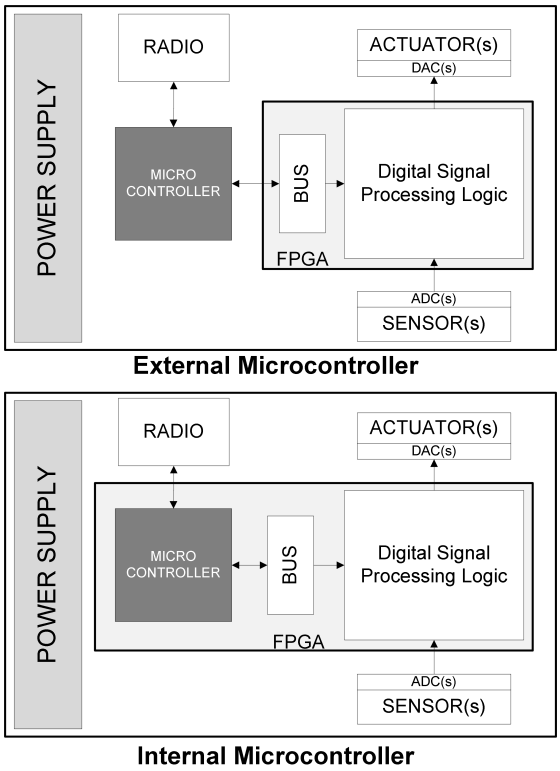


Figure 3.9: *Two FPGA-based WSN node's architectures*

in Section 3.2.1, Multimedia Wireless Sensor Networks [147] take advantage from the FPGA processing capabilities to improve the performance of communication.

Thanks to their programmability, FPGAs offer a unique opportunity to create an high performance, low power, general purpose WSN node. In fact, the node can be produced, according to the two architectures proposed in Figure 3.9 with no prior knowledge on the amount of sensors/actuators or signal processing algorithms that will run on it. The FPGA exposes a set of digital I/O pins that can be used to connect any type of sensor/actuator and can be configured to perform application specific processing.

Controlling Power Consumption

The *Sleep* mode allows the system to reduce the power consumption of the device by switching off all the input/output signals of the FPGA including clock and reset. Energy aware applications require system architectures able to directly and effectively control the *Sleep* mode at runtime. To reach this objective I developed an Intellectual Property (IP) called *Sleep IP* able to manage the FPGA during the *Sleep* mode, allowing the system to go into *Active* mode after a predefined period of time.

Considering that all the input/output pins are blocked during the Sleep mode, none of the clock/reset/interrupt can be seen by the FPGA. Microsemi IGLOO FPGAs are equipped with an internal digital pin called *Freeze Pin*, which allows the system to enter the *Sleep* (Freeze) mode when the pin is set to one, and to exit when the pin is set to zero.

The proposed component is shown in Figure 3.10 and is characterized by the fol-

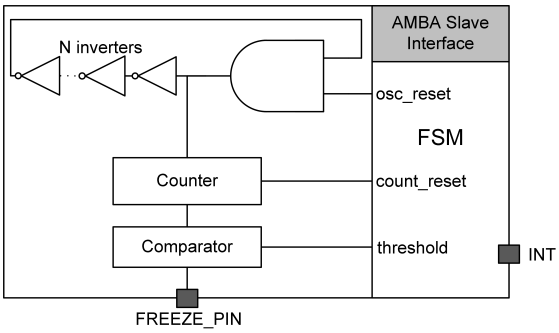


Figure 3.10: *The ring oscillator is used to create a clock signal during the Freeze mode*

lowing input/output ports:

- AMBA SLAVE SIGNALS [IN/OUT]: to connect the component to the AMBA bus;
- FREEZE_PIN [OUT]: to control the *Sleep mode*;
- INT [IN]: a pin to send a direct request to enter into the *Sleep* Mode.

and the following registers are accessible through the bus:

- THRESHOLD [READ/WRITE]: used to define the number of clock cycles in which the system is in *Sleep mode*;
- ACTIVE [WRITE]: if a 1 is written into, the FPGA switch from *Active* to *Sleep mode*;
- INT_ENABLE [READ/WRITE]: used to define if the INT port is enabled. If zero, no INT signal will be considered;
- BUS_ENABLE [READ/WRITE]: used to define if the ACTIVE register must be considered to switch the device to the *Sleep mode*.

Since the system does not have any clock input, we instantiate a chain of inverters in order to generate a clock signal during the *Sleep mode* to count how many time is spent in this mode in order to allow the FPGA to return back to *Active mode* correctly. The number of inverters must be odd to generate an oscillation. Such system oscillates at a certain frequency given by the amount of inverters of the chain. Higher is the number of inverters, lower is the generated frequency. The generated clock increments a *counter* which value is used by a *comparator* to detect if the FPGA must return to *Active mode*. The *comparator* check if the input value overcomes a given threshold. In this case, the *Freeze PIN* is set to zero, forcing the FPGA to exit from the Sleep mode.

To enter into the Sleep mode, an external pin (INT) is used; when the value of this pin moves from 0 to 1, the system enters in the Sleep mode. Another, way to enter into sleep mode is to write a one into a specific register.

The component has an AMBA slave interface, used to configure the core during the active period. The master of the AMBA bus (typically the processor) can change the value of the threshold or force the system to enter in Sleep mode without using the INT port. The FSM is under the domain clock of the AMBA bus, while both the counter and the comparator works under the domain clock of the oscillator. During the Active

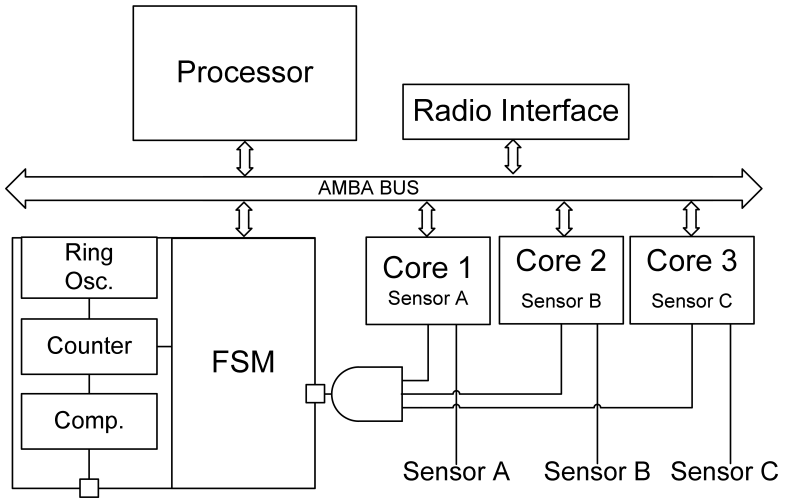


Figure 3.11: *A complete bus-based system*

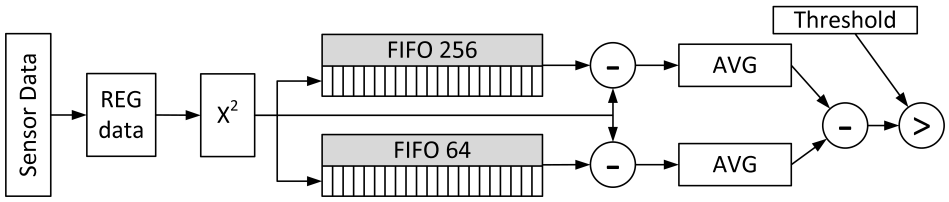


Figure 3.12: *The circuit used to detect a landslide event*

mode, the OSC_RESET is set to 0, thus the inverter chain does not oscillate; it reduces the energy consumption. According to that, the FSM and the counter/comparator do not evolve at the same time, since the inverter chain oscillates only during the sleep period, when the system clock is stopped, and vice versa.

The IP can be used in various custom or bus-based designs. Generally, the AMBA bus interface can be used to reprogram the parameters of the IP according to the application's needs. An application can require to freeze the FPGA for a certain period of time writing a 1 to the ACTIVE register. At the same time, custom component can be directly connected to the IP, using the INT port, in order to have a way to control the power, even a processor.

A complete system is proposed in Figure 3.11. The whole system is implemented on the FPGA, according to the second example of Figure 3.9. It includes a processor, which is the master on the bus, a set of cores connected to different sensors on the board and connected to the INT pin of the *Sleep IP*. The INT signals of the cores are connected to an AND port in order to force the switch if and only if all of them finished their computation. This is a common connection when multiple components control the FPGA modes since it is not desired that the FPGA switches to the *Sleep mode* when some component is still computing data. A slave radio interface is connected to the bus in order to guarantee a connection with an external radio device.

The presented *Sleep IP* and the architecture in Figure 3.11 are intended to be generic and reusable in many WSN applications. Regardless the amount of sensors connected to the FPGA, the use of a set of separate cores for the filtering and signal processing is

crucial to exploit parallelism on the FPGA. Moreover, HW cores for digital signal processing reduce the processing load on the microcontroller that can be used to manage network communications and user's applications.

Application Case Study

The proposed architecture has been implemented on a real case study in order to validate the approach with valuable experiments. The reference application is an event detector architecture for landslide applications, and the algorithm used to detect an event is based on [27]. The algorithm compares recent sampled data with historical data. In this architecture, the system stores the updated value of the average, made on the last 256 samples (long average), with the last 64 samples (short average).

The landslide event detector core is depicted in Figure 3.12. It takes sensor data as input, squared it and stores the value in two FIFOs (one for short sample and another one for long sample). The absolute value of the average of the FIFOs is kept updated by adding the difference between the output value and the input value. The average is compared and, if and only if the difference between the averages overcomes a predefined threshold, an event message is sent to the processor. The implemented circuit spends 12 clock cycles to compute a new sample and to establish if an event occurs. Since data come from sensors at a predefined sampling frequency, usually more than three orders of magnitude slower than the system's frequency, in more than 95% of the time, the FPGA is idle. In this case, and in other related cases, using the *Sleep* mode between consecutive samples can be an effective way to increase the power efficiency of sensor nodes.

In this case study the FPGA filters data from the sensors, sending data to the microprocessor only if an event is detected. Three accelerometer's sensors are connected to the FPGA using the I/O pins of the FPGA. In order to test the ability of the FPGA to switch from and to the *Sleep* mode, no interrupt signal from the ADC-chain is used.

The system has been implemented on a Microsemi IGLOO AGL600-FGG256 FPGA [21]. The processor is the IP core 8051s [20] and a Digimesh XBee module [51] has been used for wireless communications.

Experimental Results

To prove the efficacy of the approach, three experiments have been performed: the first experiment validates the whole system, the second experiment evaluates the power consumption with various values of sampling frequency, system frequency and number of inverters, and the third experiment evaluates the power consumption of the proposed *Sleep IP*.

In the first experiment a landslide event is simulated in order to check if the system is able to correctly and timely deliver the information through the XBee interface. Another XBee interface has been connected to a PC, waiting for data. In this experiment, clock frequency of the FPGA is set to 10 MHz and sampling frequency is equal to 1kHz. As expected, if no vibrations are applied on the system, no messages are sent to the system. Once a vibration is applied on the accelerometers, a package is immediately sent to the PC. As aforementioned, this test has been conducted to validate if the system is able to work as a whole. A Digimesh XBee module has been used on both

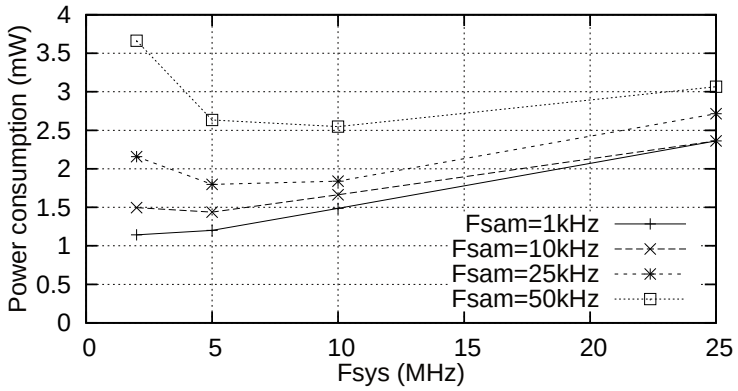


Figure 3.13: *Power consumption of the simulated circuit with fixed f_{sam}*

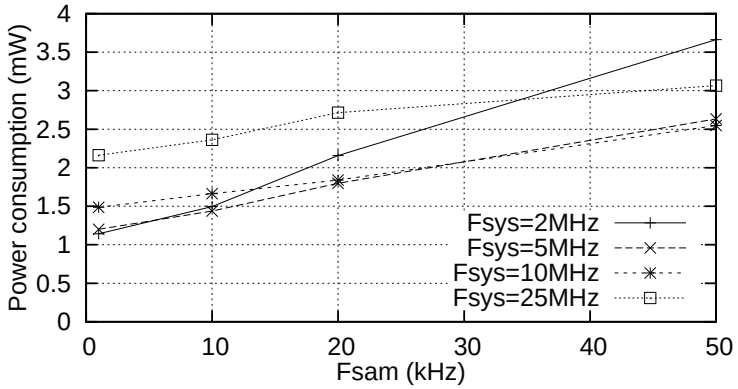


Figure 3.14: *Power consumption of the simulated circuit with fixed f_{sys}*

sides (FPGA and PC).

The second experiment aims at evaluating the correlation between parameters and power consumption. The parameters are: the sampling frequency (f_{sam}) and the system's clock frequency (f_{sys}). In this experiment, only the power consumption of the FPGA is considered, and no messages over the XBee are sent and no stimulus on the accelerometers is applied. Testing this configuration is important since a landslide event is (hopefully) very rare. Moreover, in case a landslide event occurs, the power consumption used to deliver the information is not critical as delivering it correctly. The system has been tested varying the parameters as follows:

- f_{sam} = 1kHz, 10kHz, 20kHz, 50kHz
- f_{sys} = 2MHz, 5MHz, 10MHz, 25MHz

In all the experiments, a chain of 401 inverters is used. The results of the experiments are plotted in Figure 3.13 and 3.14. The results show that minimum energy consumption is achieved combining f_{sam} and f_{sys} properly. For example, for $f_{sam} = 1kHz$, $f_{sys} \simeq 2MHz$ is a good solution, while for $f_{sam} = 50kHz$, the optimal solution is $f_{sys} \simeq 10MHz$. Considering that $P_{on} > P_s$, computing the data as soon as possible is important. As sampling frequency increases, higher system frequencies offer a better energy solution.

Regarding sampling frequency (Figure 3.14), as expected, higher is the sampling frequency, higher is the power consumption since frequent wake-ups are required. In addition, the solution with $f_{sys} = 2MHz$ crosses all the others for increasing values of the sampling frequency, and a similar behavior can be observed on $f_{sys} = 5MHz$. Low values of f_{sys} increase t_{on} (since the amount of clock cycles is fixed), while higher values of f_{sys} increases P_{on}. Considering the Equation 1, a tradeoff between f_{sys} and f_{sam} is needed to keep the overall power consumption of the system as low as possible.

It is important to notice that, in all the configurations, even with high sampling rates, average energy consumption of the FPGA is below 4 mW, that allows the system to be battery powered.

The third experiment aims at estimating the relationship between the number of inverters used in the ring oscillator and the power consumption of the ring oscillator and the counter. Power consumption measurements were performed on only the Sleep IP, excluding the other components, that are off during the sleep phase. Figure 3.15 shows the power consumption of the device with respect to the number of inverters composing the ring oscillator. It shows a minimum power consumption of than $220\mu W$ with 2000 inverters. Considering that the Microsemi IGLOO AGL600v5 consumes, in Sleep (Freeze) mode, more or less $36\mu W$ [21], the minimum overhead of the ring oscillator is about $180\mu W$.

Regarding the relationship between frequency and number of inverters, we measure $2.5ns$ of delay for each inverter in the chain, obtaining the following relation:

$$f_{osc} = \frac{1}{2.5nsN_{inv}} \tag{3.4}$$

where N_{inv} represents the number of inverters in the chain. According to our experiments, three inverters generate $133MHz$ of clock frequency, 400 inverters create a frequency of $1MHz$, and 2000 inverters creates $200kHz$ of clock frequency.

Considering that the inverter switches if the value of the input changes, the average switching activity of the inverter chain is the same for each length of the chain. What changes is the power consumption of the counter, since it counts at different frequencies. It explains why the power consumption decreases when the number of inverters increases.

The size of the circuit linearly depends on the number of inverters used to generate the clock. Synthesis results shows the following relation between the number of Inverters ($Ninv$) and the used Core Cells (C) (a metric used by the synthesizer to represent the number of logic primitives in the design, i.e. logic gates or memory cells):

$$C = 200 + N_{inv} \tag{3.5}$$

where 200 is the size of the counter and the management logic. The FPGA used here, the AGL600v5, has 13824 Core Cells. According the these results, a chain of 500 inverters is a reasonable tradeoff between power consumption and area occupation. In such situation, the power consumption of the *Sleep* mode is about $330\mu W$ with an area occupation of 5% on an AGL600v5.

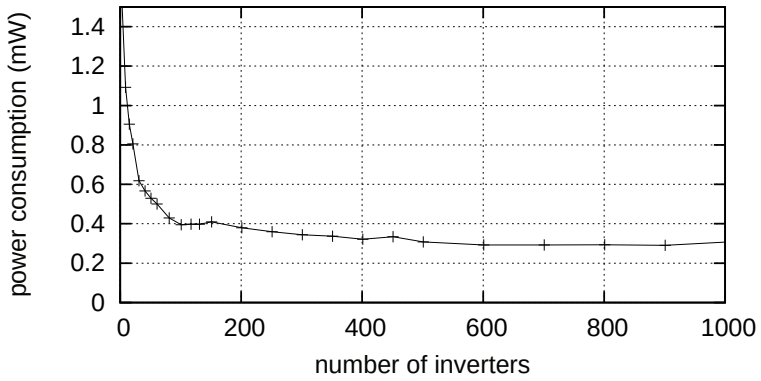


Figure 3.15: *Power consumption of the ring oscillator with respect to the number of inverter used to implement the ring oscillator*

3.3 Concluding Remarks

This Chapter provides a list of microcontroller-based platforms and a proposed FPGA-based node. The purpose of this Section was to provide the reader a list of (traditional) available platforms, and present an alternative approach in WSN's node design. By analyzing the available platforms it is clear that the number of design alternatives is high, and the identification of the optimal design is not trivial. The innovative contribution of this chapter is the study of a FPGA-based sensor node, that I developed and tested on a real case study. Next Chapter introduces a novel Design Flow for Wireless Sensor Networks in order to guide the designer during the design process toward the optimal design solution.

4 The Proposed Design Flow

A complete and satisfactory design of a working Wireless Sensor Network requires the definition of three aspects: the position of the nodes (**topology**), their hardware and network configuration (**HW/NET Configuration)** and the software (**SW Development**). All these components are mandatory to have a working WSN and the constraints of a WSN makes WSN design very challenging such as it is even described as requiring "2.5 Ph.D's" [72]. The identification of a trade-off among power, performance and reliability needs an efficient engineering work.

This Chapter introduces a WSN-specific design flow aiming at guiding the developer through the design process from the high-level application specification, design requirements and constraints to the final deployment. The proposed design flow has been specifically designed to meet WSN requirements, and all the processes that compose such flow are focused on specific parts of the final design such as their *topology*, *hardware*, *software*, etc.

It has been designed for both manual and computer aided design, in fact input/output interfaces have been specified to integrate automated optimization tools that support the user during the design process. This Chapter illustrates also examples on how to use optimization tools into the design flow and presents an innovative optimization technique based on Markov Decision Processes.

4.1 An Iterative Three-Step Design Flow

The design process of a Wireless Sensor Network is composed of three phases: **placement**, **HW/NET configuration** and **SW development**. Although these phases will be presented to be executed in order, it is a good practice to use a Spiral Model approach [35], where all the design flow is re-executed until no further optimizations are possible. For the sake of simplicity, this Section illustrates a single iteration of the design flow.

The **placement** phase consists in the identification of the optimal position of the nodes to ensure coverage and connectivity requirements. The position of the nodes highly affects the performance and the kind of network organization to be used as well as their power consumption.

Placement problem has been deeply analyzed in the past decade [166], resulting in

the development of several placement algorithms. As an example, relay node placement algorithms detect the optimal position of relay nodes (routing nodes) such that the number of relay nodes is minimized, connectivity is ensured and reliability is maximized. Relay node placement has been analyzed for large scale [150], constrained [113], fault-tolerant [66] placement. Although the problem has been deeply analyzed, it remains an open issue in WSN design and a general solution has not been identified yet, requiring customized algorithms for different application fields.

Hardware and network design is a crucial aspect to design efficient and reliable sensor networks. Nowadays, many low-cost, low-energy hardware platforms have been proposed and adopted in many application fields. The **HW/NET configuration** phase consists in the identification of the optimal hardware and network configuration such that the resulting system performs as expected. In this phase, each component of the node (processing unit, MAC layer, routing protocol, etc.) must be selected among a large set of different design alternatives. Moreover, each component can be further tuned with a large set of configuration's parameters (i.e. memory size, radio TX power, etc.). The identification of the optimal configuration is extremely important to design efficient WSNs.

When topology and hardware/network configuration have been defined, the designer can proceed with the development of the application. **SW development** consists in the design and implementation of the software components able to implement the desired application's functionalities. Applications for WSNs typically read data from devices, perform some signal processing and transmit it to a central node. Software development for WSN is a very active research field and many solutions such as operating systems, programming languages or software abstractions have been proposed, developed and applied in various application fields [115]. Although many solutions have been proposed, the specific application's definition is usually technology-dependent, thus applications cannot be easily ported among different devices. A generic software model is preferred for a better hardware/software codesign.

Figure 4.1 illustrates the proposed design flow. The design problem has been divided in two distinct functional part: the application definition (or application-specific aspects) and the architecture and network definition. Note that architecture and network definition are not application-independent since their customization depends on both sensing position and sw specification.

The design flow takes, as input, a set of information concerning *application specification*, *the constraints*, *requirements* and *metrics*. These information guide the design space exploration and the optimizations such that the resulting design, ready for deployment, is correct with respect to the user requirements and constraints. The design of a WSN starts from **sensing coverage**. It defines the position and the kind of sensors in the network such that all the data required in the application are correctly gathered from the environment. Next, the **SW development** process defines a high-level specification of the software that runs on the network according to the given sensor placement. At the same time, sensing coverage completion allows the designer to proceed with the evaluation of the **network connectivity** and the **hardware design** processes. The first is used to place intermediate (relay, secondary, ...) nodes in order to have a connected network, while the second specifies the hardware platforms that best suits the user needs. At this point, given the software and hardware specifications, it is possible

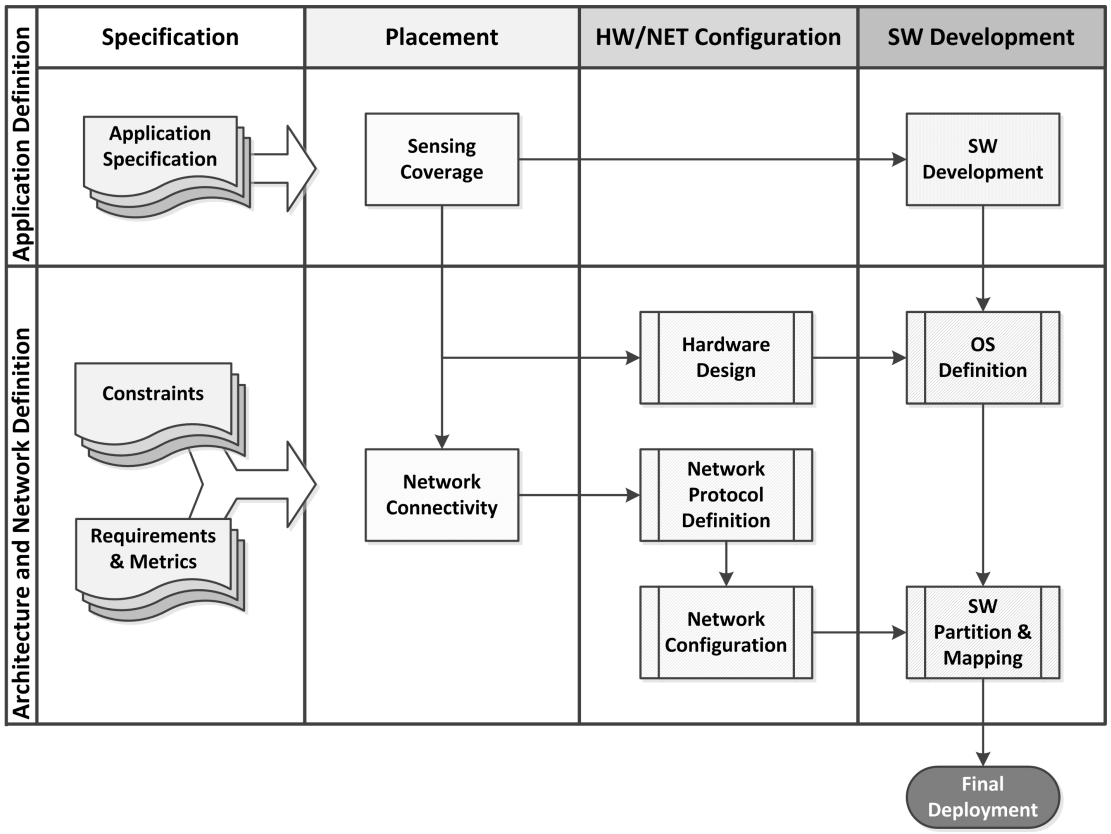


Figure 4.1: *The Proposed Design Flow*

to define the operating system (**OS definition**). At the same time it is possible to define which protocols (MAC, routing, transport, ...) will provide good performance from the given network topology (**Network Protocol Definition**), then the given protocols should be configured accordingly (**Network Configuration**). Once hardware, network and OS have been defined, the software must be translated into a compilable language, partitioned and mapped on the nodes (**SW Partition & Mapping**). All these processes can be repeated several times to refine the solution. Once the final design has been identified, it is ready for the **final deployment**.

In the remaining of the section, a more detailed definition of each process is given.

SENSING COVERAGE

This process is in charge of defining which sensors are needed, how many and in which position these must be placed. It takes high-level application definitions such as *"[...] the system must be able to track an object in the area of interest [...]"*, and provides a specification of the required sensors (such as *nine cameras, two microphones and five proximity sensors*) and their position in the three-dimensional space. Figure 4.2 illustrates the detail of this process. It takes application-specific requirements as input and provides two distinct outputs: the sensors list (which sensor is required) and the sensors position (where sensors are located in the 3D space). The specification of the application requirements must be as clear as possible in order to place the right amount of sensors in the right position. A specification can be, for example, *"place*

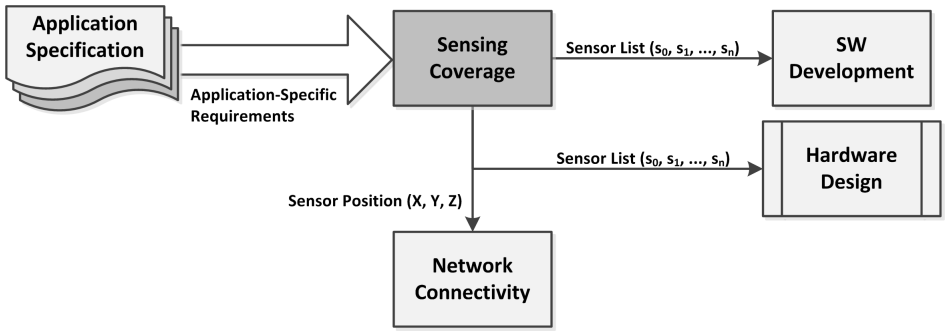


Figure 4.2: *Sensing Coverage Process*

the sensors such as the temperature of the area is read by at least three sensors and the distance between sensors must be at least 10 meters. This specification tells the user to have a three-sensor redundancy, but specifies that these measurements must be taken from different positions (at least 10 meters away). With this specification, even an automatic algorithm can be used. However, in some cases, the position of the sensor is pre-determined (i.e. *place a 3-lead ECG sensing environment*).

Considering the output, for *SW development* and *Hardware design*, the required information is the list of sensors, not their exact position. In fact, for *SW development* the list of sensors is mandatory in order to know which data can be used, and for *HW design*, the list of sensors is mandatory to know how to design the sensing nodes. On the other side, the *Network Connectivity* process requires the exact position of nodes in order to detect the position nodes of the additional nodes to guarantee network connectivity; the kind of sensors (temperature, humidity, etc.) is marginal for this process.

A correct definition of the sensing position is extremely important for two reasons: first, the sensors must be able to detect the phenomenon of interest correctly and, second, the position of the sensors is a strict constraint for the placement, thus it must be defined carefully to design an efficient network. This process is application-dependent, since the designers (or the algorithms) must know the application's characteristics and peculiarities. Therefore, algorithms automating this task should be specifically designed for the application [169] [166]. A new alternative approach, is given in Section 6.

SW Development

In this process the user must define the application assuming that the system is not composed of several nodes, but is a whole system whose inputs are sensors and outputs are actuators. This abstraction allows the user to define a design-independent definition of the software, thus the user should not optimize the code for specific hardware or networks. The idea behind this process is to provide a general definition of how the application should work; it will be the objective of the *SW partition and mapping* process to define how the code must be executed on the network.

The overview of the process is given in Figure 4.3. The process takes the list of sensors identified in *Sensing Coverage* and the application-specific requirements. Differently from all the other processes, this is usually executed manually since it results in a piece of code that satisfies the application requirements.

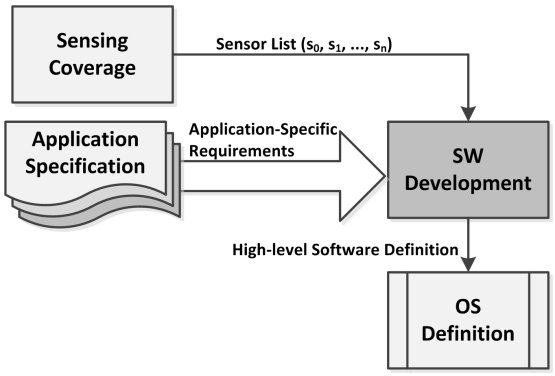


Figure 4.3: *Software Development Process*

The output of this process is a high-level definition of the software application that will run on the nodes. To perform this task the language used to define the application should be chosen having in mind that it should be translated into a compilable language automatically. For such a reason, in this thesis, a general purpose script is proposed; an example of the proposed language is given in Algorithm 1

Algorithm 1: prova

```
a = read("temperature","node_0")
b = read("temperature","node_1")
c = (a + b) / 2

  if c < 18 then
    write("heat","node_2",1)
  else
    write("heat","node_2",0)
  end if

notify(a,"sink")
notify(b,"sink")
```

The purpose of this piece of code should be immediate to the reader. The code states that the temperatures should be read from *node_0* and *node_1* respectively, then its average is computed in c, and compared to a threshold (18). In case the temperature is below 18, an actuator is activated (*heat* to 1), otherwise it is switched off. At the end both temperatures are sent to the *sink* node (that will be defined in the *network connectivity* process). Note that the position of the sensors have been identified in the previous process.

Next Section will illustrate how to convert this piece of code into a DFG, a useful mathematical representation that will be used at the end of this Chapter by the *SW partition and mapping* process.

NETWORK CONNECTIVITY

Connectivity of the network is extremely important to meet design objectives. This process is responsible for placing the minimum number of nodes that satisfy the connectivity requirements. Requirements and constraints can be specified in many ways according to the design needs. Figure 4.4 illustrates the relationship with the other

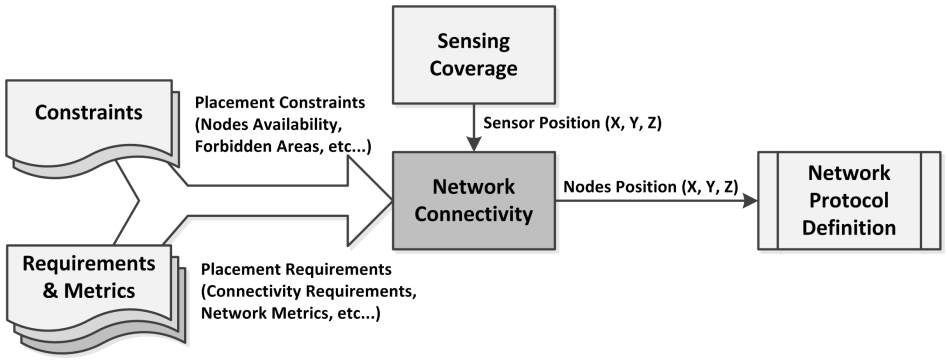


Figure 4.4: *Network Connectivity Process*

processes. It takes, as first input, the position of the sensors identified by *Sensing Coverage*; it represents a strong constraint since these nodes cannot be placed elsewhere. Then it takes the placement constraints and requirements. The first indicates, i.e., the amount of available nodes, the presence of forbidden areas, etc. The latter indicates the connectivity requirements or the metrics to optimize during the placement. An example of placement requirement is: *create a network such that all the nodes can communicate with each other using at least two independent paths*. Moreover, connectivity requirements can take into consideration fault tolerance: *create a network such that, even in presence of three node-faults, the network remains connected*. Differently from *sensing coverage*, this process is application independent, since it does not require any knowledge on the final purpose of the network to operate.

It produces the exact position of nodes (in addition to the position of sensing nodes) in the network. The position can be determined automatically or manually, according to the preferences of the user. Several placement algorithms have been proposed in literature [150] [113] [166] [34] [133] [66], thus no additional techniques will be provided in this thesis. Note that in the first iteration of the design flow, no network algorithms have been identified, so the algorithms must make assumptions on future networks organizations (plain, two-tier, cluster-based, etc...); in any case, the other iterations will converge to optimal solutions.

HARDWARE DESIGN

In this process the designer should identify or create a hardware platform that best fits its needs. The design of the network interface is not included in this process, but has been split in other two processes. Instead, this process is focused in identifying the platform to compute sensorial data and/or perform complex operations. Here the designer must decide to use a microcontroller-based general purpose platform (see Section 3.1) or a custom solution (such as ASIC or FPGA-based platforms). It is important to execute this process before the definition of the operating system since the OS is usually not supported by all the platforms, thus the hardware platform is a constraint to the next processes. Inverting the order of processes (perform OS definition prior the hardware definition) usually leads to non-optimal solutions since the definition of the correct hardware platform is more important then the specific OS with respect to energy, performance and reliability, thus it is discouraged.

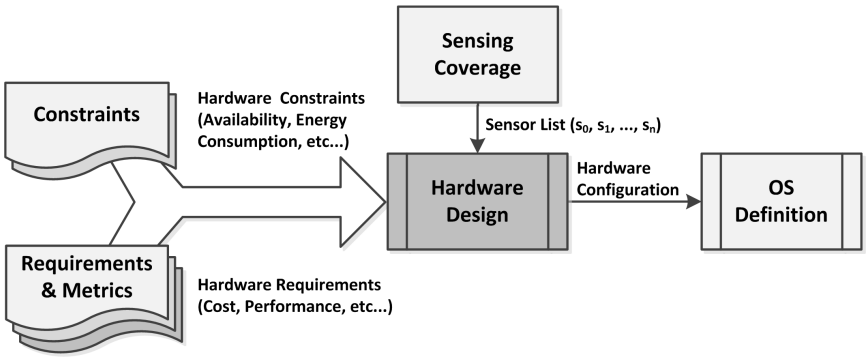


Figure 4.5: *Hardware Design Process*

Figure 4.5 illustrates the process. As usual, the process takes constraints and requirements as input. In the specific case, constraints can be related to energy consumption, stock availability or cost, that must be respected in order to have a design that will be accepted by the customer, while requirements are those aspects that must be optimized such as cost, performance, etc. The other input, is the list of sensors required by the application; considering that this process produces the final hardware configuration, sensors are part of this output. The output is the final hardware configuration of the nodes, network interfaces excluded. It includes the processing and sensing elements as well as power sources, harvesting techniques and memories.

NETWORK PROTOCOL DEFINITION

Networking is so relevant in a WSN so that the identification of the optimal protocols is extremely important to obtain optimal design solutions. The objective of this process is the identification of the radio, MAC and routing protocols that are most suitable for the given design requirements and metrics. Figure 4.6 illustrates the process. This process can be constrained by the availability of licenses or laws (i.e. only 2.4 GHz radio are allowed) and must optimize several metrics such as energy, performance or reliability. However, this process leads to the definition of an high-level network organization, thus no fine-grained optimizations are possible; in fact, this process typically uses models instead of simulations to evaluate the solutions. The other input is the position of the nodes, that dramatically affects the kind of network protocol that will be chosen. The output, as aforementioned, is an coarse-grained high-level network organization.

An example of output of this process is:

- **Radio**: CC2420, 2.4 GHz radio
- **MAC Layer**: IEEE 802.15.4
- **Routing**: LEACH protocol + AODV for routing through cluster-heads

Although this process is theoretically independent from *hardware design*, the processes have a relationship, since the definition of a hardware platform could constraint the identification of network interfaces and vice versa. For instance, if the designer chooses the SHIMMER platform, the radio cannot be changed. Vice versa, if the designer decides to use a specific radio, the available platforms that support the chosen radio are limited. The designer is free to decide the order of these two phases.

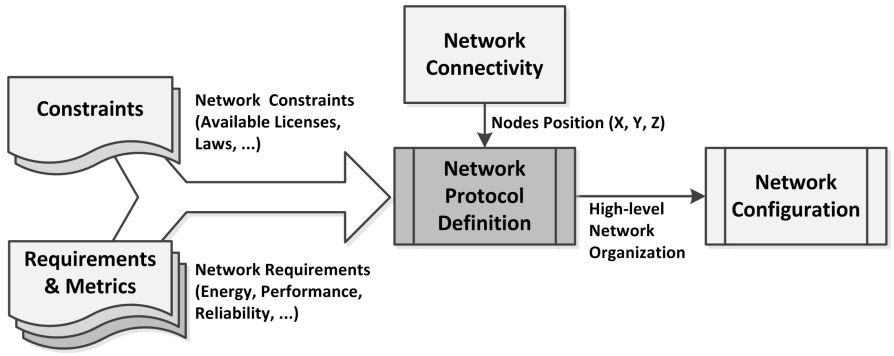


Figure 4.6: *Network Protocol Definition Process*

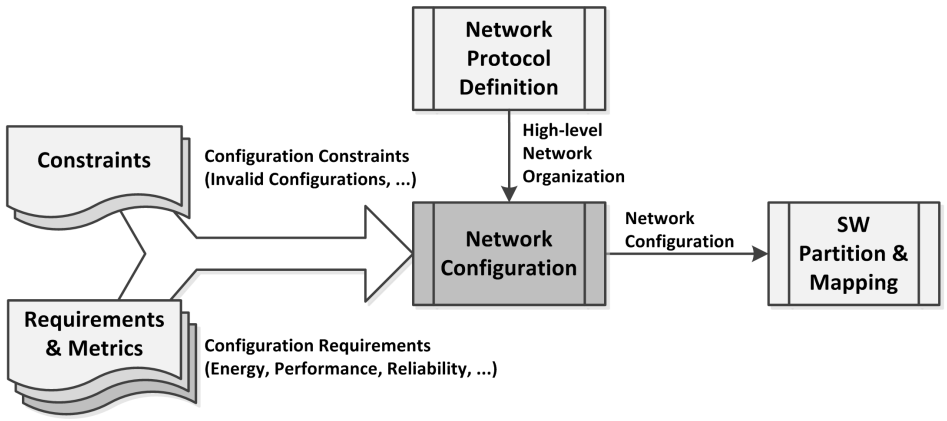


Figure 4.7: *Network Configuration Process*

NETWORK CONFIGURATION

Once the network protocols have been identified, they must be configured in order to perform optimally. All the network protocols, MAC protocols in particular, are characterized by tens of parameters whose configuration is a difficult task for non-experts, thus it should be performed automatically; this thesis presents several examples on how to configure network protocols automatically. There could be several constraints that specify invalid configurations: in IEEE 802.15.4, for instance, the value of the *FrameOrder* must be higher than the value of the *BeaconOrder*. Requirements are similar to those specified in *Network Protocol Definition*, but in this case they can be effectively optimized. The other input comes from *Network Protocol Definition* that specifies the protocols and the network organization to be used in the project. These two processes have been divided since a combined optimization could be practically unfeasible; however, the user can decide to perform these two processes simultaneously. The output of this process is a detailed network configuration: all the parameters have a specific value. The process is depicted in Figure 4.7.

Considering that most of the designers are not expert in telecommunication, this process should be performed automatically. This Chapter and in Chapter 6 provide some example on how to perform this automatically.

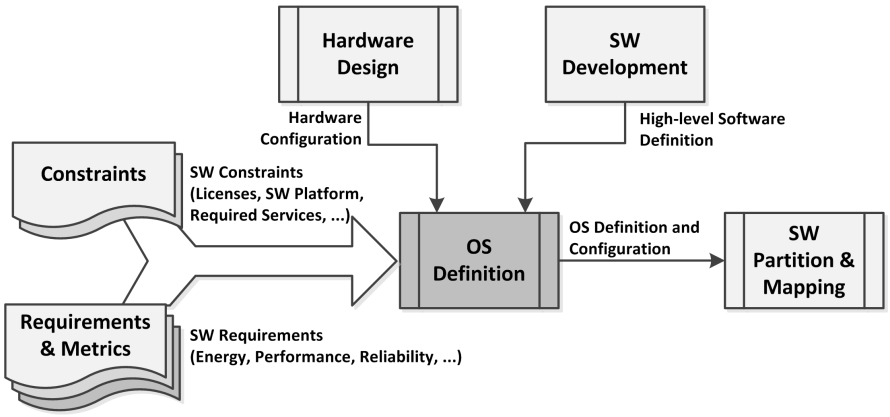


Figure 4.8: *Operating System Definition Process*

OS DEFINITION

The definition of the operating system (or its absence) is important to define how the application runs on the nodes and how the code must be translated and compiled. This process, depicted in Figure 4.8, identifies the right OS and its configuration according to the given constraints and requirements. There are two main sources of constraints: external constraints or hardware constraints. The external constraints can be defined at the beginning of the project and can depend on external aspects like licenses, required services or preferred SW platforms. In fact, the definition of the Operating System could even be constrained, for instance, by the preferences of the maintainers who could prefer one technology instead of another. On the other hand, hardware constraints come from the *Hardware Design* process, that previously defined the hardware platform which determines how the code can be executed on the node. For example, if the previous phase decides to use the *Arduino FIO* (see Chapter 3), TinyOS cannot be used as well as Java.

The high-level software definition, provided by the *SW development* process is useful not to decide which OS to use, but how it should be configured to optimally run the given code. In fact, for example, if the software requires an *automatic node-discovery*, the OS should support it and must configure it correctly.

At the end, the process generates a specification of the chosen OS and its configuration (in terms of parameters and required services).

SW PARTITION AND MAPPING

This is the last process of the design flow. The objective of this process is to take the software specification, the position of the nodes, their hardware, network and OS definition, and provides an optimal partition and mapping of the application code on the nodes. In other terms, this process must define exactly how the code must be executed on the nodes, in particular, which piece of code must be executed by each node. The granularity of the partition can be determined by the user and can be defined as task-level or instruction-level.

This process, illustrated in Figure 4.9, takes several inputs such as high-level software specification, constraints, requirements, network configuration, node placement,

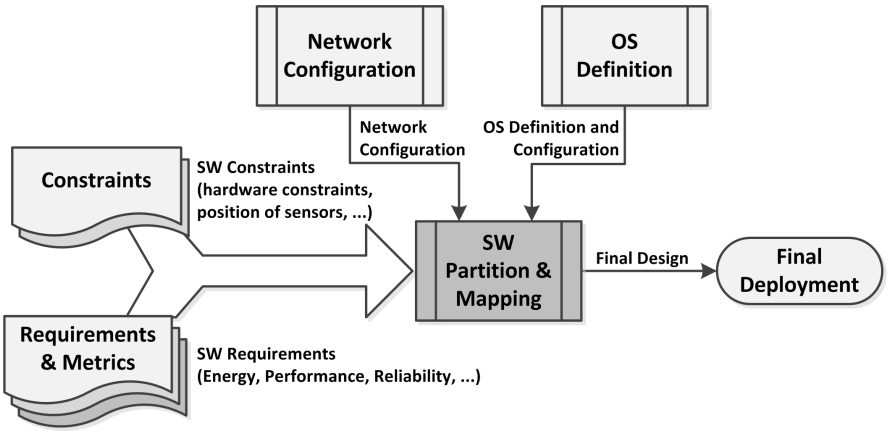


Figure 4.9: *Software Partition and Mapping*

OS details and hardware specification, and provides, as output, the final design ready to be deployed. The process, detailed in Section 4.5.1, takes the high-level software specification (defined in *SW development* process), partition the code into sub-tasks and maps these tasks on the node, considering the execution and transmission costs. The mapping is extremely important to ensure an efficient execution of the code. In fact, it is a bad practice to put all the tasks on a single node, since it can be unfeasible to transmit data wirelessly due to channel overutilization. Moreover, this process translates the high-level software definition, specified in *SW development*, into a compilable piece of code that will be deployed to run on the nodes.

FINAL DEPLOYMENT

Once the final design has been identified, the system is ready for the physical deployment. It involves the creation (if required) and configuration of the nodes and their physical deployment. This process has been defined *outside* the design flow since it is not part of the design process, but a consequence of the design.

The next Section illustrates the mathematical spaces where the various parts of the design flow operate. It provides a clear view on the complexity of the automated design space exploration and, at the same time, would like to be a reference point for the people working in this field such as all the algorithms will share the same structures, thus providing compatibility.

4.1.1 Implemented Design Flow

The design of WSN consists of the definition of a large number of elements such as: position of nodes, network configuration, hardware configuration, software development, etc. To develop an optimized WSN, the design should master all the aspects related to a WSN, including application-specific optimizations and algorithm. Moreover, the design process must take into account the time-to-market, that is constantly critic in today's markets, thus the need of time and cost effective design methodologies are required.

The actors involved in the design of a WSN are:

- **Application/DSP Expert**: sensorial data are extremely important in a WSN. This actor is involved in the design of application-dependent digital signal processing algorithms that will run on the node. Their tasks are: definition of position of sensors in the environment and the design of sensor's filters and actuator's controllers.
- **Network Designer**: this actor is responsible for the design and configuration of the communication's layer of the network. Sometimes, new protocols can be designed to have an optimized design. It includes: radio, MAC, routing protocols and network layers.
- **Hardware Designer**: this actor is responsible for the design and configuration of nodes' hardware. It includes: the design of application-specific platforms, the identification and configuration of the best platform among several available platforms (see Chapter), etc.
- **Software Engineer**: this actor is responsible for the design of the software architecture of the WSN. He is involved in: the identification and configuration of the operating system, partition, mapping and compilation of the distributed software in the network.

Figure 4.10 maps the roles and responsibilities of these actors in the design flow proposed in Chapter 4. The strong relationship among inputs and outputs of design flow's processes require an active and effective collaboration in the development team. Moreover, remembering that the design flow iterates on all the activities until no further optimizations are possible, the involvement in the design process lasts until the design is deployed.

Since the management and maintenance of such heterogeneous teams is time and cost expensive, the development of automated design tools could effectively tackle these problems and reduce design costs.

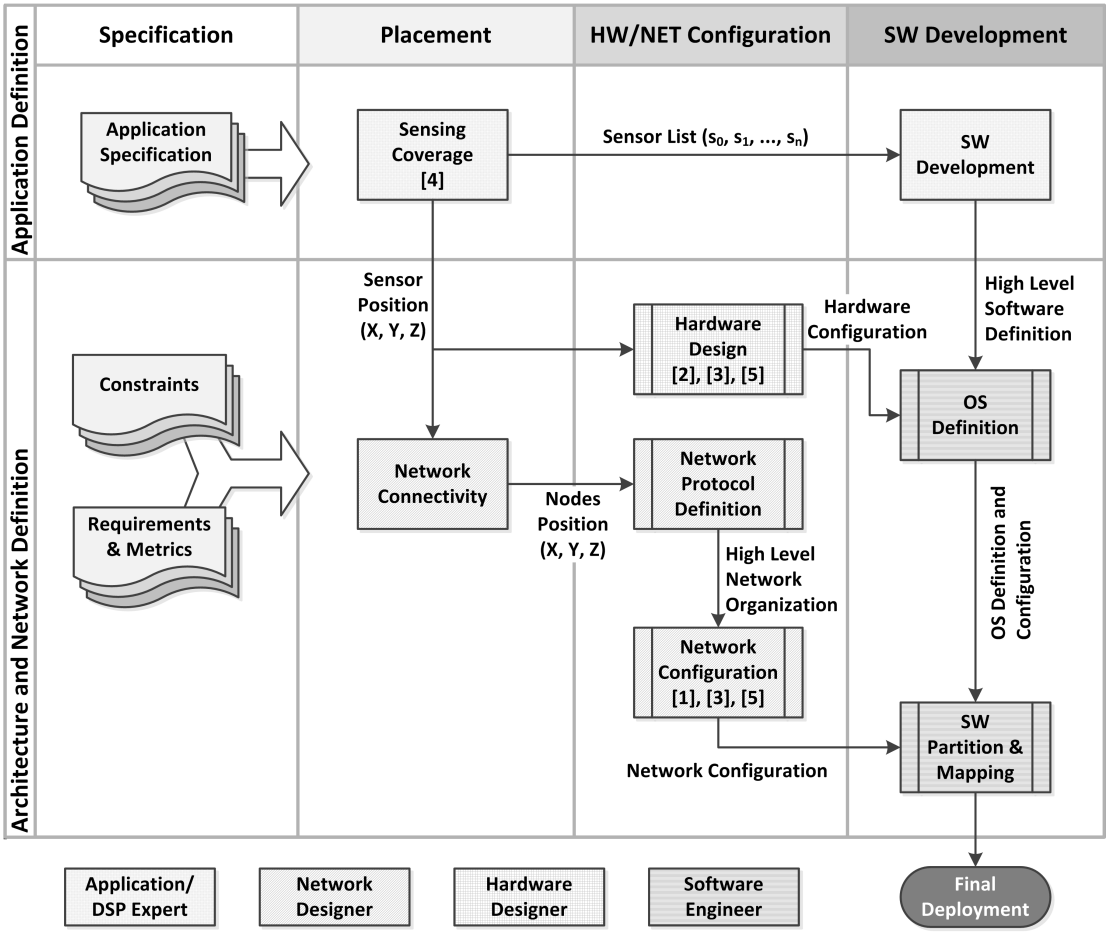
IMPLEMENTATION DETAILS

The design framework has been implemented in Python and C++. Python has been chosen according to the KISS (Keep It Simple, Stupid) principle. It is used to implement the high-level aspects of the framework such as: management of files, projects, plugins and design phases. C++ is used to perform computational-demand tasks such as optimization algorithms, simulations, models, etc. Thanks to the high flexibility of Python, C++ components can be easily integrated into the framework when required.

To allow the framework to be easily extendible, all the optimization's algorithms and tools must be compiled separately in order to make the development process easily manageable. Moreover, in this way the optimization algorithms are free to be implemented with whatever language, style and using all the required libraries. The only constraint is to keep input and output files adherent to the framework's definitions.

The framework provides a set of libraries to perform common tasks such as compare solutions or extract the Pareto frontier, and gives the wrappers to available evaluations' tools like simulators or models.

The framework itself presents a set of *black-box* components invoked in specific orders, according to design and user's needs. During the *Network Configuration* process,



Papers Published on Specific Aspects of the Design Flow

[1] P. R. Grassi, I. Beretta, V. Rana, D. Atienza, D. Sciuto, *Knowledge-Based Design Space Exploration of Wireless Sensor Networks*. In Proc. of International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2012), 7 - 12 October 2012, Tampere, Finland

[2] P. R. Grassi, D. Sciuto: Energy-Aware FPGA-Based Architecture for Wireless Sensor Networks. In Proc. of 15th Euromicro Conference on Digital System Design (DSD 2012), 5 - 8 September 2012, Izmir, Turkey

[3] I. Beretta, F. Rincon, N. Khaled, **P. R. Grassi**, V. Rana, D. Atienza, *Design Exploration of Energy-Performance Trade-Offs for Wireless Sensor Networks.* In Proc. of 49th Asia and South Pacific Design Automation Conference (ASP-DAC), 3 – 7 June 2012, San Francisco, USA

[4] P. R. Grassi, A. Ceppi, F. Cancarè, G. Ravazzani, M. Mancini, D. Sciuto, *Automatic Identification and Placement of Measurement Stations for Hydrological Discharge Simulations at Basins Scale.* in Proc. of European Geosciences Union General Assembly (EGU 2012), 22 - 27 April 2012, Vien, Austria

[5] I. Beretta, F. Rincon, N. Khaled, **P. R. Grassi**, V. Rana, D. Atienza, D. Sciuto, *Model-Based Design for Wireless Body Sensor Network Nodes*. in Proc. of 13th Latin American Test Workshop (LATW 2012), 10-13 April 2012, Quito, Ecuador

Figure 4.10: *Roles and Responsibilities of the Actors in the Design Flow*

for instance, the framework searches for the available algorithms and gives the user the choice. Once the user decides which algorithm should be used, the framework invokes and controls the optimization by providing a graphical output to the user. Once the task has been completed, the list of identified solutions is added to the project. Now the designer is free to optimize the network manually or perform another optimization.

For the framework, the design consists of several optimizations performed on the network. In case design is still not complete to perform an evaluation (i.e. MAC layer not been defined), general purpose solutions are used instead. For instance, if the MAC layer has not been defined yet, an ideal MAC layer is used instead. The purpose of this substitution is only to have all the necessary components to perform a simulation; by keeping these components fixed during the optimization process, the optimization process gives reasonable results. Note that this technique is used only during the first execution of the design flow since once all the processes have been performed, all the components of the design are identified. Further iterations have the objective to further optimize the network.

The information about the design project, analyzed solutions, simulations' output, etc. are saved in XML files in order to make the framework more interoperable with other tools. Moreover, the use of external files rather than internal structures, allows to manage plugins in an extremely flexible way. For the framework, a plugin is whatever piece of code that belongs to a specific design flow process and it is able to read input data and provide correct output data. However, although a plugin is usually an optimization algorithm - belonging to a specific (or a set of) design flow's process(es) - it can even be something different.

It specifies:

- The size of the environment in meters and the amount of nodes to be placed;
- The characteristics of the *wireless channel* (useful for the simulation);
- The available routing and mac layers with their parameters.

The framework reads the file and allows the user to select the preferred configuration or, in alternative, it explores the design space searching for an optimal (and feasible) configuration. As mentioned in Chapter 4, the size of the design space is extremely huge, thus the user or the algorithm should identify non-optimal solutions before evaluation (that could take several minutes or even hours).

4.2 The Design Space

As technology and research advances, the amount of design alternatives increases. It increases the complexity of design since a higher amount of feasible solutions must be evaluated, and the identification of the Pareto set is more difficult. In the design of WSNs, a good tradeoff among conflicting metrics required a deep exploration of the design space. In such scenario, the designer must be supported during the design phase, thus Computer Aided Design (CAD) tools are required. Automated design space exploration algorithm effectively help the designer during the design phase [34] [32]. Integration of many optimization/exploration algorithms in future design frameworks for WSNs is crucial to ensure a wide diffusion of WSNs.

This Section presents a formalization of the design space of WSNs. The proposed formalization helps the automated algorithms to explore the design space and to define the solutions such as different optimization algorithms can be integrated to cooperate in the identification of the optimal solutions' set. In order to validate and show the effectiveness of the formalization, two different case studies are presented. In this Section we would like to propose a formalization of the design space of WSN. It has been defined by analyzing how the design space of WSN is made, in particular, how P, M and K are characterized in a WSN.

As mentioned in Section 2.4.2, the Design Space Exploration of a WSN relies in the identification of configurations (or solutions) $P \subseteq \mathbb{P}$ such that P contains Pareto-optimal solutions. This Section explains how the $\mathbb{P}$ space is made in a WSN.

The parameter space ($\mathbb{P}$) of WSNs consists of three subspaces:

- **Placement Space** (Λ): a three dimensional space that defines where nodes can be placed. Each node must be placed in a specific position in $\mathbb{R}^3$;
- **HW/NET Configuration Space** (C): all the possible configurations of the sensor network and nodes (such as network interfaces and protocols, operating system preferences and policies, etc.);
- **SW Application Space** (A): the set of all implementable applications on the given network;

According to this parameters' space, in a network Δ, a node $\delta \in \Delta$ is located in $(x, y, z) \in \Lambda$, characterized by a specific HW/NET configuration $c \in C$ and programmed with an application code $a \in A$. As a result, each node of the network is defined by the tuple:

$$\delta \equiv \langle \lambda, c, a \rangle$$

And a solution is defined as a vector of tuples as follows:

$$(\langle \lambda_0, c_0, a_0 \rangle, \langle \lambda_1, c_1, a_1 \rangle, ..., \langle \lambda_n, c_n, a_n \rangle)$$

where n corresponds to the cardinality of the network. Summarizing, the design of a WSN consists in the identification of a set of nodes Δ such as the tuple $\langle \lambda, c, a \rangle$ is defined for each node, and the resulting metrics respect given requirements. Informally speaking, placement defines *"where nodes are"*, HW/NET configuration defines *"who nodes are"* (the role in the network) and application tells the network *"what they do"*.

4.2.1 Placement Space

In a working WSN, each node must be located in a specific position $(x, y, z) \in \Lambda$. The placement problem is the identification of the exact positions of nodes such that coverage and connectivity requirements are guaranteed.

The solution of the placement phase consists in a set of nodes in a three-dimensional space. Please note that the topology of the network (that tells how nodes are connected) is not part of the placement's solution since it could depend on the MAC and routing layers that impose specific topologies (for instance a two-tier network). For that reason, the connections among nodes (the actual topology), do not belong to the placement

solution and the output of the placement algorithms is only a set of nodes placed in Λ, regardless the (internal) model used to place the nodes; it improves the integration of the algorithms in the design frameworks. An example of a placement solution is provided in Figure 4.11.

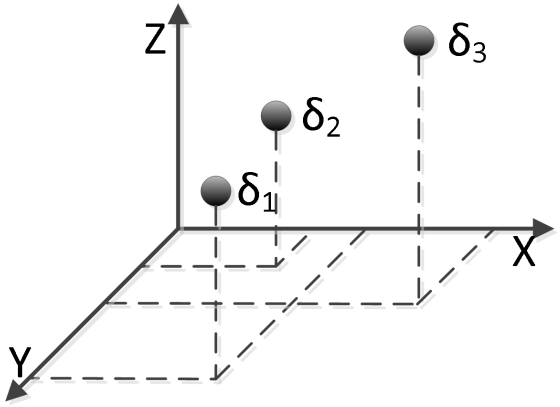


Figure 4.11: *Each node of the network must be assigned to a specific position in* Λ

4.2.2 HW/NET Configuration Space

Each node is composed of various hardware (processing unit, radio, etc.) and software components (MAC, routing, transport, etc.) that can be characterized with a certain amount of customization parameters. The value of certain parameters is not independent among the nodes; network interfaces, for instance, must be the equal, thus the choice of the network interface must be done network-level, rather than node-level.

For such a reason, configuration space C is further divided in three partitions:

- **Network Parameters** (C_n): the values assigned to those parameters must be the same for the entire network. It typically involves network interfaces or protocols;
- **Group Parameters** (C_g): the values of those parameters must be the same for the nodes belonging to the same group, but can have different values between nodes of distinct groups. This set includes, i.e. network coordinators, cluster-heads, etc. that require same, or similar, network configurations to operate;
- **Node Parameters** (C_d): the values of those parameters are independent from node to node.

The classification of the parameters in these spaces is left to the designer but, generally, network and group parameters usually refer to communication aspects such as radio, MAC or routing protocols, while node parameters refer to specific aspects of a node such as the amount of memory or the transmission power.

Another important aspect in defining the configuration space are the dependencies. $c_2 \in C$ depends on $c_1 \in C$ if $\exists c_1 \Rightarrow \exists c_2$. For example, the parameter *Superframe Order* (SO) must be defined if and only if IEEE 802.15.4 protocol is chosen as MAC, otherwise, the value and the existence of this parameter is completely irrelevant.

To describe the configuration space completely and correctly, a tree-based structure called *Configuration Tree* (CT) is introduced. In a CT, nodes can be of two kinds:

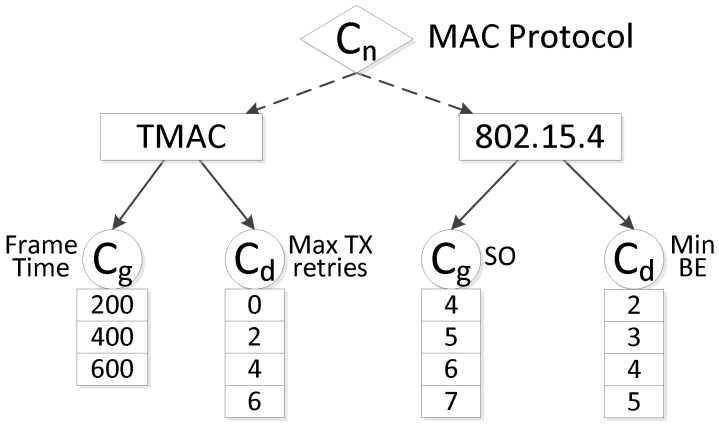


Figure 4.12: *An example of Configuration Tree (CT). Diamond node represents a choice node while circle nodes are normal nodes. Letters inside nodes indicate if the parameter is a Network parameter (C_n), group parameter (C_g) or a node parameter (C_d)*

choice nodes and **normal** nodes. A *choice* node is connected to various nodes that defines different design alternatives and it is used to define different, mutually exclusive scenarios (such as the MAC protocol to be used). The designer (or the algorithm) must select one and only one design alternative, otherwise the solution is unfeasible. A *normal* node defines a specific configuration parameter, whose valid values are defined in a list.

An example of CT is given in Figure 4.12 where two MAC protocols (TMAC and IEEE 802.15.4) and two parameters for each protocol are defined. Letters inside nodes indicate if the parameter is a network parameter (C_n), group parameter (C_g) or a node parameter (C_d). Parameters' dependencies are defined by lines: solid lines indicates a normal node and choice options with dashed lines. A feasible solution is {*MAC Protocol* = 802.15.4, SO_{G0} = 5, SO_{G1} = 4, *Min* BE_{N0} = 3, *Min* BE_{N1} = 3, *Min* BE_{N2} = 4, *Min* BE_{N3} = 5}, where $N0$ and $N1$ belongs to $G0$, and $N2$ and $N3$ belongs to $G1$.

CT has several advantages: it is a compact way to define the configurations parameters, it takes into account design alternatives, it defines existence dependencies between parameters. Tools for the automated design space exploration can take advantage from this structure to perform efficient (and correct) exploration of the design space.

4.2.3 SW Application Space

A common and generic way to define software applications is using **Program Dependence Graphs** (PDG) [57]. A PDG is a graph where nodes are tasks or conditions and arcs are data or control dependencies. Although, traditionally, nodes in the PDG represent a single instruction, they can be even used to define more complex tasks (data and control dependencies must be defined accordingly). To take advantage from PDG description, it is important to define technology-independent tasks or instructions (such as *"read from sensor X"* or *"send data to node Y"*). In that way, application development is not constrained to specific technologies (such as hardware devices or operating systems), and the design space exploration can analyze all the possible configurations.

Once the application and configurations have been defined, from the PDG it is always possible to generate platform-specific executable code. An example of PDG is

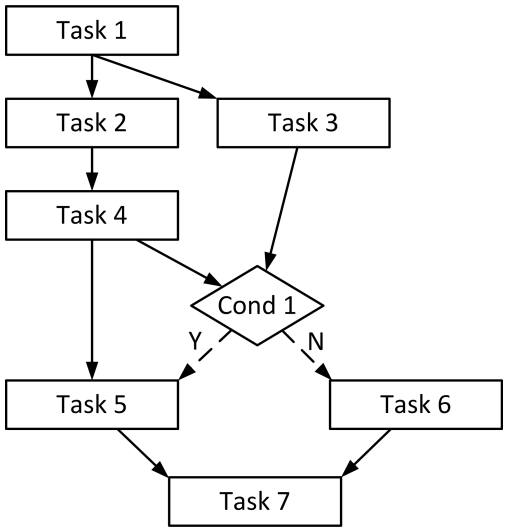


Figure 4.13: *An example of Program Dependence Graph (PDG). Square boxes represent tasks, diamond boxes Boolean conditions, solid lines data dependencies and dashed lines control dependencies*

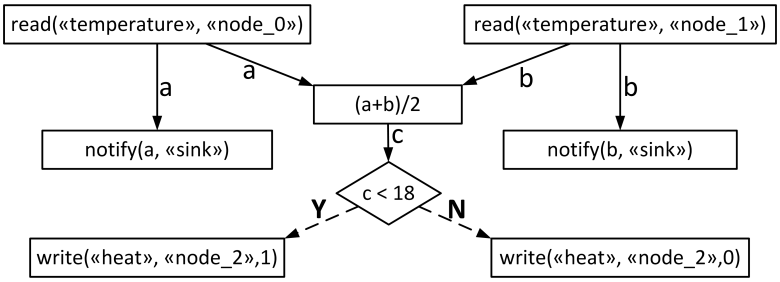


Figure 4.14: *PDG for Algorithm*

provided in Figure 4.13. In this example, *task 5* can be executed if and only if *task 4* has been previously executed and *cond 1* is true. Please note that dependencies are only *true-dependencies*, so only RAW dependencies [70] are considered here; false dependencies are solved by renaming variables, obtaining a Single State Assignment (SSA) representation.

To give a better understanding of the PDG, a practical example of a PDG representation is provided in Figure 4.14; this is the PDG representation of the Algorithm 1 in Section 4.1. From the translation in PDG it is possible to notice that although the NOTIFY functions have been written after the conditional block, they have no relationship with it, thus they can be executed *in parallel*. For the purpose of automated design of WSN, PDG is a very powerful representation since it allows the algorithm to estimate the amount of messages that must be transmitted between nodes to execute the desired application; for more information, please read Section 4.5.

4.2.4 Constraints

As aforementioned, design constraints are a set of (logical) conditions applied on P that reduce the number of possible solutions in the design. The resulting space $\mathbb{P} \in \hat{\mathbb{P}}$ is then used for the exploration ($\hat{\mathbb{P}}$ represents the unconstrained space). Various types

of constraints can be defined on each one of the three design subspaces:

- *Placement constraints* can be used to restrict the place where nodes can be placed. They can be defined with Boolean conditions on (x, y, z);
- *Configuration constraints* are defined on the CT in order to specify particular relationships between parameters;
- *Applications constraints* can be used to constrain the application's implementation (i.e. sensing task must be performed on the node that holds the sensor).

Differently from parameters' dependencies, constraints cannot be included directly into the CT and must be checked on each solution to verify if it satisfies the constraints or not. In order to leave the user free to define the constraints using whichever format, optimization algorithms must be designed in order to keep these constraints into consideration.

In addition, constraints can help the user to guide the design space exploration to good solutions quickly. In fact, they can be used to avoid not only unfeasible solutions, but also to avoid non-optimal solutions identified by previous experiments. Section 4.4.3.1 presents a very effective solution on how to use knowledge to speedup the design space exploration phase.

4.3 Placement

The placement problem consists in the identification of the optimal position of nodes such that sensing information are gathered correctly (**sensing coverage**) and the network satisfies connectivity and reliability requirements (**network connectivity**). As previously stated, many solutions were proposed to find a solution to this process, thus no additional algorithm has been developed. This Section will only provide an overview of the techniques proposed in literature.

Deployment Analysis in Underwater Acoustic Wireless Sensor Network (Pompili et al.) [137] This paper deals with the sensor coverage in underwater acoustic WSNs. The objective of this work is to determine the minimum number of sensors to achieve optimal sensing and communication coverage. Although the target applications are underwater acoustic sensor networks, their analysis can be extended to other kind of sensor networks since they present several mathematical investigations on sensing coverage. Figure 4.15 illustrates a triangular-grid deployment: subfigure (a) illustrates the overall grid deployment obtained using a triangular-based distribution and subfigure (b) shows the coverage issues (uncovered area) of this kind of deployment. For more details on the mathematical formulations and analysis, please read [137].

Relay Node Placement in Large Scale Wireless Sensor Networks (Tang et al.) [150] In this paper, the authors provide two polynomial time approximation algorithms to place relay nodes in a two-tier network to solve the *Connected Relay Node Single Cover (CRNSC)* and the *2-Connected Relay Node Double Cover (2CRNDC)* problems. A relay node is an intermediate node, without sensors, used to route data from sensors to sink. The relay node placement belongs to the *network connectivity* process, and aims at identifying the position of relay nodes given the position of sensing nodes.

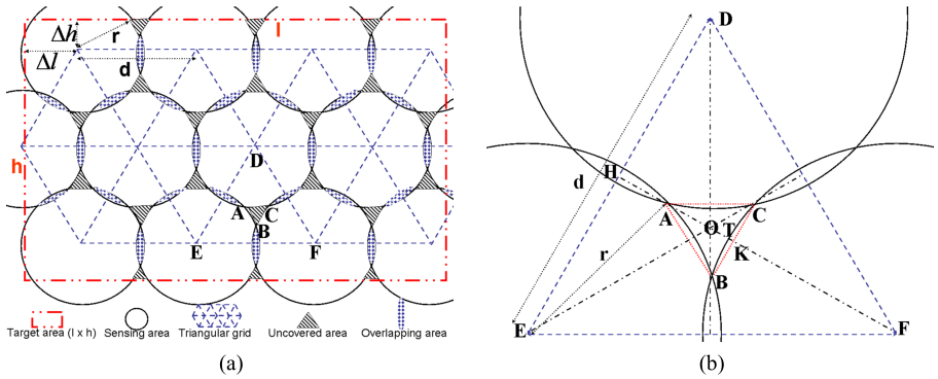


Figure 4.15: *Triangular-grid deployment. (a): Grid structure and side margins; (b): Uncovered area (from [137])*

The objective of the CRNSC problem is to find the minimum number of relay nodes and their corresponding location so that each sensor node is covered by at least one relay node, and that the network composed by the relay nodes is connected. On the other side, the objective of the 2CRNDC problem is to find a minimum number of relay nodes and their corresponding locations so that each sensor node is covered by at least two relay nodes, and that the network of relay nodes is 2-connected.

In both the proposed approaches, the algorithm divides the space into cells of fixed size and, as first step, places into each cell the minimum amount of relay nodes such that all the sensors are connected to at least one relay node. Then, for all the relay nodes H_i in each cell B_i, the algorithm identifies the set of relay nodes required to connect the cell on its right and bottom placing the nodes on the grid. The second algorithm is conceptually similar to the first, except that the first step ensures that all the sensors are connected to at least two relay nodes, and the second terminates when the network of relay nodes is 2-connected. For more details on these algorithms, please read [150].

Genetic Algorithm Based Node Placement Methodology for Wireless Sensor Networks (Bhondekar et al.) [34] In this paper, the authors present a multi-objective genetic algorithm to perform placement and high-level role assignment in WSN. The authors decode the 2D space into a numeric vector, where each item of the vector corresponds to a specific location into the (X, Y) field. The value of the vector item indicates whether the node is inactive (not placed into this field) or, in case of activity, which is its role. In the presented work, the authors consider three operative modes, but the proposed approach can be generalized to an arbitrarily amount of operative modes. The bitstream representation used by the genetic algorithm is depicted in Figure 4.16.

The fitness function used in that work is a linear composition of metrics such as *application-specific parameter*, *connectivity parameters* and *energy-related parameter*. However, the fitness function can be arbitrary defined by the user according to its needs.

Optimal Placement of Base Stations in a Two Tiered Wireless Sensor Network (Paul et al.) [133] Another known problem is the identification of the optimal position of base stations given a placed network. In this paper, the authors provide a geometrical solution using a partitioning technique. The network is clustered in a two

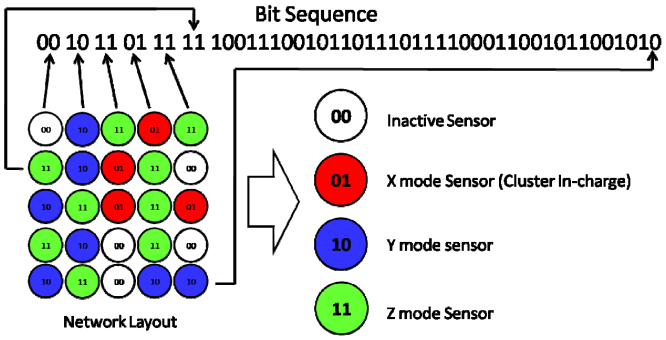


Figure 4.16: *Bitstream representation of the network layout (from [34])*

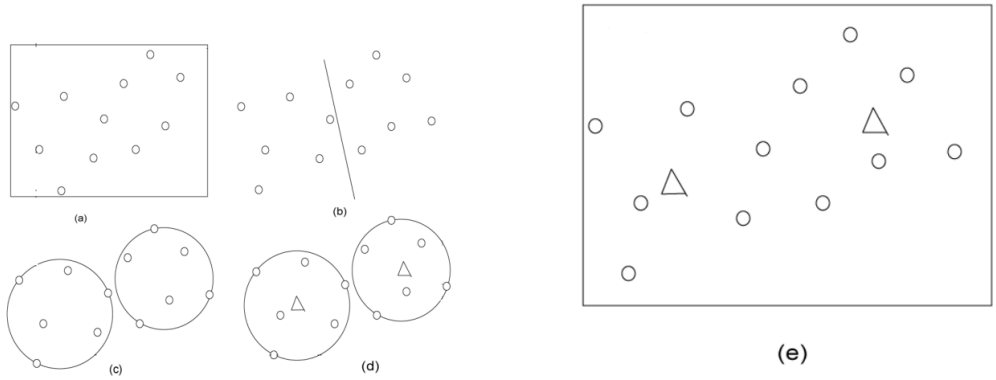


Figure 4.17: *Selection of BS locations using node partitioning technique (from [133])*

tiered WSN, where two types of nodes are deployed: *Sensor Nodes* and *Cluster Heads (CH)*. The proposed solution provides two major benefits: first, the algorithm does not suffer convergence issues and, second, the solution is identified deterministically, as opposed to the randomness caused by meta-heuristics such as Particle Swarm Optimization proposed in [71]. Figure 4.17 illustrates an example of how this technique works; for more details, please read [133].

This Section presented an overview of various works on node placement; the proposed analysis does not aim to be exhaustive since the placement problem is very complicated and is hardly generalizable, thus any application requires a specific placement algorithm. This Section illustrates some general approaches to show how the problem has been tackled in the past.

4.4 Hardware/Network Configuration

The second phase of the design concerns the HW/NET configuration. In this phase, the designer must define the hardware of the nodes (processor, memory, sensors, etc...), the network interfaces and protocols, and their configuration. At the end of this phase, the design is composed of nodes configured with *general-purpose* applications (used to test the network) and to estimate the quality of the configuration.

This Section presents three classes of techniques to perform HW/NET configuration: **simulation-based techniques**, **model-based techniques**, and **hybrid tech-**

niques. *Simulation-based techniques* are general purpose techniques that use simulation data to guide the Design Space Exploration. Meta-heuristics such as genetic algorithms, simulated annealing, tabu search, etc. are examples of simulation-based techniques. On the other hand, *model-based techniques* aim at identifing the optimal design by analyzing models of the node. In this Section, a specific model and the correspective analysis is shown. At the end of the section, a novel *hybrid technique* is presented. Considering that *simulation-based techniques* are accurate but extremely slow, *model-based techniques* are fast but not-so-accurate (at least as accurate as the model), an hybrid technique is required. The proposed algorithm simulates the design only if the accuracy of the model is not accurate enough; it reduces the amount of simulations required to identify Pareto-optimal solutions, reducing the overall amount of time required during this phase of the project.

4.4.1 Simulation-based Design Space Exploration

As mentioned in Chapter 2, evaluations can be performed with models, simulations or testbeds. Simulations offer a good trade-off between evaluation speed and accuracy. *Simulation-based Design Space Exploration* is a technique that uses simulators to evaluate the solution. Differently from model-based techniques, the quality of the identified solutions should be more accurate.

This Section illustrates a set of meta-heuristics [103] techniques to perform DSE in WSN. These techniques do not require any knowledge on the problem to optimize, thus they only need a clear specification of the design space. However, if this is an advantage, it is even its major limitation, in fact, the lack of knowledge requires the algorithms to perform a large amount of evaluations (simulations) to extract information on the problem to optimize. An analysis of how it could impact the effectiveness of the design space is given in Section 4.4.3.

META-HEURISTICS ON WIRELESS SENSOR NETWORKS

Meta-heuristics refer to many algorithms that use some degree of randomness to find the optimal (hopefully global) solution of a specific problem [103]. Some of them are included in the domain of **stochastic optimization**, while others, like **Markov Decision Processes (MDP)** do not belong to this class. A meta-heuristic is applied to those problems where no accurate models are available, when the optimality of the solution cannot be described analytically. Instead, the optimality of the solution is known only *after the evaluation*, so when its quality has been made explicit.

Algorithm 2: General Behavior of Meta-Heuristics

```
1 initialize(S)
2 repeat
3 |   {s_0} = selection(S)
4 |   evaluation({s_0})
5 |   S = S ∪ {s_0}
6 until final conditions satisfied;
```

The general behavior of a meta-heuristics is depicted in Algorithm 2 and it is composed of five parts:

- **Initialization** (line 1): data structures as well as configuration databases are initialized;
- **Main Loop** (lines 2 to 6): the algorithm iterates until certain conditions are satisfied. These conditions can be the maximum number of evaluated solutions or convergence;
- **Selection** (line 3): this function generates a new solution starting from the set of the evaluated solutions S;
- **Evaluation** (line 4): the given solution $\{s_o\}$ is evaluated and metrics are extracted;
- **Update** (line 5): the database of solutions is updated with the latest evaluated solution $\{s_0\}$.

Meta-heuristics mainly differ in the **selection**, where research studies are mainly focused. Genetic Algorithms, for instance, use **crossover** and **mutation**, while Simulated Annealing uses **local move**. *Crossover* requires (at least) two solutions, called *parents*, and generates a child solution mixing the characteristics (parameters) of the parents. *Mutation* is a change of a random parameter with a random value within the set of acceptable values. *Local Move* is a *small* change of the solution; it is usually implemented changing a single parameter with another value whose distance is limited (i.e. ± 1).

A comparison between meta-heuristics is presented in Chapter 6, where IEEE 802.15.4 and IEEE 802.15.6 have been optimized using meta-heuristics.

4.4.2 Model-based Design Space Exploration

Model-based Design Space Exploration is very effective and quick if accurate models are available. Differently from *simulation-based techniques*, presented in the previous Section, *model-based techniques* are problem-specific, since problem-specific models are required. In order to show the reader how to perform a *model-based DSE*, two approaches are presented in the next Sections.

4.4.2.1 Model-Based Design for Wireless Body Sensor Network Nodes

Wearable wireless body sensor networks (WBSNs) for health monitoring and diagnosis, as well as emergency detection, are gaining popularity and will deeply change healthcare delivery in the next years [131]. A WBSN node is a low-power device that collects vital signs, preprocesses and then sends them to a coordinator that performs most of the workload [153].

The design of WBSN nodes is mainly focused on maximizing the lifetime of the node by reducing the energy consumption, although other performance requirements such as the delay and quality of the delivered data must be kept into account. As a design may depend on tens of parameters, an efficient multi-objective optimization framework is required to explore the design space and to identify the Pareto-optimal solutions. The most critical part of this process is to provide a fast yet reliable estimation of all the optimization metrics. Three possible techniques to evaluate a solution are: a) an exhaustive set of experiments, which however cannot be automated; b) a network

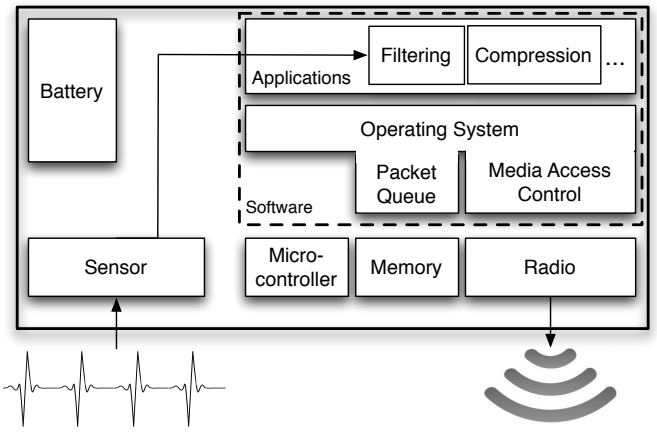


Figure 4.18: *Block diagram of the reference node architecture*

simulation, which is slow and hence impractical when a large number of potential solutions needs to be explored; c) an analytical model of the node, which favors a quick optimization and a better analysis of the node behavior.

In this work, the model-based optimization problem is analyzed from a different perspective, i.e., by narrowing the scope of the model to WBSNs and focusing on their most typical features (defined in [153]). In particular, this allows the designers to discard those aspects that are not generally required, such as complex networking or task assignment. In this way, they are able to analytically capture aspects like lifetime, transmission quality and application performance with a high precision, while still keeping the model general and reusable. Then, an adapted version of the multi-objective *simulated annealing* algorithm to perform the design space exploration and find the Pareto-optimal configurations is proposed. This has been tested on a real application for ECG monitoring that uses compressed sensing [105], and is implemented on the SHIMMER™ [42] state-of-the-art commercial platform.

Architecture of the Target Node

As a reference, a popular WBSN architecture is considered: a software application for data processing is executed on a microcontroller, and the remaining services are delegated to an operating system. Figure 4.18 provides a structural outlook of this class of nodes. Assume that a node transmits its data to a central coordinator through the typical star topology used in WBSNs. Each node in the network generates a constant traffic, thus avoiding data bursts that may interfere with the periodic transmission of the other nodes in the network. The result of this assumption is that the data delivery delay can be analytically estimated, and that the characterization of the network is simplified and can be seamlessly included in the node model.

- The *sensor* component describes the hardware that conditions and samples the raw signal at a frequency that depends on the signal and on constraints such as the Nyquist–Shannon theorem. Then, the samples are quantized by an A/D converter using a number of bits that depends on its resolution;
- The *applications* components comprise all the software programs used to process

the sensed data, including filtering, feature extraction, compression and aggregation. For illustration and without loss of generality, let consider the *compressed sensing* application [105], which reduces the amount of data by exploiting the sparsity of many body signals. In particular, the sampled signal is compressed with a certain ratio, which is set equal to one if no compression is adopted;

- The *operating system* (OS) provides services such as the interaction with the hardware, the software-level management of the sensing, the memory and the radio transmission. The OS also manages a set of queues, including the ones for interprocess communication, and the one containing the packets to be transmitted. Furthermore, the OS implements a MAC protocol to share the access to the wireless medium among the nodes in the star-topology network;
- The *microcontroller* (μC) is the component of the platform in charge of executing the OS and the software applications. Depending on the hardware, techniques such as dynamic voltage scaling [64] might be available to allow the microcontroller to be active for a limited time (*duty cycle*), and to switch to a low-power state when there is no task to be executed;
- The *memory* bank is used to store the data required by the applications and the OS. Although a larger memory exhibits a higher energy consumption [81], a limited size may affect the capacity of the internal queues, and hence the performance of the applications and the throughput of the transmission;
- The *radio* component describes the hardware used to modulate and transmit the data through the wireless channel. Depending on the characteristics of the platform, the wireless transmission power and the modulation scheme can be adjusted, even dynamically [142], to determine the distance that is covered with a predetermined packet error rate (PER).

RELATED WORK

Model-based optimization of wireless sensor nodes is a topic that has been already explored in the literature. However, most of the related works characterize the energy consumption of one of the node components shown in Figure 4.18, whereas just few of them aim at optimizing multiple components and performance metrics, which is instead the purpose of our work. However, the models of the single components cannot capture the interdependencies that exist between the different parts of the node, and in particular they often discard the effects of the application. In [146], the authors propose an alternative MAC protocol for the *ZigBee* standard that introduces new power-saving policies. In [99], a model that relates the routing performed at the MAC level to the node lifetime is proposed. However, both works assume a multi-hop routing, and thus they cannot be applied to the star topologies used in WBSNs [153].

A few existing works have tried to propose an optimization that considers several components of the node. In [36] the authors propose a platform-based design methodology for industrial control. Although the work considers all the aspects of the node design, it is not based on an analytical model, and it mainly focuses on the generation of a complex network, which is not a critical aspect in WBSNs. In [88], different energy/delay tradeoffs are explored by exploiting voltage and modulation scaling.

Similarly, [163] proposes a model-based optimization framework for star-topology networks, and a genetic algorithm to reduce the energy consumption by acting on the voltage and the modulation level. However, the number of parameters involved in the optimization is a small subset of the ones that can be tuned on real nodes. In [118], the authors propose an optimized transmission schedule to minimize the packet delay. The work shows good potential if transmission delay is the main objective, but it is not proved to scale in more sophisticated multi-objective optimization scenarios. A very relevant example of application-driven design is proposed in [68], where a multi-objective optimization involving all the system components is provided in the field of wildlife monitoring. The work shows how a deep knowledge of the final application can lead to an optimized node lifetime, while guaranteeing the quality of service (high data rate and low distortion). However, it heavily relies on the experimental data and hence is very specific for the target domain.

THE PROPOSED MODEL-BASED OPTIMIZATION FRAMEWORK

The proposed model-based optimization framework is targeted to WBSN nodes whose architecture has been depicted in Figure 4.18. The framework includes two parts: a node model that estimates the relevant quality metrics associated to the system, and an algorithm to find the optimal configurations.

Analytical Node Model The node model includes fundamental parts that describe the common structures of every WBSN node, and advanced parts that can be further detailed according to the specific scenario (e.g., the application, the communication channel), and parameters that need to be determined through experimental data. The contribution to the energy consumption of each component is characterized, although the level of detail differs from component to component depending on the number of relevant design parameters.

The proposed model is comprehensive as it captures the interdependencies between different components. The main metric, i.e., the overall energy consumption, is expressed as a function of the all hardware components of the system (sensor, microcontroller, memory and radio), whose behavior is influenced by the applications and the OS. In particular, the energy consumption per second is expressed as:

$$E_{node} = E_{sensor} + E_{\mu C} + E_{memory} + E_{radio} \ . \tag{4.1}$$

However, since the straightforward reduction of the energy consumption may lead to the loss of performance in one or more components of the node, we defined a set of performance metrics that only involve one or two components, in order to keep them monitored during the design.

Sensor The sensor component consists of a transducer to detect the signal, and a hardware circuit to sample the data at a frequency $f_{sampling}$ and quantize it with a resolution of ρ_{ADC} bytes. The energy consumption of the former can be considered as a constant related to the specific sensor, while the latter is linearly related to the sampling frequency [41]:

$$E_{sensor} = E_{transducer} + [\alpha_1 \cdot f_{sampling} + \alpha_0] \ , \tag{4.2}$$

where α_1 is a constant depending on the capacitance and the square of the supply voltage [41] of the A/D converter, while α_0 describes leakage effects and is determined experimentally.

Although specific metrics can be defined to estimate the performance of the sensor, e.g. the signal-to-quantization-noise ratio, a more meaningful evaluation can be obtained by combining it with the subsequent application component.

Applications The applications are software programs that do not directly dissipate energy, but influence the performance of the microcontroller, the memory and the radio. In particular, the applications define the duty cycle (ψ_{app}) of the microcontroller, the memory requirement (σ_{app}) and the average number of memory accesses per second (γ_{app}), which can be determined using software profiling. As previously assumed, data can be compressed at the application level with ratio CR, which generates the following amount of packets to be transmitted per second (R_p):

$$R_p = f_{sampling} \cdot \frac{CR \cdot \rho_{ADC}}{H_{payload}} \ , \tag{4.3}$$

where $H_{payload}$ is a parameter that defines the number of data bytes included in each communication packet, and hence $H_{payload}/\rho_{ADC}$ denotes the number of samples per packet.

In order to evaluate the performance of the applications, domain-specific metrics can be defined. If compression is the only processing performed at the application level, such a metric is defined as the quality of the reconstructed signal.

Operating System The OS is composed of software routines that implement services such as the packet queue and the MAC layer. The software routines of the OS can be modeled as any other software application, thus requiring a duty cycle ψ_{OS} from the microcontroller, and a maximum memory σ_{OS} that is accessed γ_{OS} times per second on average. For the sake of analysis, a separation between memory required by the transmission queue and the remaining memory occupied by the OS is required, since a detailed model of the transmission queue is crucial to characterize the throughput of the system.

The MAC layer implemented in the OS manages the access to the wireless channel shared among a known number of nodes (N_{nodes}) connected to the WBSN coordinator. The access policy defined by the MAC algorithm can be modeled using two quantities: a transmission window of length Δ_{tx} when the node can transmit without conflicts, and the number of times this window is repeated per second, N_{tx}. Those quantities can be directly computed for contention-free access mechanisms, but they can also be determined statistically for contention-based policies. In order to enforce the access policy, the algorithm may require a number of control messages to be exchanged from the node to the coordinator. This number, denoted as $\Phi_{Node \to C}$, and the length of those messages as $H_{Node \to C}$), and the opposite as $\Phi_{C \to Node}$, of length $H_{C \to Node}$. Finally, the MAC protocol defines the control information that must be included in each packet (typically a header and a checksum), thus determining the final length H_{packet} of each packet. The transmission queue, which can contain up to λ packets, has then a size of λH_{packet} bytes.

A set of performance metrics can be defined for the OS component, like the throughput and the packet delivery delay. In particular, the system should guarantee a throughput of R_p packets per second as required by the application, but certain packets might need to be retransmitted with a probability equal to PER. Let $R_p^{(r)}$ the packet rate including the retransmissions, which is equal to $R_p \cdot [1 + \varpi(PER)]$, where $\varpi(PER)$ is the estimated number of retransmissions, including possible multiple retransmissions of the same packet. Then, a sufficient condition to guarantee that the desired throughput is met is the following:

$$N_{tx} \left\lfloor \frac{\Delta_{tx}}{T_{packet}} \right\rfloor \geq R_p^{(r)} \ \wedge \ \lambda \cdot N_{tx} \geq R_p \ , \tag{4.4}$$

where T_{packet} is the packet transmission time, and $N_{tx} \cdot \lfloor \Delta_{tx}/T_{packet} \rfloor$ is the maximum channel capacity guaranteed by the MAC layer. The second condition prevents the capacity of the queue from being a bottleneck and to avoid dropped packets. If the conditions are satisfied, the worst-case estimation of the packet delivery delay is:

$$Delay = \frac{1}{N_{tx}} \cdot \frac{\sum_{i=N_{tx}-\nu}^{N_{tx}-1} i}{\nu} \ , \ \ \nu = \left\lceil \frac{R_p}{R_p^{(r)}} \cdot N_{tx} \right\rceil \ , \tag{4.5}$$

where ν indicates the fraction of the N_{tx} transmission windows that are required to send R_p packets. $Delay$ is then computed by considering that all the retransmissions occur before the R_p packets are transmitted.

Microcontroller Similarly to the sensor, the consumption of the microcontroller is expressed as a function of its frequency $f_{\mu C}$. The processor needs to be active for a duty cycle defined by the application and the OS, before switching to a low-power mode where only leakage effects occur:

$$E_{\mu C} = (\psi_{app} + \psi_{OS}) \cdot \beta_1 \cdot f_{\mu C} + \beta_0 \ , \tag{4.6}$$

where β_1 depends on the capacitance and on the square of the supply voltage, and β_0 describes the leakage effects and should be determined experimentally. Note that, if the specific scenario does not allow the microcontroller to switch to a low power state, $\psi_{app} + \psi_{OS}$ should be set equal to one.

Memory The system memory is used for the execution of the applications and the OS, and to store the packets queue. The memory size M, which is also the main quality metric for the design of the memory component, can be written as:

$$M = \sigma_{app} + \sigma_{OS} + \lambda H_{packet} \ . \tag{4.7}$$

The energy consumption of a memory component is due to two factors [81]: a dynamic consumption due to the memory accesses, and a leakage that is known to be proportional to the memory size and appears when the memory is not being accessed. The software applications and the OS access the memory γ_{app} and γ_{OS} times per second, respectively. The transmission queue is filled with a number R_p of packets per second and, since the defined throughput guarantees that no packet is dropped, it is eventually

read at the same rate. Assuming that the memory access in read and write modes has the same cost, the energy consumption can be expressed as:

$$E_{memory} = (2 \cdot R_p + \gamma_{app} + \gamma_{OS}) \cdot T_{mem} \cdot \zeta_{access} + [1 - (2 \cdot R_p + \gamma_{app} + \gamma_{OS}) \cdot T_{mem}] \cdot M \cdot \zeta_{idle} \ , \quad (4.8)$$

where T_{mem} is the access time in read or write mode, while ζ_{idle} and ζ_{access} are hardware parameters that define the consumption in idle and accessing modes.

Radio The energy consumed by the radio depends on the number of packets that are sent and received. In particular, when a transmission of one bit takes place, the energy consumption can be expressed as:

$$E_{tx} = [P_{carrier} + P_r] \cdot T_{bit} \ , \quad (4.9)$$

where $P_{carrier}$ is the power required to generate the signal carrier, P_r is the remaining consumption related to the radio circuit. T_{bit} indicates the average time to transmit one bit, which also includes all the control information added by the physical layer [142]. The value of $P_{carrier}$ can be determined according to the desired PER. In particular, given the level of noise at the receiver, it is possible to compute the signal-to-noise ratio and consequently the bit error rate (BER), as a function of $P_{carrier}$ and the modulation scheme [142]. Once the BER is known, the packet error rate can be expressed as the probability of one bit being erroneous in a packet of length H bytes, i.e., $1 - (1 - BER)^{8H}$.

The energy required to receive a bit (E_{rx}) is computed as in (4.9), where $P_{carrier}$ is equal to zero, and P_r has a different value during the receiving phase. As a consequence, the energy consumption of the radio can be expressed as:

$$E_{radio} = E_{tx} \cdot \left[R_p^{(r)} \cdot 8H_{packet} + \Phi_{Node \to C} \cdot 8H_{Node \to C}\right] + E_{rx} \cdot \left[\Phi_{C \to Node} \cdot 8H_{C \to Node}\right] \ . \quad (4.10)$$

Node Optimization The proposed model provides an accurate evaluation of the node, hence it is suitable for scalar or multi-objective exploration to find a set of Pareto-optimal configurations. However, even in an existing platform where all the hardware constants are already fixed, the optimization may still involve tens of design parameters, ranging from low-level hardware aspects to the tuning of software algorithms. In order to show how the design space can be efficiently explored using the estimation provided by the proposed node model, a heuristic algorithm, based on *simulated annealing* [80], is introduced. The choice of this technique is motivated by its ability to handle a large number of design parameters, its scalability from single-objective [80] to multi-objective optimizations [121], and its good convergence properties. The proposed Multi-Objective Simulated Annealing (MOSA) algorithm is shown in Algorithm 3, and it features a different non-dominance policy with respect to the standard MOSA to improve the stability of the algorithm and consequently the quality of the results.

In order to set up the optimization, one or more metrics (e.g., E_{node}, $Delay$, M, PER) should be picked as objective functions, thus generating a cost vector $Cost(S)$ for each node configuration S. Since the single-objective simulated annealing inherently finds a single solution, the MOSA requires multiple executions ($N_{executions}$) of

Algorithm 3: Multi-Objective Simulated Annealing

```
for i = 1 ... N_executions do
  S ← GenerateRandomConfiguration()
  T ← T_0
  repeat
    S' ← GenerateNeighbor(S)
    if Cost(S') dominates Cost(S) then
      S ← S'
    else if Cost(S) dominates Cost(S') OR
    no domination between Cost(S) and Cost(S') then
      if RandomNumber() < P(T) then
        S ← S'
      end if
    end if
    T ← Annealing(T)
  until T < T_min
end for
```

the annealing process [121] to populate the Pareto set: the larger $N_{executions}$, the more accurate the Pareto set is, although the MOSA outputs Pareto solutions from the early executions.

Each execution starts from an initial temperature T_0 (typical values are 300 or higher) and a random initial configuration S. A new candidate solution S' is then obtained by randomly modifying one of the parameters of S and, if the S' dominates S, then it is accepted. If the candidate is dominated by the current best, it can still be accepted with a probability P. Different criteria to compute P can be found in the literature [121]: after analyzing them, is possible to conclude that the best way to compute P only involves the temperature T as follows:

$$P(Cost(S), Cost(S'), T) = e^{-C/T} \ , \ C \in \mathbb{R} \ . \tag{4.11}$$

According to the classical formulation of the MOSA [121], a non-dominance condition between $Cost(S)$ and $Cost(S')$ leads to the acceptance of the candidate, in order to explore more solutions. This approach, however, may discard a good solution even at low temperatures. In this framework, a different technique has been proposed, which employes the transition probability even in non-dominance scenarios and allows the algorithm to converge as the temperature decreases.

The annealing scheme should guarantee that the temperature decreases slowly enough to allow the algorithm to converge. A popular scheme is the geometric one, where the temperature T is multiplied by a constant value lower than 1 (typical values are 0.99 or higher) at each iteration. The execution finishes when the temperature reaches a lower bound T_{min} (typically close to 5), that corresponds to a low transition probability.

A Case Study The considered real system samples the ECG and uses the compressed sensing [105] to reduce the amount of data to be transmitted. The compressed signal is then sent to a smartphone that acts as a central coordinator, reconstructs the ECG and performs analysis and detection tasks. The choice of compressed sensing is motivated by the improved node lifetime, indeed, experimental results [105] showed that com-

pressing and sending data can increase the node lifetime by 9.7% when compared to transmission of the raw ECG.

The system is implemented on the Shimmer™commercial platform [42], which features an ultra low-power microcontroller working at a maximum frequency of 8*MHz*, 10*kB* of RAM memory, and a radio implementing the IEEE 802.15.4 communication. A 3-lead ECG sensor is connected through a daughter board. On the software side, FreeRTOS [2] was ported on the node, to control the sensing, the queue services and the beacon-enabled mode of the IEEE 802.15.4 MAC layer [18]. In this MAC protocol, a beacon is periodically sent by the coordinator to define the time structure in terms of superframes. A superframe is a time interval divided into an inactive and an active part, the latter being further divided into a contention-free and a contention-active portions. In this case study, only the contention-free part is used, so the transmission only occurs during *guaranteed time slots* (GTSs).

Mapping the Case Study on the Analytical Model The proposed model provides a good characterization of many parts of the target node, but additional information can be included to further describe the application, the memory, the MAC and the radio modulation.

At the application level the duty cycle ψ_{app} required by the compressed sensing, in the current implementation, it only marginally depends on the value of CR. The performance metric considered for this component is the *percentage root-mean-square difference* (PRD), which quantifies the difference between the original ECG and the one reconstructed by the coordinator from the compressed data. By analyzing the experimental data provided in [105], the PRD can be expressed as a fifth-order polynomial function of CR:

$$PRD = \omega_5 CR^5 - \omega_4 CR^4 + \omega_3 CR^3 - \omega_2 CR^2 + \omega_1 CR - \omega_0 \ , \tag{4.12}$$

where the coefficients ω_n are positive constant values.

The total available memory on the node is $10kB$: according to the experimental results, $6.5kB$ are required by the compressed sensing application (σ_{app}), while $3.5kB$ are reserved for the FreeRTOS routines (σ_{OS}) and for the transmission queue, whose size is then upperbounded. In particular, for a packet length H_{packet} of $127B$ (i.e., the maximum value for the selected MAC), λ must be lower than 10.

The beacon-enabled IEEE 802.15.4 MAC layer can also be easily included in the node. Two protocol-specific parameters need to be defined: the *Superframe Order* (SFO), and the *Beacon Order* (BCO). The former determines the active period or *superframe duration* (SD), while the latter defines the interval between two beacons (BI) as follows:

$$SD = 15.36ms \cdot 2^{SFO} \ , \quad BI = 15.36ms \cdot 2^{BCO} \ . \tag{4.13}$$

The superframe structure can be mapped on the transmission window Δ_{tx} of our model, as the average transmission time per second is equal to SD divided by the number of nodes in the network. Similarly, BI defines how many times a superframe is repeated, hence it can be related to N_{tx}:

$$\Delta_{tx} = \frac{SD}{N_{nodes}} \ , \quad N_{tx} = \frac{1}{BI} \ . \tag{4.14}$$

In terms of control messages, the standard does not require any control message from node (thus $\Phi_{Node \to C} = 0$), whereas the coordinator sends a number of beacons that depends on BI, and an acknowledgment for each transmitted packet, hence:

$$\Phi_{C \to Node} = R_p^{(r)} + \frac{1}{BI} \, . \tag{4.15}$$

Finally, the estimation of the PER for this case study can be obtained by computing the BER for the 4-PSK modulation [142] with the selected value of $P_{carrier}$.

Node Optimization The proposed MOSA has been used to determine a set of Pareto-optimal node configurations for the case study application. The design parameters available on the target platform are $f_{\mu C}$, CR, $H_{payload}$, λ, BCO, SFO, and P_{tx}, while the cost function includes E_{node}, PRD, $Delay$, and PER. For the sake of illustration, it has been adopted a coarse discretization of the parameters to reduce the design space to 10^8 solutions, which can be explored by an exhaustive algorithm to provide a comparison between the MOSA and the real Pareto set.

The estimation provided by the proposed model proves to be effective as the error with respect to the experimental data is very low (i.e., it does not exceed the 1.9%). Moreover, results show that the proposed MOSA effectively explores the Pareto set, as the optimal solutions found by the MOSA perfectly match the ones found by the exhaustive algorithm, and cardinality of the Pareto set scales well with the number of executions. The solutions show a wide range of tradeoffs, e.g., the difference between the extreme values of E_{tx} exceeds 44%, the values of the PRD span from 0 to 93 (the maximum range is up to 100), and it is possible to achieve real-time transmission as well as packet latencies of tens of seconds.

Let us compare the proposed exploration technique with the other state-of-the-art approaches. Those works aiming at energy minimization, such as the one in [163], only produce a single solution that minimizes E_{node}, which is also found by the proposed MOSA. A more interesting comparison involves the proposed approach and a multi-objective optimization of energy and delay, as proposed in [88]. Figure 4.19 shows that both approaches can identify the Pareto curve in terms of energy and delay. However, [88] shows some limitations when the application is considered, as it generates solutions with a high PRD (see Figure 4.20) and a high PER, and only finds 2.3% of the solutions found by the proposed approach. Finally, the executions of the MOSA and the approach in [88] takes a few minutes, whereas the exhaustive search takes a few hours and does not scale well on larger solution spaces.

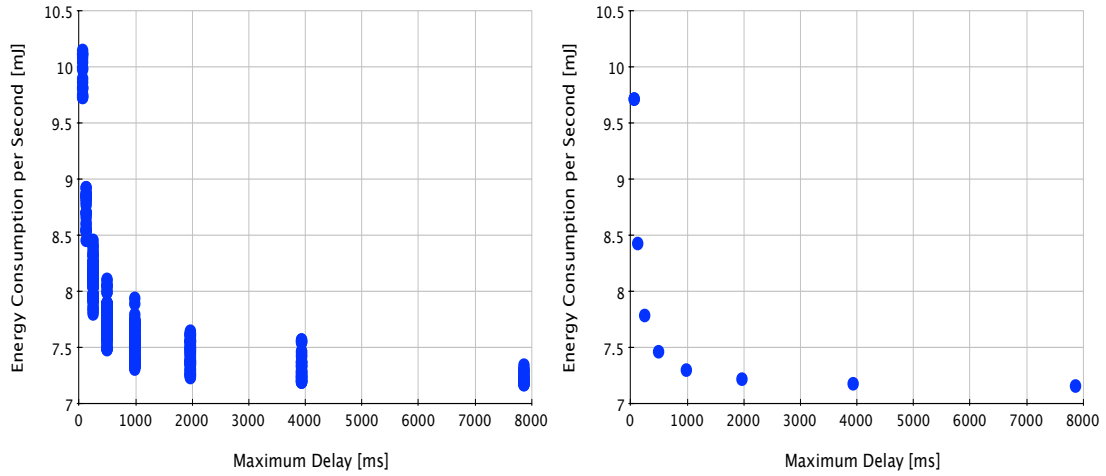


Figure 4.19: *Delay and energy consumption of the Pareto solutions found by the proposed approach (on the left), and the work in [88] (on the right)*

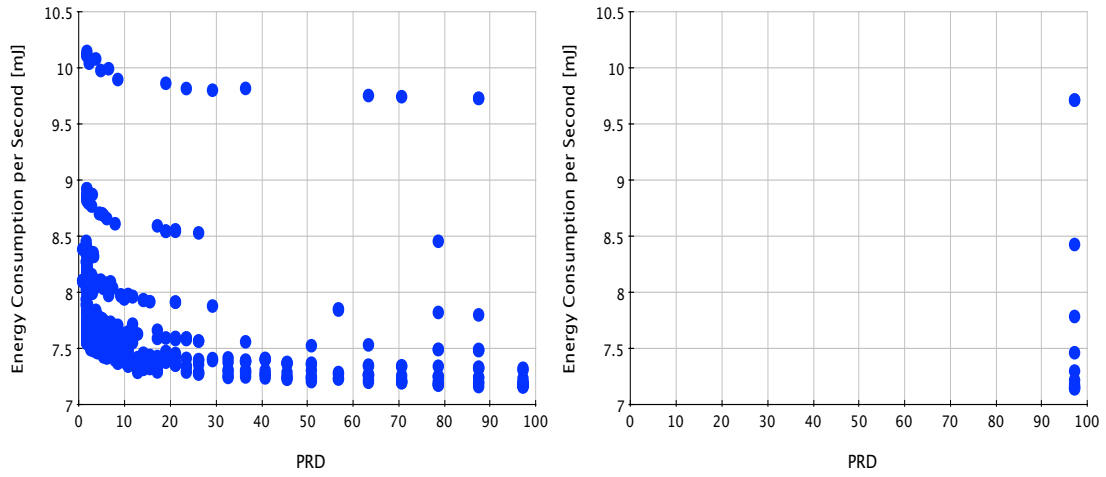


Figure 4.20: *PRD and energy consumption of the Pareto solutions found by the proposed approach (on the left), and the work in [88] (on the right)*

Summary This work presented a model-based optimization framework for the design of WBSN nodes. Interdependencies between the most recurring components of a WBSN node, and their effects on the energy consumption have been modeled and a multi-objective optimization algorithm to explore the optimal tradeoffs available in the design space has been proposed. The proposed model has been validated on a real case study, showing how it can effectively handle real standards, and how the optimization algorithm finds solutions that are consistent in terms of both energy and performance.

4.4.2.2 Design Exploration of Energy-Performance Trade-Offs for Wireless Sensor Networks

This second example is focused on the design of WBSN with trade-offs among energy and performance in mind.

The evaluation of a particular WSN includes aspects that span across multiple layers (from the network to the hardware, to the application level), and it can be performed in three ways [29]: a set of physical experiments, a network simulation or an analytical model. However, when a large number of configurations needs to be evaluated during the DSE phase, both the empirical experiments and the simulation approaches become impractical, as the former cannot be automated, while the latter takes an unacceptable amount of time. Conversely, the analytical model enables a fast evaluation and a deep understanding of the dynamics of the network, but its definition raises several challenges related to its accuracy and reusability. In fact, a detailed characterization of a specific WSN has been shown to lead to efficient network designs [68], but such a model requires a detailed knowledge of the application and the target platform, and it cannot be reused to model different classes of WSNs. On the other hand, a generic system-level model that can be easily instantiated to a specific WSN would greatly simplify the task of the designer, but no model with these characteristics has been proposed yet, as it is complex to define a characterization that can describe *all* the different classes of WSNs with a sufficient accuracy.

Although the definition of a general model is limited by the great differences among the WSN domains, differently from the previous example, it is shown that it is possible to focus the scope of the model to wide classes of networks in order to capture their most relevant aspects, thus providing a model that is both detailed and reusable on many instances of WSNs. A multi-layer characterization of the nodes and of their interactions in a well-defined class of WSNs leads to an accurate estimation with respect to both actual and simulated data, and that a DSE algorithm greatly benefits from a model-based evaluation in terms of execution time. Furthermore, in order to provide a coherent system-level estimation of the network during the DSE, a set of performance metrics that belong to different layers (i.e., delay, application quality, energy consumption) is proposed; it leads to the determination of the optimal energy-performance tradeoffs. As a proof of concept, the proposed model targets the wide class of wearable wireless body sensor networks (WBSNs), which are a rising technology in the field of human health monitoring [131] both for medical and personal use. Experimental evaluations conducted on a real-world WBSN show that the proposed model never generates an estimation error greater than 1.74%.

Related Work Over the last years, model-based evaluation as a support for DSE has been extensively explored in many fields. In the WSN domain, node and network models have been traditionally proposed to characterize specific aspects of the network, and to validate new protocols [146] or energy management strategies [163]. None of them, however, guarantees a general system-level description that can be easily adapted to describe a real WSN.

At the node level, analytical models for all the most common hardware blocks were proposed even before the advent of WSNs. In particular, detailed energy characterizations are available for hardware circuits, sensors and microcontrollers [64], memory banks [81], and radio circuits [142] [43]. However, these models do not consider any interdependency between the different parts of the system, hence they are not sufficient to describe the behavior of a set of networked nodes. In [163], the authors relate the energy consumption and the throughput of the node to the supply voltage of the micro-

controller and the modulation level of the radio. Although the work is a good example of how different aspects of the node (i.e., sensing, processing and transmission) can be combined, the parameters that are considered are only a small subset of the ones that can be found on real nodes.

At the network level, several works propose a model of different media access control (MAC) protocols, and in particular the widely-adopted IEEE 802.15.4 [18] standard. For example, [82] characterizes the behavior of the IEEE 802.15.4 MAC layer on large-scale networks, both in terms of energy consumption and packet transmission probability. In [151], a similar analysis is proposed for WBSNs, with a particular emphasis on the radio activity of the node. The works in [83] and [128] focus on the part of the IEEE 802.15.4 standard that works in TDMA mode, and propose two separate techniques to estimate the expected packet delay. However, none of the aforementioned network models propose an in-depth analysis of the application executed by the nodes, which is crucial to have a coherent global evaluation of the WSN.

Another important aspect of the trade-off analysis is the definition of a set of metrics that capture all the relevant dynamics of the network. Traditionally, energy consumption is always a major concern during the network evaluation [146] [163] [82], but other metrics such as throughput and end-to-end delay may be considered. For example, different energy/delay tradeoffs are explored as a function of the voltage and the radio modulation in [88]. However, no application-related metric is generally proposed in order to characterize the overall behavior of the network as seen by the end user.

As a conclusion, none of the existing models provides a coherent system-level description that can be applied to real-world WSNs, mainly because they only focus on specific aspects of the system and often neglect the final application. This example shows that a general –and yet reliable– analytical model for the nodes and the whole network can be defined if its scope is limited to a set of WSNs sharing similar structures and application domains.

System-Level Model for WBSNs A typical WBSN [153] follows the structure illustrated in Figure 4.21. The network comprises a set of low-power nodes that can be worn by the same person or by different ones (e.g., the patients in a hospital, or a team of athletes) to monitor one or more vital signs to be sent to a central network coordinator. Once the signal has been sensed, each node performs a data pre-processing using a software application executed on a microcontroller-based hardware architecture (see Figure 4.21), and finally sends the output to the coordinator through the wireless channel. The coordinator is responsible for the analysis of the data, and the definition of the network activity (e.g., the enforcement of the MAC protocol). In WBSNs, a star topology network is generally employed, hence the communication between a node and the coordinator is direct [153]. Moreover, the wireless channel is shared among the nodes using a collision-free, time-division multiple access (TDMA) policy, which leads to a lower energy consumption with respect to a contention access. These assumptions are sufficient to characterize a wide set of networks in the WBSN domain in order to define an abstract model that can be easily adapted to real nodes and standards, as shown later by means of a case study.

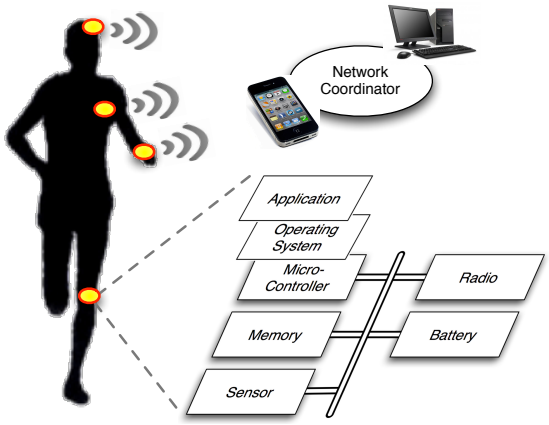


Figure 4.21: *Overview of a typical WBSN*

Network Model The network model captures the interactions among the nodes and, in particular, how they share the wireless channel. For this purpose, the N nodes of the network are now considered as black boxes generating an output stream of ϕ_{out} bytes per second (B/s). The transmission is regulated by the MAC protocol, which aims at assigning a transmission interval $\Delta_{tx}^{(n)}$ (the index denotes that the quantity refers to node n) per second to each node, by acting on protocol-specific parameters that form a *configuration* χ_{mac}. Each node is then in charge of tuning the throughput $\phi_{out}^{(n)}$ in order to be able to deliver its data in the time $\Delta_{tx}^{(n)}$.

To describe the MAC layer, the following abstractions that capture its most recurring characteristics are introduced:

- a *data overhead* due to packetization and flow control, consisting of a number of extra bytes that are required to transmit ϕ_{out} (e.g., headers and tails). This overhead will be indicated as $\Omega(\phi_{out}, \chi_{mac})$ (measured in $Bytes/sec$);
- a *control overhead*, which includes the control messages (e.g., synchronization packets and acknowledgements) that are exchanged between a node and the coordinator. These messages generate an energy dissipation due to their transmission/reception. The volume of control messages from the coordinator to the node and vice versa is defined as $\Psi_{c\to n}(\chi_{mac})$ and $\Psi_{n\to c}(\chi_{mac})$ (measured in $Bytes/sec$);
- a *timing overhead* per second, i.e., time intervals where the channel is unavailable, either because of the transmission of control messages or because the network is kept idle. This quantity is $\Delta_{control}(\chi_{mac})$;
- a *time discretization*. Since a protocol does not generally assign an arbitrary and continuous transmission time to each node. δ is the base time unit that is used in the selected protocol, thus transmissions' intervals are expressed as multiples of δ.

The goal of the network design is to size the transmission intervals to enable each node to deliver all its data and the corresponding control information. This is modeled as an assignment problem that is tailored for the typical star-topology TDMA transmission of WBSNs, but it can be also adapted to a contention access protocol (in fact, the

$\Delta_{tx}^{(n)}$'s can be statistically determined as the average amount of time a node can successfully transmit per second, as shown in [40] for the CSMA/CA). In particular, the MAC protocol has to find a number $k^{(n)}$ for each node n such that:

$$\Delta_{tx}^{(n)} = k^{(n)} \cdot \delta \geq T_{tx}\left(\phi_{out}^{(n)} + \Omega\left(\phi_{out}^{(n)}, \chi_{mac}\right)\right) \tag{4.16}$$

where $T_{tx}(\cdot)$ denotes the transmission time required to send the specified amount of data, and depends on the physical radio. Additionally, the assignment of the transmission intervals by the MAC protocol must be constrained in order not to exceed the total of one second:

$$\sum_{i=1}^{N} \Delta_{tx}^{(i)} + \Delta_{control}(\chi_{mac}) = 1 \tag{4.17}$$

From the DSE perspective, allowing the network to stay silent for a long time leads to good solutions in terms of energy consumption, but in practice it increases the data delay. Hence, we define the *delay* function $d(\chi_{mac})$ to quantify the average (or the maximum) time between the generation of the data and the instant it is received by the coordinator. Such a function cannot be defined in the general case, but it can be determined according to the specific MAC and the traffic patterns of the nodes, as we show in the case study.

Node Model A typical WBSN node follows the microcontroller-based architecture shown in Figure 4.21. This Section introduces a model that captures the interdependency among the hardware components in terms of consumption and application-related metrics, as well as the influence of the network configuration on the single node. The node has been characterized by means of a *configuration* χ_{node}, which includes the configurable parameters both on the hardware side (e.g., frequency, transmission power), and on the software side. All the parameters that cannot be tuned, or that are not relevant for a system-level optimization, will not be detailed in this model.

The node first samples the physiologic signal with a frequency f_s, and the samples are then quantized by an A/D converter to produce values of L_{adc} bytes, thus generating an input stream ϕ_{in} of $f_s \cdot L_{adc}$ (B/s). The sampling activity leads to an energy dissipation that can be expressed as:

$$E_{sensor} = E_{transducer} + [\alpha_{s,1} \cdot f_s + \alpha_{s,0}] \tag{4.18}$$

A linear function of f_s (with coefficients $\alpha_{s,1}$ and $\alpha_{s,0}$) captures the behavior of the A/D circuit [41], whereas $E_{transducer}$ is an overhead included by the transducer.

The input stream ϕ_{in} is then processed by an application, which typically consists of filtering or data compression. The behavior of the application layer is determined by a set of parameters (e.g., approximation factors and compression ratios), which determine three key aspects:

- the *output stream* ϕ_{out}. From a quantitative perspective, the application can be modeled as a function *h* that processes the input stream ϕ_{in} and produces a certain amount of results to be transmitted. As a consequence, the output of the node is equal to $\phi_{out} = h(\phi_{in}, \chi_{node})$ and, if an estimation of the transmission errors is available (e.g., [142]), then the average amount of retransmitted data can be added to the original ϕ_{out};

- the *resource usage*. The vector $\mathbf{u} = (Duty_{app}, M_{app}, \gamma_{app}, u_4, ..., u_n)$ represents these quantities. It contains n elements, one for each hardware resource that can be tuned on the target platform. A different notation is used to identify the duty cycle of the application on the microcontroller ($Duty_{app}$), the amount of memory required during the execution (M_{app}), and the number of memory accesses (γ_{app}), which will be used later for energy considerations. The resource usage depends on how the node is tuned (i.e., χ_{node}) and on the amount of data to be processed. Hence, a function vector $\mathbf{k} = (k_1, ..., k_n)$ is defined such that $\mathbf{u} = \mathbf{k}(\phi_{in}, \chi_{node})$, where $k_i(\phi_{in}, \chi_{node})$ computes the usage of resource i.
- the *output quality*. As the application generally introduces an approximation, let be $e(\phi_{in}, \chi_{node})$ an application-specific function that measures the loss of quality between the original and the transmitted data.

The execution on the microcontroller generates an energy dissipation that linearly depends on the duty cycle and on the operating frequency ($f_{\mu C}$) [41]:

$$E_{\mu C} = Duty_{app} \cdot [\alpha_{\mu C,1} \cdot f_{\mu C} + \alpha_{\mu C,0}] \tag{4.19}$$

The execution also leads to an energy consumption due to memory access, which can be estimated as follows [81]:

$$E_{mem} = \gamma_{app} T_{mem} \cdot E_{acc} + (1 - \gamma_{app} T_{mem})\, 8 M_{app} \cdot E_{idle}^{bit} \tag{4.20}$$

This equation includes two contributions: a dynamic consumption due to the γ_{app} memory accesses, and a residual that occurs during idle periods and is proportional to the memory size. In the equation, T_{mem} indicates the access time, E_{acc} defines the consumption of a single access, and E_{idle}^{bit} denotes the dissipation per bit due to leakage.

Finally, the output stream ϕ_{out} and the control information need to be transmitted to the coordinator by the radio unit during the assigned transmission intervals. The physical radio determines the transmission time in Equation (4.16) and the dissipation associated to the reception (E_{rx}) and the transmission (E_{tx}) of one bit, the latter being related to the power of the carrier signal [142], which must be chosen to achieve a low packet error rate. Thus, the energy consumption due to the radio can be expressed as:

$$\begin{aligned} E_{radio} = &[8\,(\phi_{out} + \Omega(\phi_{out}, \chi_{mac})) + 8\Psi_{n\to c}(\chi_{mac})] \cdot E_{tx} \\ &+ 8\Psi_{c\to n}(\chi_{mac}) \cdot E_{rx}\ . \end{aligned} \tag{4.21}$$

Then, after including the contribution of all the analyzed layers, the overall node consumption can be expressed as:

$$E_{node} = E_{sensor} + E_{\mu C} + E_{mem} + E_{radio}\ . \tag{4.22}$$

System-Level Evaluation Metrics The performance metrics of each node (i.e., E_{node} and $e(\phi_{in}, \chi_{node})$) into consistent network-level objective functions. As already stated, finding balanced configurations is a major concern while combining the different metrics, in order to avoid situations where the coordinator receives data of different quality, or where heavily optimized nodes are alternated to other nodes with an insufficient lifetime. As a consequence, the *network-level energy consumption* (E_{net}) is a weighted

combination of the average energy consumption of the nodes, and the sample standard deviation of this quantity over the WBSN:

$$E_{net} = \sum_{i=1}^{N} \frac{E_{node}^{(n)}}{N} + \vartheta \cdot \sqrt{\frac{1}{N-1} \sum_{i+1}^{N} \left[E_{node}^{(n)} - \sum_{i=1}^{N} \frac{E_{node}^{(n)}}{N} \right]^2}, \tag{4.23}$$

where ϑ is a positive constant that determines the importance of the balance among the nodes. A *network-level application quality* metric can be defined in a similar way, by combining all the loss-of-quality functions $e^{(n)}(\phi_{in}, \chi_{node})$ as we did in Equation (4.23) for $E_{node}^{(n)}$.

A Real-World WBSN Case Study The proposed multi-layer model for WBSNs can be easily used to model a real network that uses a commercial platform and widespread standards. The illustrative case study consists of a WBSN for electrocardiography (ECG) monitoring. It is assumed the following scenario: a hospital where N patients (in this example, $N = 6$) are wearing a node that is connected to a central base station. The nodes reduce the size of the output stream by applying one of the two available data compression techniques, i.e., digital wavelet transform (DWT) [33] and compressed sensing (CS) [105]. The two techniques have different properties in terms of complexity, signal quality and hardware requirements: for the sake of illustration, the half of the nodes employ DWT, and the remaining ones execute CS.

The target application is the SHIMMER commercial platform [42], which includes an ultra low-power microcontroller, 10*kB* of RAM memory, and an IEEE 802.15.4 [18] radio module. The transmission is performed using the beacon-enabled mode of the IEEE 802.15.4 MAC layer [18]. Considering the set of parameters on the node and the MAC protocol, the number of possible network configurations of this case study exceeds the tens of millions, thus making a deep DSE impractical by using network simulation or by collecting experimental data. The proposed model, on the other hand, contains all the structures that are needed to fully describe the target network.

IEEE 802.15.4 Network Model The proposed system-level WBSN model can capture the relevant dynamics of the beacon-enabled mode of the IEEE 802.15.4 [18] MAC protocol. In this protocol, a beacon is periodically sent by the coordinator to define the structure of the next *superframe*, a time interval whose structure is shown in Figure 4.22. The superframe is divided into an inactive and an active part, the latter being divided into 16 slots, 7 of which (known as *guaranteed time slots*, GTSs) are granted using a TDMA-like protocol.

The IEEE 802.15.4 MAC configuration is defined as

$$\chi_{mac} = \{L_{payload}, SFO, BCO, \Delta_{tx}^{(1)}, ..., \Delta_{tx}^{(N)}\}$$

where $L_{payload}$ is the payload in a data packet, and SFO and BCO denote the superframe and the beacon orders, which in turn determine the interval between two beacons (BI) and the duration of the active part (SD) (see Figure 4.22) [18]. Finally, the $\Delta_{tx}^{(n)}$'s indicate the transmission time allocated to each node.

The IEEE 802.15.4 MAC protocol can be easily mapped on the proposed model's equations. For example, the data overhead introduced by the MAC is equal to 13 bytes

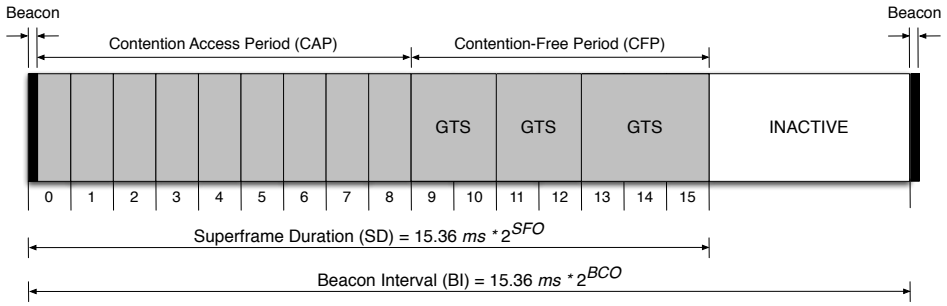


Figure 4.22: *Structure of the IEEE 802.15.4 superframe*

(11 for the header, 2 for the checksum) for each packet, hence $\Omega(\phi_{out}^{(n)}, \chi_{mac}) = 13 \cdot \phi_{out}^{(n)}/L_{payload}$. In terms of control overhead, the protocol does not require any control message from node (thus $\Psi_{n\to c}(\chi_{mac}) = 0$), whereas the coordinator sends a number of beacons (of variable length, which we denote as L_{beacon}) that depends on the number of superframes per second (i.e., $1/BI$), and an acknowledgment (4 bytes) for each transmitted packet, thus $\Psi_{c\to n}(\chi_{mac}) = 4 \cdot \phi_{out}^{(n)}/L_{payload} + L_{beacon}/BI$. Furthermore, $\Delta_{control}(\chi_{mac})$ is the time required by the coordinator to transmit $1/BI$ beacons per second, plus at least 9 slots reserved for contention access (which are not exploited in this case study), and the inactive period of the superframes.

The model can handle additional protocol-specific constraints on the assignment of the $\Delta_{tx}^{(n)}$'s. Firstly, the $\Delta_{tx}^{(n)}$'s cannot be arbitrarily assigned because of the time discretization imposed by the slots. Hence, the base transmission time δ is equal to the slot length, i.e., $SD/16$, and all the $\Delta_{tx}^{(n)}$'s are expressed as multiples of δ. Then, as the protocol specifies that at most 7 slots can be used as GTSs, the overall transmission time that can be allocated for the nodes is constrained as follows: $\sum_{i=1}^{N} \Delta_{tx}^{(i)} \leq 7/16 \cdot SD/BI$.

Finally, thanks to the nature of data compression that leads to a uniform output rate, a simple delay model (based on the one in [83]) can be formulated. In particular, the worst-case delay for a node n occurs when the remaining nodes use all their slots (and the control overhead for all the corresponding frames) before node n is enabled to transmit:

$$d^{(n)}(\chi_{mac}) \leq \sum_{i=1,\, i\neq n}^{N} \Delta_{tx}^{(i)} + \left\lceil \frac{1}{7} \sum_{i=1,\, i\neq n}^{N} \Delta_{tx}^{(i)} \right\rceil \Delta_{control} \,. \tag{4.24}$$

Shimmer Node Model Since the SHIMMER platform is already implemented, some parameters are fixed. In particular, the sampling frequency is determined by the nature of the ECG signal and is fixed to f_s=250Hz, and the resolution L_{ADC} of the A/D converter is set to 12 bits, thus generating a constant input stream $\phi_{in} = 375$ B/s. The contribution of the 10kB memory block is also constant, as the memory accesses are determined by the *Shimmer*-specific implementations of the DWT and CS algorithms [105]. At the radio level, the power of the carrier signal has been set to a sufficient level in order to minimize the probability of a packet error, thus avoiding an increment of ϕ_{out} due to retransmission. Hence, the configuration of a node is characterized as $\chi_{node} = \{CR, f_{\mu C}\}$, where CR is the compression ratio, and $f_{\mu C}$ is the

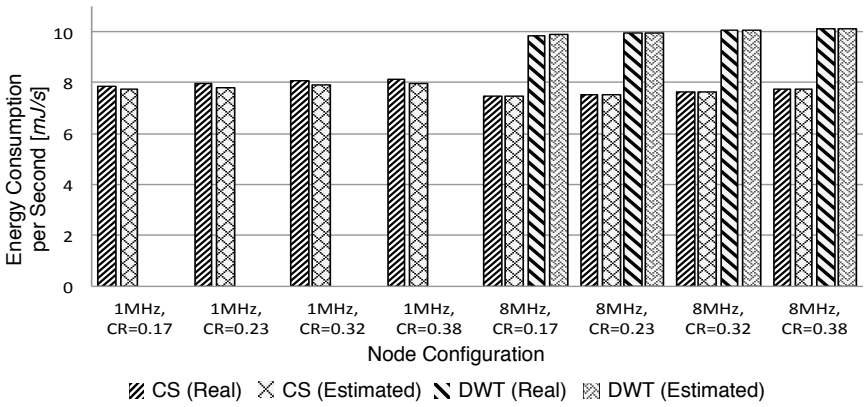


Figure 4.23: *Estimation of the node consumption with different configurations*

frequency of the microcontroller.

The output stream ϕ_{out} can be easily expressed as a function of CR, i.e., $\phi_{out} = h(\phi_{in}, \chi_{node}) = \phi_{in} \cdot CR$, which holds for both the DWT and the CS applications. However, the two compressions show different duty cycles and loss-of-quality functions. The duty cycle of the Shimmer implementations of DWT and CS show a marginal dependency on CR, but there is a relation with respect to $f_{\mu C} \in \chi_{node}$. By analyzing the execution, we can define the resource usage function as $\mathbf{k}(\phi_{in}, \chi_{node}) = (k_{DWT}, k_{CS}) = (2265.6/f_{\mu C},\ 388.8/f_{\mu C})$. To estimate the quality of the application, similarly to the previous example, the *percentage root-mean-square difference* (PRD) [105], which quantifies the difference between the original ECG and the one reconstructed by the coordinator, is considered. Although the actual PRD value can only be determined by measuring or simulating the actual reconstructed signal, an analytical estimation is computed using two fifth-order polynomial functions $P_5^{(DWT)}(CR)$ and $P_5^{(CS)}(CR)$ that fit the experimental data provided in [105].

Experimental Results The first set of experiments aims at validating the estimation provided by the model. The model equations have been validated against the actual experimental data obtained under different operating conditions. Figure 4.23 shows the estimation of the overall energy consumption of the nodes with set of realistic configurations χ_{node}. The energy estimation proves to be very accurate, as the average error on all the $f_{\mu C}$'s and CR's is equal to 0.88% for the CS, and to 0.13% for DWT, and the maximum error does not exceed 1.74%. The model also predicts that the DWT cannot complete its execution with $f_{\mu C} = 1$ MHz because its duty cycle exceeds 100%. Figure 4.24 shows the estimation error for the PRD's, which proves to be very low (0.92% for the CS, 0.46% for the DWT), thus showing that the model accurately estimates a crucial metric that can be exactly determined only by analyzing or simulating the actual compressed ECG.

To validate the network model, the estimated delay has been compared to the results of a network simulation performed using the popular *Castalia* framework [9]. The choice of a network simulator over experimental data is justified by the possibility of deeply monitoring the packet flow. In spite of being a worst-case estimation, the delay function in Equation (4.24) provides an average overestimation lower than 100 ms over

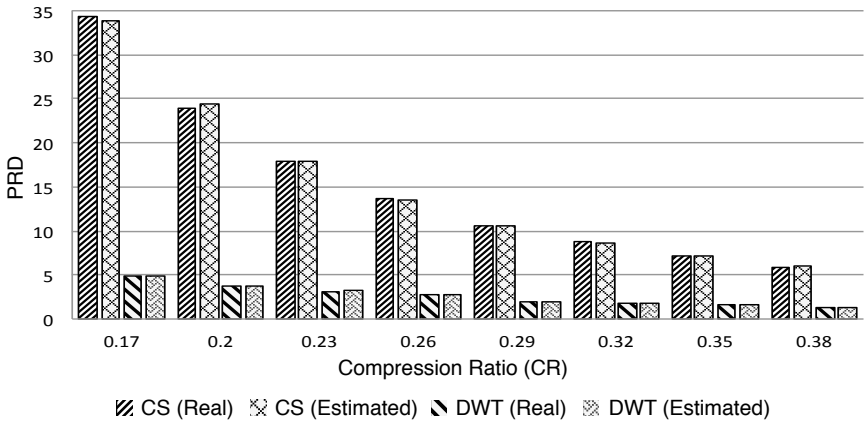


Figure 4.24: *Estimation of the application behavior by means of the PRD metric*

a set of 130 simulations with realistic ϕ_{out}'s and χ_{mac}'s, which is acceptable in this application.

Design Space Exploration Performance The proposed WBSN model has been employed in a set of multi-objective optimization techniques, including genetic algorithms (which have been already used in the WSN domain [163]) and simulated annealing [121], without experiencing any relevant difference in terms of quality of the solutions. However, in terms of execution time, the proposed evaluation clearly outperforms a complete network simulation, in fact, a network simulation takes 5 to 10 minutes in our case study, while the model can be evaluated approximately 4800 times per second.

Figure 4.25 shows the optimal tradeoffs between the three metrics included in our model and, in order to underline the importance of considering all these metrics, these solutions are compared to the ones found by using a state-of-the-art energy/delay model [88]. It can be observed that the Pareto set generated according to the energy/delay model only contains a subset (i.e., approximately 7%) of the tradeoffs that are found using the proposed model: this is due to the fact that the energy/delay model does not include an additional application-aware metric. As a consequence, it only approximates the energy/delay curve, but it does not allow the DSE algorithm to recognize the solutions that are optimal in terms of PRD. In order to detect the large number of Pareto tradeoffs characterized by acceptable mid-range PRD's, the proposed multi-layer model must be employed.

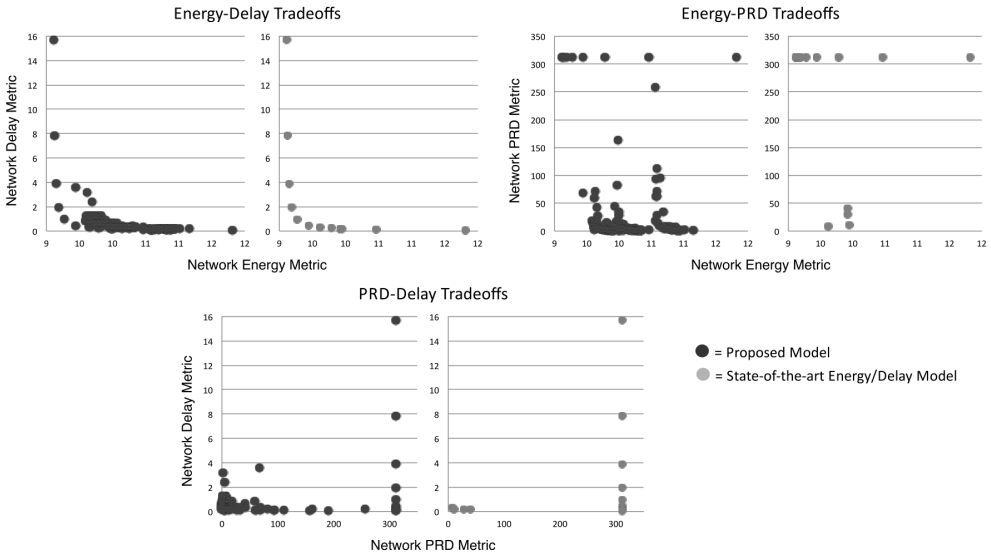


Figure 4.25: *Tradeoffs detected using the proposed model and a state-of-the-art energy/delay model [88]*

Summary This example highlights the benefits of a quick and accurate analytical evaluation in the context of model-based design of WSNs. Although WSNs show a wide range of different characteristics in different fields, it is possible to formulate abstract system-level models for broad classes of networks, which share common architectural and network structures, or more generally belong to the same domain. As a proof of concept, the class of WBSNs has been considered in order to prove that a general and comprehensive model can be defined and it can be applied to real networks with a low effort and a high accuracy. The results on a real case study show that the estimation error for energy and performance never exceeds 1.74% with respect to real data, while the estimation time is up to six orders of magnitude lower than an evaluation performed by a network simulator.

4.4.3 Hybrid Design Space Exploration

A critical aspect of the Design Space Exploration (DSE) is the evaluation of a WSN configuration. Among the possible evaluation techniques, network models provide a deep understanding of the network dynamics that can be exploited during the DSE, but their definition is generally difficult and requires a deep knowledge of the working domain [29]. As a consequence, the evaluation is typically performed using extensive network simulations [56], which are more accurate and reusable. However, a WSN simulation may take from several minutes to hours (depending on the network size) to be completed, and it should be repeated tens of times (generally more than 30) to mitigate the effects of randomness and ensure reliable statistical results. Moreover, a simulator can be considered as a black box that does not provide any information about the inner dynamics of the network, therefore no reasoning can be performed by the DSE algorithm. As a consequence, simulation is employed in the context of semi-random algorithms (e.g. genetic algorithms or simulated annealing), which however require a high number of evaluations – and hence an unacceptable amount of time – to converge.

4.4.3.1 KNOWLEDGE-BASED DESIGN SPACE EXPLORATION OF WIRELESS SENSOR NETWORKS

The technique that will be introduced in this Section has already been presented in [62]. It is a knowledge-based DSE algorithm that effectively tackles the design complexity of WSNs, by combining the domain knowledge of the analytical models with the generality and the flexibility of network simulators. Unlike semi-random algorithms, which require an evaluation at each step, the proposed approach performs several moves by relying on a set of domain-specific rules, thus greatly reducing the number of simulations. For this purpose, the *Markov decision process* (MDP) algorithm [32] has been adapted to the WSN domain. Moreover, its inherited scalability issues have been tackled to face the large design spaces of WSNs. In addition, a general domain knowledge has been introduced for the most popular WSN MAC layer, i.e., the IEEE 802.15.4 [18], which can be reused on any WSN based on this protocol. Validation results prove that the proposed approach effectively reduces the number of simulations with respect to state-of-the-art semi-random algorithms from 60 to 87% on multiple real-world scenarios.

State of the Art Design space exploration is a well established research topic in many different fields, such as embedded architectures or system-level design. However, automatic DSE is still immature in WSN design, where the complexity of the network and the peculiarity of each application domain often push the developers towards manual and *ad hoc* optimizations (e.g., the work in [68] for wildlife monitoring).

The main obstacle towards the complete automation of the DSE of WSNs is a rapid and accurate evaluation of a solution. Currently, the two evaluation techniques [29] are models and simulations. Analytical models describe the WSN dynamics by means of a set of equations that provide a *white-box* view of the system, thus extensive analysis are possible. When a model-based evaluation is employed, the DSE is considerably faster compared to simulation-based approaches, but the accuracy of the model is typically guaranteed only for specific domains or for specific aspects of the WSN. In fact, only models that have been proposed for single classes of WSNs (e.g., [151] for body area networks) or for specific protocols (e.g. [82], [40] and [129] for three different operations of the IEEE 802.15.4 MAC layer) show an accuracy that is comparable to a simulation. On the other hand, simulations are usually more precise and reusable, and they better profile all the aspects related to communication, energy and resource usage of distributed applications. However, a robust simulation takes several from minutes to hours, thus making it impractical for extensive DSEs.

In this context, the evaluation technique heavily affects the choice of the optimization algorithms. When an analytical description of the WSN (or, at least, of a specific part of it) is available, the optimization can be performed using *ad hoc* heuristic algorithms (e.g., in [61] to solve the network connectivity problem) or efficient techniques such as convex optimization (e.g., [118] for energy/delay optimization). The simulation-based estimation, on the other hand, has a black-box nature that does not allow any analytical consideration during the execution of the algorithm. This limitation leads to the employment of semi-random approaches such as genetic algorithms (e.g., [56] for placement and role assignment), multi-objective evolutionary algorithms (e.g., [119] for gossip-based WSNs), simulated annealing and Bayesian networks (e.g., [112] for a cross-layer optimization of cognitive wireless networks). However, given

Table 4.1: *Speed and accuracy of existing DSE techniques*

	Low Accuracy	**High Accuracy**
Slow		Simulation-based ([56] [119] [112])
Fast	Model-based ([61] [118])	MDP

the high execution time required by a simulation, the class of semi-random algorithms does not scale well on large design spaces that are typical of real-world WSNs.

Model-based and *simulation-based* exploration offer a speed/accuracy tradeoff that is summarized in Table 4.1. The proposed technique combines the high accuracy and reliability of a solution that is evaluated by a simulator, and the high speed that can be achieved when we include model information, calibrated with the simulator, in the DSE. In particular, the rationale is to move within the design space by exploiting the model information until they are accurate, and then use the simulation whenever it is strictly necessary.

Markov Decision Process The proposed knowledge-based design space exploration algorithm for the WSN domain is based on a discrete-space Markov decision process (MDP). The classical MDP approach, which has been successfully applied in other domains such as multiprocessor systems design [32], has been tailored to the WSN domain in order to enhance its performance and to increase its scalability. The proposed algorithm combines the available domain knowledge – which may come from an analytical model or by an analysis of the specific application – with a simulation-based network evaluation, in order to obtain an accurate and yet efficient DSE for WSNs.

In this approach, the DSE is considered as a path from the initial configuration P to a final configuration $\widehat{P}$. The path is identified by applying sequential transformations on the parameters, and the quality of these transformations is evaluated thanks to both models and simulations. The configuration P is composed by a set of parameters such as $P = \{p_1, ...p_k...p_n\}$ and an action $a \in A$ specifies how a configuration should be modified (i.e. *"double the CPU frequency"*). Actions transform the configurations as follows:

Definition 1. *Given a configuration of parameters $P = \{p_1, ...p_k...p_n\}$ and an action $a \in A$, a transformation $\tau(p_k, a)$ produces a new configuration $P' = \{p_1, ...p'_k...p_n\}$ where $p_k \neq p'_k$*

Once a transformation $\tau(p, a)$ has been performed, its effect on the metrics is evaluated using *movement vectors*:

Definition 2. *A movement vector is a vector of intervals in the metrics space corresponding to a transformation vector in the parameter space defined as:*

$$\Phi = \langle f_1(\tau(p_k, a)), f_2(\tau(p_k, a)), ...f_i(\tau(p_k, a)), \rangle$$

where $i = |M|$, and $\overrightarrow{f} = f_1, f_2, ..., f_i$ are functions that determine the effect of the transformation τ on each metric $m_j \in M$.

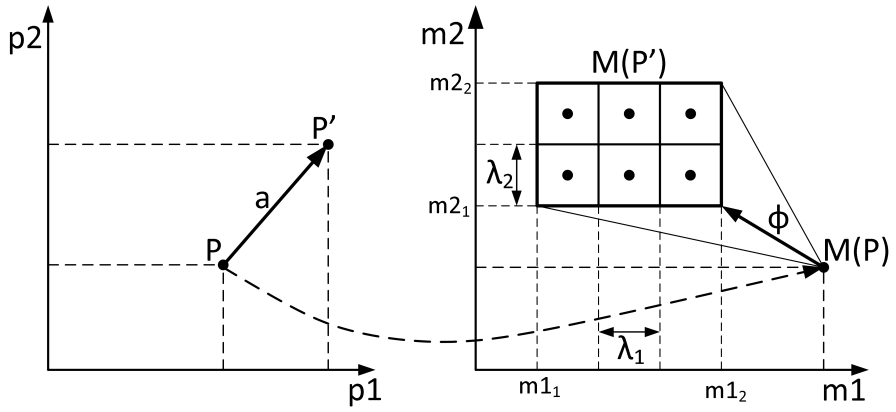


Figure 4.26: *An example of actions on P. The metrics space is partitioned in 6 areas and centroids are illustrated with black dots*

Movement vectors are used to estimate the metrics of a configuration P' generated from an action a applied to a configuration P. For each metric $m_i \in M$, an interval in the metrics space $[m_i^L, m_i^H]$ specifies the range where the actual (real) value of the metric is included. $\overrightarrow{f}$ functions and movement vectors Φ are problem-specific (based on a model), thus problem-specific models are required.

For instance, the effect of the action *"double the operational frequency of a processor"* could in the worst case increase of up to two times of the energy consumption (m^H) or, in the best case, leave it unaltered (m^L). Thus, the movement vector associated with this action is $[E, 2E]$, where E indicates the current energy consumption. It indicates that, whereas the action *"double the operational frequency of a processor"* is applied on a configuration P, the energy consumption of the resulting configuration P' belongs to the interval $[E, 2E]$.

To tackle exploration accuracy, the metrics space is partitioned according to a parameter $\overrightarrow{\lambda}$ defined for the exploration. $\overrightarrow{\lambda}$ is a vector of scalars that specifies the maximum width of a partition for each metric. Partitioning is used to divide large areas into smaller areas such as the maximum error is defined by λ, given by the difference between the present value of the metrics (to be determined through simulation) and the centroid of the partition that best approximates it. An example of an action on a parameter is illustrated in Figure 4.26 where action a, applied to P, generates $P' = \tau(P, a)$. The metrics $M(P')$ are evaluated from $M(P)$ through domain knowledge Φ and results in an area included into $\langle [m1_1, m1_2], [m2_1, m2_2] \rangle$. The area is partitioned in six partition (whose centroids are depicted with black dots) according to $\overrightarrow{\lambda}$.

The utility function (Ψ) has the utility property [140] and is a function of the metrics M. In a multi-objective exploration, metrics are combined in order to be able to enumerate the solutions. Ψ can be linear ($w_1 m_1 + w_2 m_2 + ... + w_i m_i$) or exponential ($m_1^{w_1} + m_2^{w_2} + ... + m_i^{w_i}$). To obtain the Pareto curve, the exploration is performed in a multivariate environment, thus $\overrightarrow{W} = \{w_1, w_2, ...w_i\}$ must change during the exploration in order to explore all the directions of the metrics space. In a two-dimensional metrics space, this can be achieved, i.e., using the scalarizing function $m_1^{\alpha} m_2^{(1-\alpha)}$ with α in $[0, 1]$.

The exploration is modeled as a MDP:

Definition 3. *A MDP is a tuple* $< S, A, TP, R >$ *where:*

- S *is the set of possible states describing a solution of the DSE problem;*
- A *is the set of possible actions that can be applied on the states;*
- $TP : S \times A \rightarrow \Pi(S)$ *is the state transition function as the probability density function for every state-action pair;*
- $R : S \times A \times S' \rightarrow R$ *is the expected reward for each state-action pair.*

The behavior of the MDP is described with the Decision Tree D, a tree-based structure used to evaluate the various configurations. A node $s \in S$ in D represents a state of the system expressed by the tuple $\langle P, M \rangle$, where P is a point in the parameters space and M a point in the metrics space. An edge $e \in E$ is defined as:

$$e = \langle s_i, s_j, a, TP(s_i, a, s_j) \rangle$$

and it represents the transition probability (given by the transition function TP) from state s_i to state s_j when action a is applied.

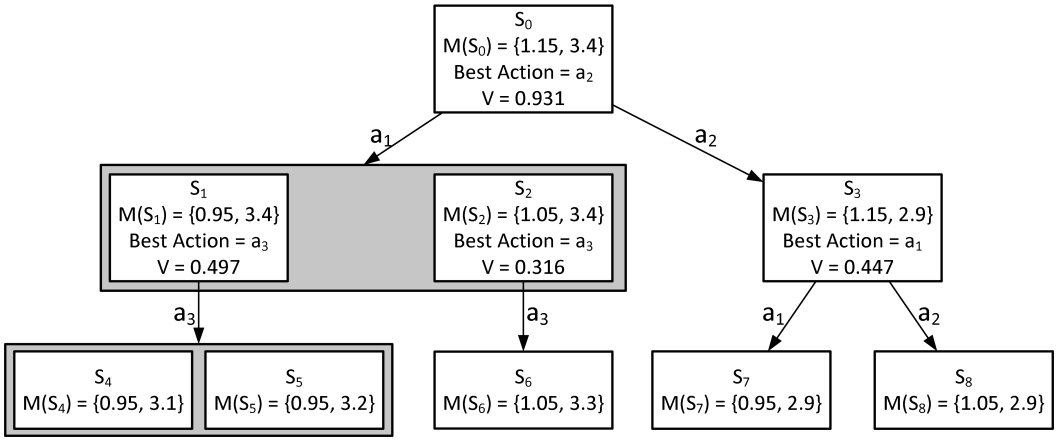


Figure 4.27: *Example of a Decision Tree D. Cumulative return V is computed using the Value Iteration Algorithm (see Algorithm 4)*

Each partition identifies a new node in the decision tree, and the reward of the actions (R) is computed as the difference between parent (Ψ_p) and child (Ψ_c) utility functions:

$$R = \Psi_p - \Psi_c \tag{4.25}$$

For each partition $\langle s, a, s' \rangle$, the probability $TP_i(s_i, a, s_k)$ is computed as the number of times $\langle s, a \rangle$ ends in s' when traversing the decision tree from root to leaf. The decision tree is progressively built by iterating on the newly generated nodes breadth-first, until either is not possible to apply any other action on the leaf nodes or the l-th level is reached. During the creation of D, in order to avoid useless exploration, *opposite* actions are included into a *forbidden list* such as they will not be applied. For example, if the action *Increase CPU frequency* is applied, the action *Decrease CPU frequency* in included into the *forbidden list* since it voids the previous action.

An example of the Decision Tree D is shown in Figure 4.27. From the initial state s_0, two actions are applied (a_1, a_2) resulting in three states (s_1, s_2, s_3) where s_1 and s_2

are two partitions. Actions a_3 and a_1 are further applied on the second level of the tree. For each state, metrics are associated; this example has two metrics that must be minimized. Rewards estimate the benefits of the actions on Ψ and are used to guide the exploration to better solutions. To determine the reward of the actions, Value Iteration Algorithm (see Algorithm 4) is used on D. For all the states in the tree and for all the available actions, the reward of an action $Q(s, a)$ is computed by adding the reward of choosing that action ($R(s, a, s')$) with the cumulative return on the destination node ($V(s')$) scaled by γ. γ is a scalar value ($0 \leq \gamma \leq 1$) used to control the influence of (expected) cumulative returns. Transition probability $TP(s, a, s')$ is considered in the formula. In this specific example, V is computed by considering $\gamma = 0.6$ and the utility function $m_0^{1-\alpha} + m_1^{\alpha}$ with $\alpha = 0.5$. According to the metrics, the algorithm identifies a_2 as the best action in this situation; in fact, s_7 is optimal with respect to the given metrics. The new state is s_3 and no simulation is required since no uncertainty is detected here.

Algorithm 4: Strategy Evaluation Algorithm

```
1 initialize V(s) = 0
2 repeat
3     forall the s ∈ S do
4         forall the a ∈ A do
5             Q(s,a) = Σ_{s'∈S} TP(s,a,s')[R(s,a,s') + γV(s')]
6         end
7         V(s) = max_a Q(s,a)
8     end
9 until strategy converges;
```

The overall exploration strategy is illustrated in Algorithm 5. It starts (**first step**) by generating an initial configuration P_0 (line 32) – that can be identified randomly or pseudo-randomly. Once P_0 has been evaluated by simulation and the initial state s_0 has been generated (line 6), the set of states to be examined ($\overline{S}$) is initialized (line 7).

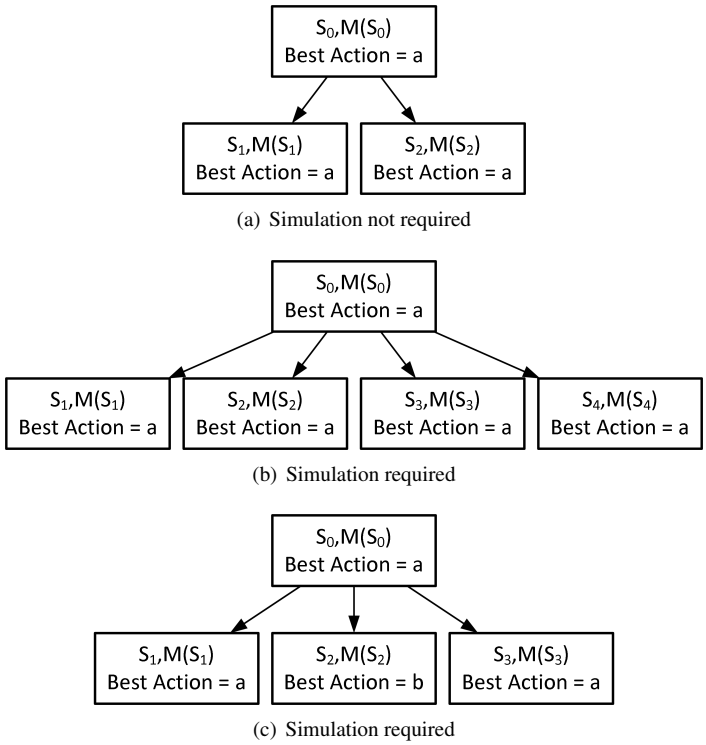
The **second step** of the algorithm solves the MDP. For each state in $\overline{s}$, all possible actions are applied, generating the configurations that differ from s_0 by one parameter (lines 13-26). For all the generated configurations (obtained by applying a in s_i), s_k metrics are partitioned, D and $TP(s_i, a, s_k)$ are updated. The generation of the Decision Tree continues until no new states are available or the maximum depth l has been reached. At this point (line 27) the best value iteration algorithm [140] is applied and the *best_action*s are updated on D.

The **third step** applies the $best\_action$ of s_0 in s_0 to get the set of reachable states (NS). At this point, three situations are possible:

1. **No Uncertainty**: the action can lead to a single state. The action is determined with an accuracy λ and no simulation is necessary;
2. **Uncertainty of the First Kind**: the action leads to a set of states and the algorithm maps the same action to all of them. It means that, whichever state the system will end into, the same next action is chosen. In case the amount of states is below a given threshold K, simulation is not required and parallel exploration will follow, otherwise simulation is required; K controls the amount of solutions to be

Algorithm 5: Overall Exploration Strategy

```
1  Identify the configuration parameters P⃗
2  Define the movement vectors Φ
3  num_runs=0
4  repeat
5      Generate an initial configuration P_0
6      s_init = Simulate(P_0)
7      Initialize the set of states to be examined S̄ = s_init
8      repeat
9          depth = 0
10         reset D
11         get an element s_0 ∈ S̄,  S̄ = S̄ − s_0
12         s̄ = s_0
13         repeat
14             s̄_new = {}
15             forall the s_i in s̄ do
16                 forall the a applicable in s_i do
17                     apply a in s_i creating child nodes s_k
18                     partition s_k metrics
19                     update D and TP(s_i, a, s_k)
20                     s̄_new = s̄_new + s_k
21                     add a to forbidden action list of s_i
22                 end
23             end
24             s̄ = s̄_new
25             depth++
26         until s̄ = ∅ or depth == l;
27         Value Iteration Algorithm(D)
28         NS = τ(s_0, best_action(s_0))
29         if |NS| ≥ K or (∃s_i, s_j ∈ NS: best_action(s_i) ≠ best_action(s_j) and i ≠ j) then
30             s_next = Simulate(P_{s_0})
31             if s_next ∉ NS then
32                 Error ⟶ Restart From Line
33             end
34             NS = s_next
35         end
36         if convergency then
37             Simulate(P_{s_0})
38         else
39             S̄ = S̄ + NS
40         end
41     until S̄ = ∅;
42     num_runs++
43 until num_runs ≥ MAXRUNS;
44 Change Utility Function
45 Repeat All
```



(a) Simulation not required

(b) Simulation required

(c) Simulation required

Figure 4.28: *Uncertainties of First and Second Kind with* $K = 3$

explored and is used to tackle the scalability of the algorithm. Parallel explorations could evolve differently since they start from different partitions, thus they must be considered separately;

3. **Uncertainty of the Second Kind**: the action leads to a set of states, but the algorithm maps two or more different actions on those states. In this situation, simulation is needed to determine to which state the action really leads to.

Three examples of uncertainties are shown in Figure 4.28. In 4.28(a) an uncertainty of the first kind is detected and $|NS| < K$ thus simulation is not required. In 4.28(b) an uncertainty of the first kind is detected but $|NS| \geq K$ thus simulation is required. In 4.28(c) simulation is required since an uncertainty of the second kind is detected.

In other terms, simulation is performed only if the cardinality of NS (the amount of states reached by the best action) is above a given value K or the action a brings the system to *a set* of states and the Value Iteration Algorithm mapped different actions to each of those states. In case simulation results s_{next} are not contained in NS the exploration restarts from line 32, otherwise NS is updated with the actual value s_{next}. At this point the algorithm checks if convergence has been reached (line 36); in the positive case, a simulation is performed (line 37) to get the real values of the metric of s_0 (unless it has been previously simulated), otherwise, $\overline{S}$ is updated and the exploration continues.

Step one, two and three are repeated until a maximum number of runs (*MAXRUNS)* has been performed.

In the **last step**, the weights $\overrightarrow{w}$ of the utility function Ψ are updated (line 44) and

the algorithm is entirely repeated. It allows to *change the direction* of the exploration in order to cope multi-objective optimizations. The algorithm stops when no more utility functions can be used.

In conclusion, the algorithm explores all the possible actions for all the reachable states with an event horizon of l to determine an optimal local action considering various sequences of actions. The effectiveness of the approach strictly depends on the quality of the movement vectors; as the accuracy of movement vectors increases, the amount of required simulations decreases. In fact, in the optimal case, movement vectors identify areas with size lower than λ, thus *No Uncertainty* is detected. In this case, assuming that movement vectors are accurate, simulation is required at the beginning (line 6) and end (line 37) only, thus simulations are minimized and exploration speed is maximized. In the worst case, non accurate movement vectors may conduct to *Uncertainties of First* (with $|NS| > K$) or *Second Kind*, thus simulations are required at each step, leading to slower explorations. More generally, to control the accuracy and the speed of the exploration (number of states in the MDP) two mechanisms have been identified:

- **Control the minimum desired accuracy with** λ. It identifies the size of the partitions during the generation of the Decision Tree; increasing λ reduces the number of partitions (states), improving the evaluation speed, but it increases the approximation error;
- **Define a good event horizon** l, which determines the maximum depth of the Decision Tree. It limits the number of steps required for the creation/evaluation of the MDP. Reducing l improves the speed of MDP evaluation since fewer states are generated into the decision tree D. On the other hand, the higher is l the higher is the lookahead of the algorithm.

Domain Knowledge Definition For IEEE 802.15.4 Networks The proposed approach requires the movement vectors to guide the exploration to optimal solutions. A domain knowledge definition for the IEEE 802.15.4 MAC layer is presented here. The proposed MDP does not require accurate movement vectors to operate [32], thus users do not need to provide accurate models to use this methodology; moreover, movement vectors are reusable. However, this analysis has two main goals: first, to show in practice how to build a set of domain-specific rules to exploit the potential of the MDP algorithm. Second, it provides a good characterization of one the most popular MAC protocols in WSNs, hence the proposed rules can be reused.

The IEEE 802.15.4 standard [18] has been introduced to satisfy energy requirements of emergent devices. The protocol is quite common, thus the proposed knowledge domain can be useful for future works. The results presented here are based on the models presented in [102] [151] [82] [40] [67] [129]. For the details of the protocol, please refer to Section 2.6 of this thesis.

The metrics of interests are Average Energy Consumption (E), to be minimized, and Percentage of Packets Received (P), to be maximized. The IEEE 802.15.4 is characterized by a certain amount of node and network parameters. For the sake of simplicity, the analysis has been restricted to four parameters: **SuperframeOrder**, **BeaconOrder**, **enableCAP** and **requestGTS**. According to the state of the art, these parameters have

considerable effects on the metrics of interests, thus their optimization is important to the final design. In all the equations the following constraint must be satisfied:

$$(0 \leq P \leq 100) \wedge (E \geq 0) \tag{4.26}$$

In case the equation is not satisfied, P and E are set to the nearest value which satisfies the equation. I.e., if $P < 0$ then P is set to zero. The next two sections presents in detail the movement vectors. In all the equations, E and P represents the actual value of the metrics while $\widehat{E}$ and $\widehat{P}$ represents their estimated (next) value.

Beacon Order and Superframe Order Beacon Order (BO) and Superframe Order (SO) define the main structure of the superframe since they determine the distance between the beacons and the size of the active period. The ratio between SO and BO defines the duty cycle between active and inactive period. The overall effect of increasing BO and decreasing SO is similar, since both actions modify the duty cycle in the same way. Increasing BO or decreasing SO will halve the duty cycle, thus both E and P can be reduced by 2. Resulting movement vectors to actions **increase BO** and **decrease SO** are:

$$\widehat{E} = \left[\frac{E}{2}, E\right] \tag{4.27}$$

$$\widehat{P} = \left[\frac{P}{2}, P\right] \tag{4.28}$$

On the other hand, actions that **decrease BO** and **increase SO** have an opposite behavior, since duty cycle is doubled:

$$\widehat{E} = [E, 2E] \tag{4.29}$$

$$\widehat{P} = [P, 2P] \tag{4.30}$$

All these actions can be applied if and only if the constraint:

$$FO \leq BO \tag{4.31}$$

is satisfied. This constraint is imposed by the standard [18].

Guaranteed Time Slots Each node requires a certain amount of GTS to the coordinator, according to the *requestGTS* parameter. The coordinator assigns the GTS according to the policy *first come first served* (FCFS). It implies that if requested GTS are not designed properly, performances of the network dramatically decreases.

The maximum amount of available slots in slotted IEEE 802.15.4 is given by the following formula:

$$M = NSS - \left\lfloor \frac{minCAP}{BSD * 2^{FO}} \right\rfloor \tag{4.32}$$

where NSS is the *Number of Superframe Slots*, minCAP is the minimum number of symbols in CAP and BSD is the *Base Slot Duration*.

The average amount of slots per node is equal to:

$$A = \frac{M}{N} \tag{4.33}$$

and the overall amount of requested slots is equal to:

$$U = \sum_{i=0}^{N} G(i) \tag{4.34}$$

where $G(i)$ represents the value of *requestGTS* of node *i*. From [129] and experimental results, it can be noted that, increasing the GTS requests improves P and reduces E. P is improved because contention is reduced and E decreases since nodes wake-up are scheduled more efficiently in CFP with respect to CAP. In particular, when a GTS slot is allocated and few packets are in the buffer, the node sleeps during CAP and wake-ups just at the beginning of the GTS slot. In addition, since IEEE 802.15.4 uses a TDMA protocol during CFP, no additional energy is required to perform *Carrier Sense*.

According to the FCFS policy, if GTS have been already allocated and it is not possible to satisfy the request, the request is rejected and the node is obliged to communicate into the CAP. GTS requests are rejected if the amount of requested GTS slots (Equation 4.34) overcomes the maximum amount of slots in CFP (Equation 4.32). Moreover, in order to balance GTS requests, the amount of GTS requests per node n ($G(n)$) should not overcome the average GTS requests (Equation 4.33). The effect of increasing/decreasing GTS is limited to a single-slot of a single-node, so the value of P and E should be divided by NM to provide more accurate movement vectors.

Action **increase requestGTS** on node n results in:

$$\widehat{E} = \begin{cases} \left[E - \frac{E}{NM}, E\right] & \text{if } U \leq M \wedge G(n) \leq A \\ \left[E, E + \frac{E}{NM}\right] & \text{otherwise} \end{cases} \tag{4.35}$$

$$\widehat{P} = \begin{cases} \left[P, P + \frac{P}{NM}\right] & \text{if } U \leq M \wedge G(n) \leq A \\ \left[P - \frac{P}{NM}, P\right] & \text{otherwise} \end{cases} \tag{4.36}$$

On the other hand, action **decrease requestGTS** has the following movement vectors:

$$\widehat{E} = \begin{cases} \left[E, E + \frac{E}{NM}\right] & \text{if } U \leq M \wedge G(n) \leq A \\ \left[E - \frac{E}{NM}, E\right] & \text{otherwise} \end{cases} \tag{4.37}$$

$$\widehat{P} = \begin{cases} \left[P - \frac{P}{NM}, P\right] & \text{if } U \leq M \wedge G(n) \leq A \\ \left[P, P + \frac{P}{NM}\right] & \text{otherwise} \end{cases} \tag{4.38}$$

Enable CAP Considering the enableCAP parameter, two actions are possible: activate and deactivate CAP. The overall effect of enabling CAP is the increase of energy consumption (due to CAP period) and an increase of packets received. Similarly to GTS, the action has an effect on a single node only, thus both E and P changes are scaled to N. Moreover, the higher is the amount of GTS requests of a node, the lower is the effect of activation/deactivation of CAP, so the metrics are divided by G(n)+1.

Differently from energy, packets received have a known behavior in case $G(n)$ is equal to zero. In fact, when $G(n) = 0$ and CAP is not enabled, the amount of packets sent by a node n is equal to zero, thus activating CAP when $G(n) = 0$ has a increases the number of packets received ($\left[\frac{P}{N}, \frac{100}{N}\right]$); the deactivation of CAP when $G(n) = 0$ has the opposite effect. The movement vectors of the action **activate CAP** are:

$$\widehat{E} = \left[E, E + \frac{E}{N(G(n)+1)}\right] \tag{4.39}$$

$$\widehat{P} = \begin{cases} \left[P + \frac{P}{N}, P + \frac{100}{N}\right] & \text{if } G(n) = 0 \\ \left[P, P + \frac{P}{N(G(n)+1)}\right] & \text{otherwise} \end{cases} \tag{4.40}$$

for the action **deactivate CAP**, the movement vectors are:

$$\widehat{E} = \left[E - \frac{E}{N(G(n)+1)}, E\right] \tag{4.41}$$

$$\widehat{P} = \begin{cases} \left[P - \frac{100}{N}, P - \frac{P}{N}\right] & \text{if } G(n) = 0 \\ \left[P - \frac{P}{N(G(n)+1)}, P\right] & \text{otherwise} \end{cases} \tag{4.42}$$

Initial Points Selection In the classical MDP, the set of initial points is randomly generated but, especially with large design spaces, the probability to obtain bad (or even unfeasible) results is high. Therefore, the selection of the initial points can be guided by the model since, considering that the knowledge base has been already created to compute the actions, the same information can be used to extract the initial points.

To define the rules for the selection of the initial points, we conduct several experiments on star networks with 4, 6 and 8 nodes with various packet rates (5, 15, 30, 50, 65, 80 $\left[\frac{pkts}{sec}\right]$) in order to cover a large set of applications. All the experiments where conducted with Castalia simulator [9] (see next Section). From the experimental results, we notice that assigning a *requestGTS* greater than the average (A), reduces the quality of the solution, so we suggest to create the initial population with a starting value of *requestGTS* randomly chosen in the interval $[0, A]$. In addition, a value of Beacon Order lower than 3 or greater than 12 does not usually provide good results, thus we generate the initial solution with BO included into $[3, 12]$. Another aspect concerns the duty cycle ($\frac{FO}{BO}$); good solutions usually have a duty cycle included into $[0.25, 0.85]$ in all the configurations.

Summarizing, to determine the initial points, we propose to generate the set of initial points in such a way:

$$\begin{cases} 0 \leq G(n) \leq A & \forall n \in N \\ 3 \leq BO \leq 12 \\ 0.25 \leq \frac{FO}{BO} \leq 0.85 \end{cases} \tag{4.43}$$

so that these constraints are all satisfied.

Experimental results in the next Section show that a considerable improvement on search efficiency is obtained if initial points are determined using this technique.

Experimental Results The proposed approach has been validated on two sets of experiments. Each solution is evaluated with Castalia [9], a popular simulator for Wireless Sensor Networks and Body Area Networks, based on the OMNeT++ framework [155]. The design space for all the experiments is presented in Table 4.2. The design space has been explored in four different scenarios described in Table 4.3. These scenarios have been chosen in order to cover a large set of applications (i.e. Body Area Networks [91]).

Table 4.2: *Explored Design Options for the Experimental Results*

Parameter	From	To
Beacon Order	1	14
Frame Order	1	14
Enable Cap	false	true
requestGTS	0	6

Table 4.3: *Experimental Scenarios*

	Num of Nodes	Packet Rate (Pkts/Sec)	Size of the Design Space
Scenario 1	4	30, 65, 80	$7.5 * 10^6$
Scenario 2	6	15, 40, 60	$1.4 * 10^9$
Scenario 3	8	5, 15, 30	$2.9 * 10^{11}$
Scenario 4	10	5, 10, 15	$5.6 * 10^{13}$

Since the cardinality of the design space is extremely high in all the scenarios, the optimal Pareto curve cannot be extracted with an exhaustive search, thus it has been obtained by running all the exploration algorithms for 3000 iterations (solutions) for 20 iterations each. The distance between the Pareto sets have been compared using the *Average Distance from Reference Set* (ADRS) [126]. The ADRS is usually measured in terms of percentage and should be minimized.

Evaluation of the Proposed Algorithm The first set of experiments aims at evaluating the improvements achieved thanks to the tailoring of the standard MDP technique to the WSN field. Table 4.4 presents a comparison between the standard implementation of the MDP algorithm and the one proposed within this work. The experimental results shown in Table 4.4 demonstrate that the ADRS of the initial points of the proposed algorithm is considerably lower than the ones obtained with [32]. This advantage makes it possible to increase the overall performance of the algorithm, so that the ADRS of the final solutions found by our algorithm is always lower than the ones obtained with [32]. In addition to this, the proposed algorithm is able to converge with a lower number of evaluations, except for Scenarios 3 and 4, where the algorithm presented in [32] often falls in local minima (as shown by the quite high ADRS of the solutions found by the algorithm).

A critical aspect of the standard MDP algorithm is its scalability. In order to analyze this factor, uncertainties of the first kind should be simulated only if the number of states ($|NS|$) is larger than a given threshold K. To evaluate the effect of K on the exploration speed, several experiments with different values of K have been performed; Scenario 2 has been chosen as reference example for this analysis. Values of λ have been varied such as the average number of generated states for each action is between 3 (higher λ) and 30 (lower λ). Figure 4.29 illustrates the average number of parallel explorations for different values of K with respect to the average number of states for each step. The bigger is K, the higher is the amount of parallel (independent) explorations. However, the effective exploration time is not directly correlated with the amount of parallel explorations. Figure 4.30 illustrates the average amount of time (in minutes) required for the exploration with different values of K. Although the number of parallel

Table 4.4: *Comparison between original and proposed implementation of the MDP*

Scenario 1	[32]	[proposed work]
ADRS[%] init	45.5	21.7
ADRS[%] final	3.9	2.34
Eval. for Convergence	110	70
Scenario 2	[32]	[proposed work]
ADRS[%] init	44.7	21.47
ADRS[%] final	5.73	2.70
Eval. for Convergence	170	120
Scenario 3	[32]	[proposed work]
ADRS[%] init	35.93	24.3
ADRS[%] final	11.35	3.52
Eval. for Convergence	70	80
Scenario 4	[32]	[proposed work]
ADRS[%] init	38.93	27.3
ADRS[%] final	14.18	5.16
Eval. for Convergence	110	70

evaluations increases with both K and the number of generated states, the overall time behaves differently. In fact, for small values of generated states, *K=5* performs better then *K=2* even if the amount of parallel exploration is bigger. However, for high values of generated states (i.e. 30), *K=2* performs better. *K=inf* have no better performances in all the situations; it confirms that a bound on the generated states increase the exploration's speed. Moreover, exploration efficacy (ADRS and convergence speed) is not affected by K since it strictly depends on λ, thus it is suggested to tune K such that the exploration time is minimized.

The proposed algorithm is able to reduce/control the exponential growth of the number of parallel executions with respect to the original approach presented in [32] (where the threshold K is not used). Then, in addition to improving the quality of the final solution, the proposed approach is also able to reduce the number of explorations to be

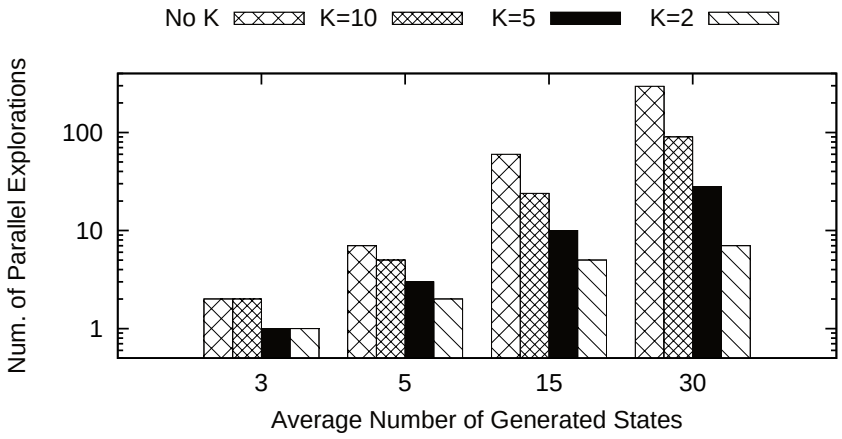


Figure 4.29: *Number of parallel explorations with different values of K (**K=inf.** refers to the original algorithm). The average number of generated states depends on the chosen λ and movement vectors' accuracy*

performed, thus reducing the computational costs and time required by the algorithm itself.

ADRS and Number of Evaluations The second set of experiments compares the proposed MDP with three state-of-the-art multi-objective optimization algorithms: controlled non-dominated sorting genetic algorithm (NSGA-II), Pareto memetic algorithm (PMA) and multiple objective simulated annealing (MOSA). The MOMHLib++ [6] library was used as a reference implementation of these algorithms. In order to ensure a fair comparison, all the algorithms exploits the same technique to generate the initial points.

Each configuration of all the optimization algorithms has been executed for 20 times in the four scenarios and the average ADRS [124] has been evaluated. The ADRS is computed every 5 evaluations in order to understand its trend with respect to the number of evaluations. The results of these experiments are presented in Figure 4.31(a) (Scenario 1), Figure 4.31(b) (Scenario 2), Figure 4.31(c) (Scenario 3) and Figure 4.31(d) (Scenario 4), which show that MDP is able to reach a low ADRS (below 5%) using less than 40 evaluations, while the other algorithms require from 100 to almost 300 evaluations to reach the same objective. This is a reduction of 60-87% in the number of required simulations.

As design space cardinality increases, the identification of the Pareto curve is more difficult, thus exploration efficiency decreases. It is interesting to notice that the reduction of effectiveness of MDP is considerably lower than the other algorithms (See Table 4.5). These results encourage the use of such algorithm on large design spaces.

During the Design Space Exploration, optimization algorithms should be run several times in order to guarantee the quality of the identified Pareto curve. By analyzing the standard deviation of the ADRS on the independent runs, it has been observed that standard deviation on MDP is the lowest. Table 4.6 summarized the computed standard deviations. A low standard deviation implies that the optimization algorithm requires few repeats to guarantee the quality; it further reduces the overall time required for Design Space Exploration.

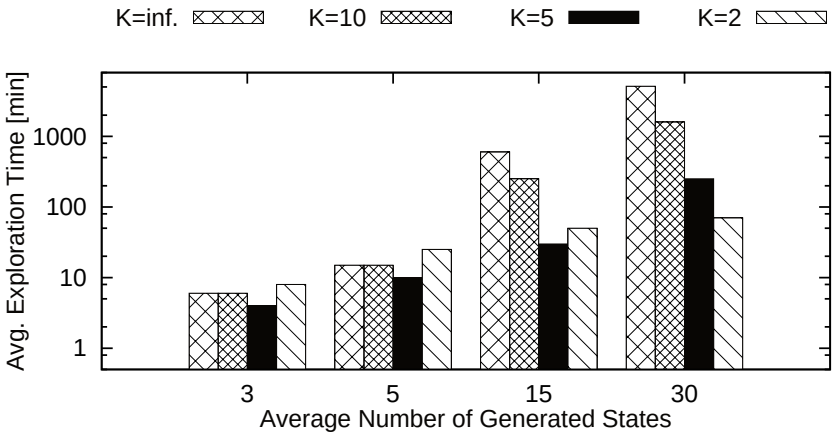
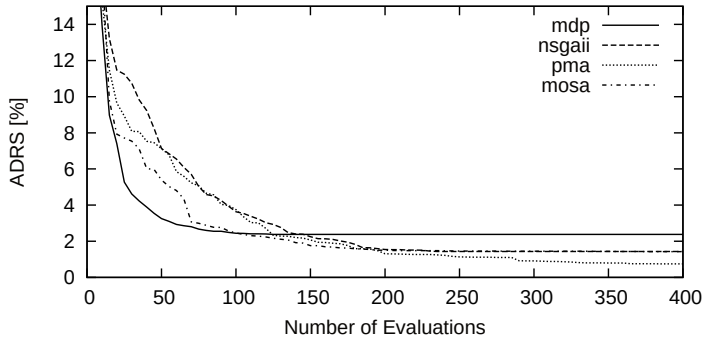
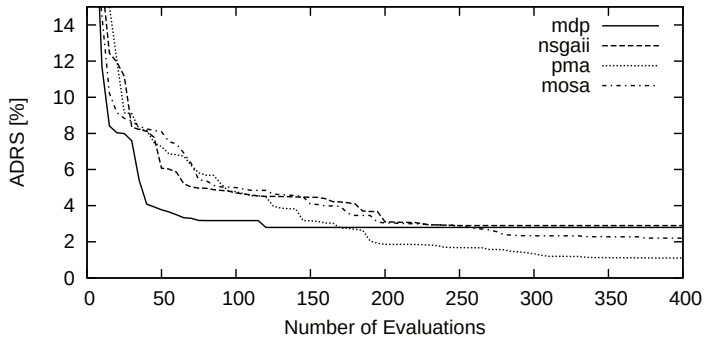


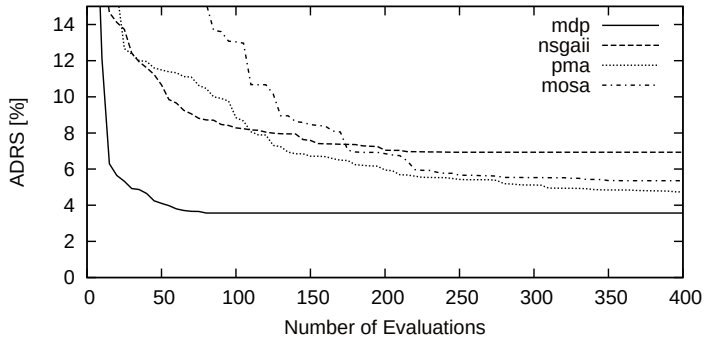
Figure 4.30: *Amount of time (in minutes) required for the exploration with different values of K (**K=inf.** refers to the original algorithm). The average number of generated states depends on the chosen λ and movement vectors' accuracy*



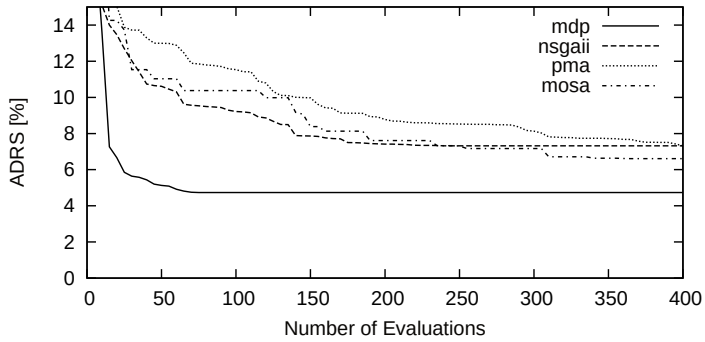
(a) ADRS per Number of Evaluations in **Scenario 1**



(b) ADRS per Number of Evaluations in **Scenario 2**



(c) ADRS per Number of Evaluations in **Scenario 3**



(d) ADRS per Number of Evaluations in **Scenario 4**

Figure 4.31: *ADRS per Number of Evaluations*

Table 4.5: *Final ADRS of search Algorithms*

	MDP	NSGA-II	PMA	MOSA
Scenario 1	2.34	1.42	0.83	1.43
Scenario 2	2.70	2.88	0.78	2.20
Scenario 3	3.52	6.52	4.76	5.36
Scenario 4	5.16	9.55	8.78	7.32

Table 4.6: *Standard Deviation of ADRS*

	MDP	NSGA-II	PMA	MOSA
Scenario 1	0.34	1.20	1.01	1.34
Scenario 2	0.55	2.56	1.94	1.91
Scenario 3	0.64	2.44	2.38	2.81
Scenario 4	0.77	2.59	2.96	2.92

Summary A technique to reduce the amount of simulations necessary to obtain the Pareto set of the design space exploration of Wireless Sensor Networks has been presented in this Section. The proposed technique uses models as soon as they provide an acceptable accuracy and simulates only when it is needed. The knowledge domain about slotted IEEE 802.15.4 has been extracted from the models of the state of the art. Experimental results have shown that proposed algorithm significantly improves the efficiency and scalability with respect to the classical algorithm. To confirm the effectiveness of the technique, the proposed approach has been compared with semi-random algorithms such as NSGA-II, PMA and MOSA. Experimental results have shown that MDP reduces the number of simulations required to converge (ADRS lower than 5%) from 60 to 87%. This reduction is more relevant as the cardinality of the design space increases, making it an effective approach for the design space exploration of WSNs.

4.5 Software Definition

A detailed discussion on the best technique to manage and develop software in WSNs is outside the purpose of this thesis, and a good survey on the topic can be found in [115]. This Section discusses the details of the *SW Partition and Mapping* process, introduced at the beginning of this Chapter.

4.5.1 Partition and Mapping

One of the main problems during the development of complex distributed software applications in WSNs is the **partition** and **mapping**. *Partitioning* is the action of dividing a whole piece of software into smaller tasks, entirely executable on single nodes. *Mapping* is the process of assigning these tasks to specific nodes of the network. The granularity of the partitions can vary according to application needs; the smaller are the tasks, the higher will be the communication overhead. On the other hand, big tasks make the mapping less effective since the mapping possibilities are lower, but it is faster to do.

This Section presents an evaluation of various meta-heuristics applied to the problem of partition and mapping of software in WSN.

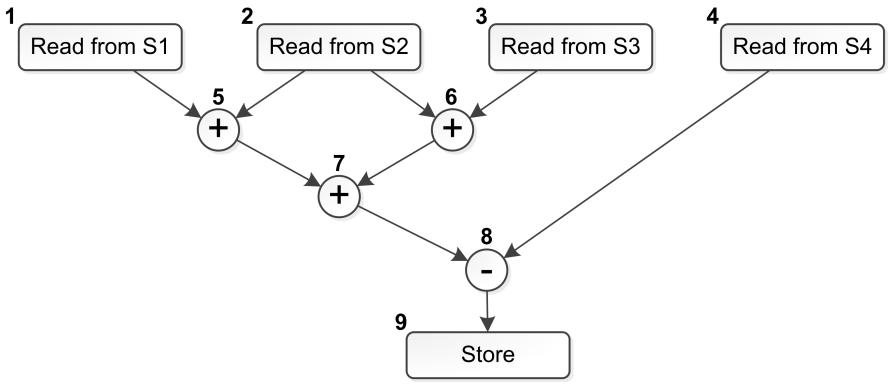


Figure 4.32: *An example of Data Flow Graph*

PROBLEM DEFINITION

A software application can be described using a DFG (Data Flow Graph) [77]. A DFG represents true data relationships (Read After Write) among different tasks, and give an application-independent representation of the software. This generality is helpful to develop application-independent algorithms for software optimization; in other terms, the algorithm does not care why a specific data are required by another task, it just knows that these data are required.

A DFG is an acyclic graph representation of software, where nodes are tasks and arcs indicate dependencies. Nodes without incoming arcs are called **initiator** nodes and nodes with no output arcs are called **terminating** nodes. In a WSN, **initiator** nodes are always a **read-from-sensor** operation since it usually does not require any information from other nodes, while **terminating** nodes are typically **store** tasks (like *store to database*) or **send-to-actuator** operations.

An short example of DFG is shown in Figure 4.32. Tasks 1, 2, 3 and 4 are *read-from-sensor* tasks, 5, 6, 7 and 8 are operational tasks and 9 is a *store* task. In this example, operational tasks are very simple and perform a binary operation only but, generally, multi-instruction tasks can be used instead. For the sake of simplicity, this work does not consider conditional or iteration statements.

TASK ALLOCATION PROBLEM

The task allocation is the problem of assigning tasks to a specific node of the network. The problem takes the network topology (N) and the DFG (D), and gives the task allocation (T) as output, that is a set of tuples describing the allocation of tasks on the nodes.

$$(n, d) \in T$$

where $n \in N$ is a node of the network and $d \in D$ is a task of the DFG. The purpose of this section is to present an automatic technique to identify T such that the execution time is minimized.

The problem requires that all the tasks are assigned to a single node. It can be described with a fixed array:

d_0	d_1	d_2	...	d_k
n_0	n_1	n_2	...	n_k

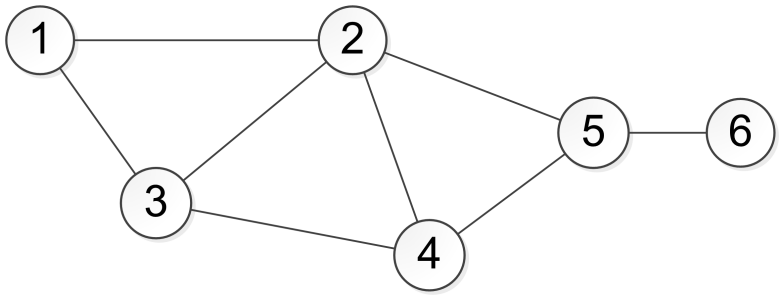


Figure 4.33: *An example of Network*

such that $d_0, d_1, ..., d_k \in D$ and $n_0, n_1, ..., n_k \in N$. Given the network in Figure 4.33 and the DFG in Figure 4.32, and assuming the following constraint:

D	1	2	3	4	9
N	2	3	1	2	6

which means that task 1 must be executed on node 2, task 2 on node 3, task 3 on node 1, etc., a feasible solution to this problem is:

D	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
N	*2*	*3*	*1*	*2*	**2**	**4**	**3**	**4**	*6*

Next paragraph presents some experimental results conducted on the given problem by applying four Genetic Algorithm based heuristics. The heuristics differ in only in the selection of best candidates, not in the crossover and mutation operations.

Crossover consists in a random mix between the two parent configurations. An example of crossover for the previous example is:

Parent 1:

D	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
N	*2*	*3*	*1*	*2*	**2**	**1**	**5**	**7**	*6*

Parent 2:

D	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
N	*2*	*3*	*1*	*2*	**5**	**2**	**1**	**1**	*6*

Child:

D	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
N	*2*	*3*	*1*	*2*	**2**	**2**	**1**	**7**	*6*

Mutation is a random change of a parameter with a feasible value. And an example of mutation for the previous example is:

Original Configuration:

D	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
N	*2*	*3*	*1*	*2*	**5**	**2**	**1**	**1**	*6*

Mutated Configuration:

D	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
N	*2*	*3*	*1*	*2*	**5**	**2**	**1**	**3**	*6*

	Network	DFG		
	# Nodes	# Tasks	# Arcs	# Min Sensing Tasks
Scenario 1	25	45	75	6
Scenario 2	35	55	95	6
Scenario 3	55	75	125	6

Table 4.7: *Scenario Summary*

EXPERIMENTAL RESULTS

These experiments have been conducted in order to understand the effectiveness and efficiency of the heuristics in solving this problem. The solutions have been validated with OMNET++ Castalia framework [9], modified to execute custom DFG. The task engine takes the DFG as input and simulates the time spent in executing the task; no semantic information has been introduced into the simulator. In these experiments tasks require from 1 to 4 seconds to be executed; the execution time of each single task is randomly generated and included into the DFG as information during the initialization phase. Simulations have been conducted with S-MAC (as MAC layer) and AODV (as Routing Protocol).

The scenarios differ in the number of nodes deployed in the network and the size of the generated DFG. Both the network layout and the DFG have been generated randomly. In these experiments, four seeds have been used for the generation of the network and the DFG; algorithms random generations have random seeds at each iteration. Each configuration (algorithm+scenario+seed) has been executed 20 times in order to mitigate the randomization effects. Table 4.7 illustrates the parameters used to generate the network and the DFG. The initialization algorithm creates a network with exactly *"# Nodes"* nodes with random positions. The algorithm guarantees that all the nodes are connected, thus there exists at least one path from one node to all the others. Then it generates the random DFG. The DFG is composed of *"# Tasks"* number of tasks, "*# Arcs*" number of arcs (representing true dependencies) and at least *"# Min Sensing Tasks"* sensing tasks. A sensing task is a task that does not require any data to operate, so in a DFG a sensing task is a task without incoming arcs.

The problem has been solved with four different heuristics and the random algorithm. All the heuristics used for this problems belong to the Genetic Algorithm (GA) family: **GA+Tournament Selection**, **SPEA2**, **NSGA-II** and **GA+Roulette**. The metric to optimize is the DFG **Execution Time**, defined as the time required to execute all the tasks in the DFG. As aforementioned, position of sensing tasks is fixed (defined by the *Sensing Coverage* process), thus the algorithms aim at identifying the position of other tasks. **Execution Time** takes into consideration the execution time of each task on the node (scheduled with the FCFS [First Come First Served] policy) and the communication overhead, introduced by S-MAC and AODV.

Figures 4.34, 4.35 and 4.36 show the experimental results obtained in these three scenarios respectively. In all the experiments, **GA+Tournament** outperforms all the other heuristics, by reducing the Execution Time from 20 to 50% with respect to the other algorithms. Performance of the other heuristics are quite aligned giving an improvement of about 20% with respect to the initial solutions (Random), except **GA+Roulette** whose average performance is the worst.

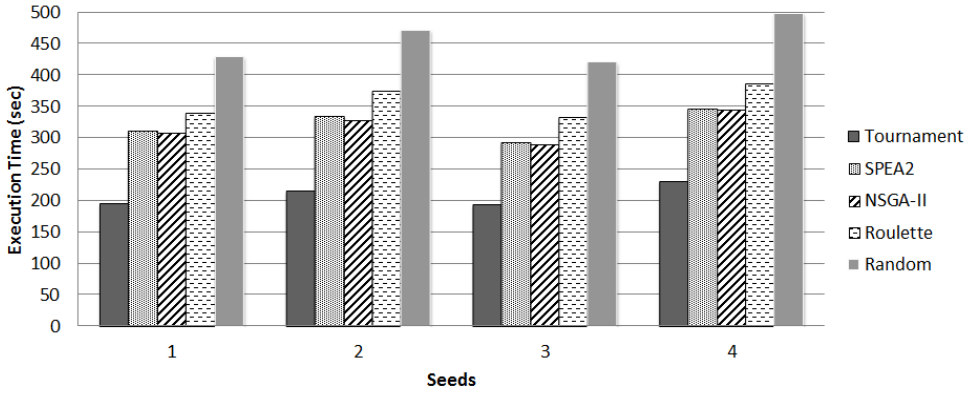


Figure 4.34: *First Scenario*

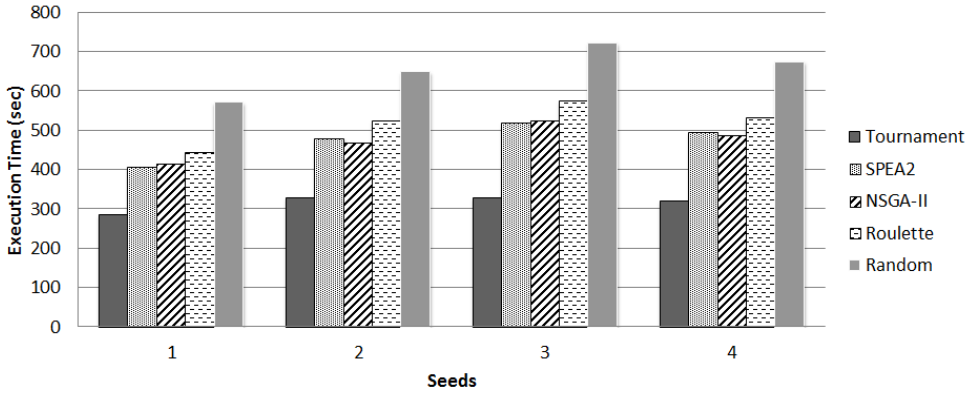


Figure 4.35: *Second Scenario*

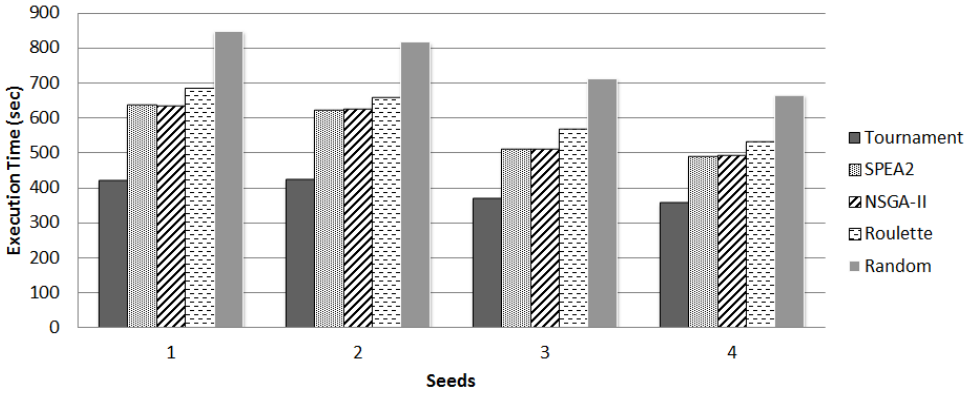


Figure 4.36: *Third Scenario*

4.6 Chapter Conclusions

This Chapter presented the proposed design flow, introduced to clearly define the design steps of WSNs. Complexity of WSNs is growing, due to the complexity of devices

and the increasing number of design alternatives in network protocols. The processes composing the design flow cover all the aspects of a WSN design and are generic in order to cover a large number of WSN applications.

The placement problem is a well researched problem in literature and, although placement is still an open issue in WSN design, no innovative techniques have been presented here, rather, a selected list of State of the Art algorithm has been overviewed. Hardware and Network Configuration represents the main topic of this thesis. Three techniques have been presented: *Simulation-based Design Space* Exploration, *Model-based Design Space Exploration* and *Hybrid Design Space Exploration*. The last aspect of the design is the software development. This chapter introduces and defines the *Task Allocation* problem. Four heuristics have been tested and results have been presented.

5

Online Adaptivity in Wireless Sensor Networks

A good design phase is important to have optimized networks whose efficiency is verified and tested. However, the environment where a Wireless Sensor Networks operate changes constantly, and these networks may be often affected by many runtime issues that potentially reduce the effectiveness or the lifetime of the network. Interferences, nodes failures or node mobility can dramatically affect how the network works. An interference, for instance, may cause several retransmissions, that have a strong impact on the power consumption and, subsequently, on the lifetime of the network. The need to develop online algorithms to adapt the network at runtime is of extreme relevance to ensure the operational effectiveness and performance of the network over the time.

Adaptivity can be used not only to mitigate the effect of faults or interferences, but to tune the system to operate at its best under various conditions. Run time dynamics can be used to optimize the network behavior efficiently. Spatial correlation is a unique and very interesting characteristics of Wireless Sensor Networks [158]. It can be used to control the amount of transmissions required to build a sensor map of the environment.

Since adaptivity is a specific-problem, this Chapter will not present general-purpose adaptive techniques, rather, it will presents two specific approaches that deal with online adaptivity of Wireless Sensor Networks: a technique to reduce BAN-BAN interferences in WSNs, and a technique to reduce the amount of required transmissions in a cluster-based network.

5.1 B^{2}IRS: a Technique to Reduce BAN-BAN Interferences in Wireless Sensor Networks

When the transmission range is limited to few meters, these systems are known as Body Area Networks (BANs). BANs can be used to monitor different physiological and biological parameters like temperature, Electrocardiogram, Electroencephalograph, and so on. Monitoring important medical information is useful for, i.e., elderly support [44] or to prevent and treat dangerous diseases like epilepsy [101].

The design of efficient BANs is a complex task, characterized by tradeoffs among energy, costs, network bandwidth, memory and computational resources [60]. Unfortunately, runtime behaviors, like node failure and network interferences, could dramatically affect efficiency and reliability of BANs, and a good offline design may not be

sufficient. In particular, BAN-BAN interferences cause dramatic performance and efficiency degradation in BANs [50]. BAN-BAN interferences occur when two or more co-located BANs (located in the transmission range of other BANs) operate on similar frequencies. BAN-BAN interferences are very dangerous since different independent systems communicate on a shared communication medium.

IEEE 802.15.4 [18] is a standard that has been introduced to satisfy energy requirement of emerging devices, and it is one of the most widely used technologies for BAN applications [46]. Operational frequencies used in the IEEE 802.15.4 overlap frequencies of existing wireless technologies like IEEE 802.11, which overcomes IEEE 802.15.4 in terms of transmission power, affecting BAN application's performance [143]. When IEEE 802.15.4 devices coexist with WiFi or Bluetooth technologies, the amount of interference-free channels is highly reduced, and classical BAN-BAN interference reduction techniques, like channel switching, cannot be applied anymore. BAN-BAN interferences can be categorized as:

- **Cross-Channel Interference**: interferences between networks operating on different channels;
- **Single-Channel Interference**: interferences between networks operating on the same channel.

This Section shows that, due to single channel interference, packet receive ratio can be reduced to less than 60% with only 2 BANs and 25% with 4 BANs. In critical applications (like medical applications), such reduction cannot be tolerated. Considering the amount of interference-free channels of IEEE 802.15.4 and its wide diffusion for BANs, an efficient technique to cope with this problem is required. A BAN-BAN Interference Reduction (B^2IR) technique which reschedules beacon packets in order to avoid the overlapping of active periods among different BANs is presented here. Experimental results show that the proposed approach considerably improves the performance of BSNs affected by interference issues.

Effects of Conflicting BANs The proposed analysis is tailored to BAN-BAN interference on slotted, beacon-enabled IEEE 802.15.4/ZigBee networks only. ZigBee is strongly based on IEEE 802.15.4 standard [18] and extends some features to provide a better energy efficiency. More details on the IEEE 802.15.4/ZigBee protocol are available in Section 2.6 of this thesis. A summary picture is depicted in Figure 5.1

Two BANs are **conflicting** if they are co-located and operate on the same frequency (single channel interference). In ZigBee networks, synchronism between end devices is ensured by the coordinator, which does not ensure synchronization with other coordinators, thus CAP/CFP periods can be overlapped: in this case, even though CSMA-CA is still able to work thanks to carrier sensing, this is not true for GTS transmissions, which do not sense the wireless channel before transmissions. In this scenario, if GTS transmissions are overlapped, they can be the cause of interferences. When an interference occurs, according to the standard, the packet can be retransmitted a certain amount of time before being considered lost. Retransmissions increase both the throughput requirement and the energy consumption.

To quantify the effect of single channel interference, extensive simulations varying the number of conflicting BAN and their throughput have been performed. These simu-

lations have been conducted with OMNET++ Castalia simulator [9]. In this experiment, each BAN is composed of three devices (one coordinator and two end devices), and the MAC layer has been set to have $SO = 4$, $BO = 6$, $GTS_num = 4$, $TX_{out} = -15dBm$, distance between end devices and coordinator was $1mt$, the average distance between coordinators was about $3mt$ and Packet size was $115Bytes$. The amount of packets per second sent to the coordinator varies from 5 to 60. As shown in Figure 5.2, higher packet rates imply more packets lost and more collisions. In a superframe, packets can be lost in two ways: with **buffer overflows** and with **collisions**. The first occurs when CSMA-CA delay the transmissions due to non-free channel, while the latter when GTS transmissions collide with other transmissions. It has been noted that CSMA-CA is responsible of buffer overflow since delaying transmissions fill the buffer, and GTS is responsible of packet collisions since the transmissions in the GTS slot occur without carrier sensing. According to this analysis, CSMA-CA helps to keep collisions as low as possible, but fills the buffer, while, on the other side, GTS does not fill the buffer but does not avoid collisions.

State of the Art The interference issues for the IEEE 802.15.4 technology have been widely studied in past years [143] [117] [136] with particular attention to the integration with existing wireless technologies working on similar frequencies (i.e. IEEE 802.11, Bluetooth). Pollin et al. [136] propose a distributed adaptation technique based on scanning and increased cognition to reduce the effect of IEEE 802.11 on IEEE 802.15.4 nodes. Shin et al. [143] evaluate the performance of IEEE 802.15.4 and IEEE 802.11b operating at 2.4 GHz, showing that the distance and the center frequency offset between the two technologies affect the Packet Error Rate (PER). An effective technique has been presented in [117], where the authors show that, by selecting the right channel, it is possible to reduce the end-to-end loss rate from 22%-58% to less than 1%.

BAN-BAN interferences among different 802.15.4-based systems have been analyzed and various solutions have been presented. De Silva et alt. [50] conducted a preliminary investigation of the BAN-BAN interference effect. According to their measurements, in presence of 5 or more high-rate BSNs in the same environment, the Packet Data Rate (PDR) can fall as low as 65%. They propose a fixed WSN infrastruc-

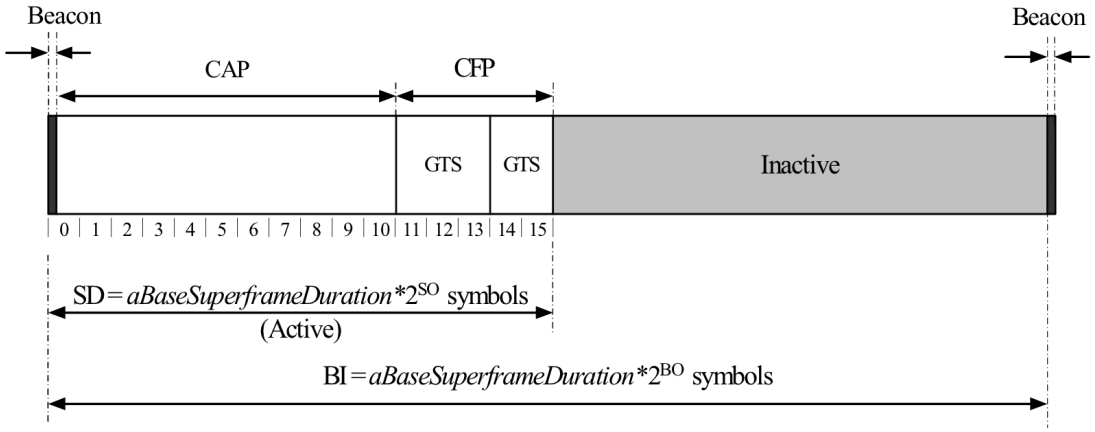


Figure 5.1: *An example of the superframe structure (from [18])*

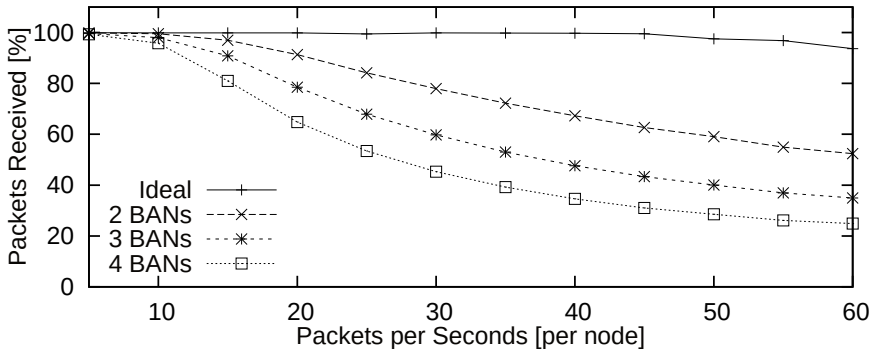


Figure 5.2: *Amount of packets received varying the throughput of the network with 2,3 and 4 conflicting BANs compared to the throughput of a single BAN without conflicts. Throughput is expressed in packets per seconds; each packet is 105 Byte*

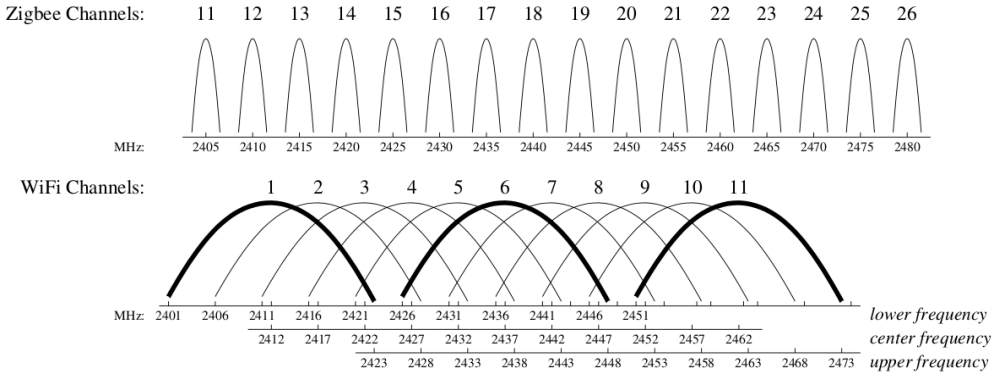


Figure 5.3: *Frequency ranges used by the ZigBee and WiFi Channels (from [117])*

ture to identify when BSNs are interfering with each other and act to make the BSNs communicate on different frequency channels. Unfortunately, a fixed network can be very expensive and difficult to deploy. To overcome this strong limitation, a decentralized suppression technique have been proposed by Guowei et alt. in [161]. They use a non-cooperative game based on a no regret learning algorithm to identify free channels, reducing the interferences between BSNs. Coexistence issues of multiple co-located networks running on adjacent radio channels has been explored by Lo Bello and Toscano in [100]. In that paper they present a testbed to evaluate cross-channel interferences and they identify that this phenomenon is negligible thanks to the distance between channels of 802.15.4 networks (see figure 5.3). A different approach to reduce BAN-BAN interferences has been proposed by Khan et al. in [78]. They propose an algorithm for the interference rejection of nearby BSNs; it performs better than common techniques like Optimum Combining (OC) and Weiner-Hopf (WH).

All these approaches try to solve the BAN-BAN interferences issues by switching to a free channel. Channel switching is an effective solution, when applicable, to reduce BAN-BAN interferences, since the distance between adjacent channels in IEEE 802.15.4 does not introduces considerable interferences [100]. However, ignoring single-channel interferences can drastically decrease the performance of a BAN. Thus, considering the effect of single-channel interferences on various co-located BANs,

arises that addressing this problem is extremely important.

The analysis of channel switching techniques shows that, although they are usually very effective, they require a number of interference-free channels (C) bigger than the amount of co-located BANs (N), in order to assign a unique channel to every BAN. At $2.4GHz$, IEEE 802.15.4 has sixteen, non-overlapped channels, so switching is particularly efficient with this technology and can be scaled up to sixteen coexistent BANs but, unfortunately, other technologies, like IEEE 802.11 (WiFi) or Bluetooth, reduce the amount of available, interference-free channels. When two BANs operate at the same frequency, if not correctly synchronized, they will interfere with each other, causing a strong reduction of performances [50]. Figure 5.3 illustrates how IEEE 802.11 WiFi channels overlaps IEEE 802.15.4 ZigBee channels.

With three active WiFi networks on three distinct channels like in Figure 5.3, only 4 ZigBee channels are interference free (15, 20, 25 and 26). In this situation, using channel switching, only 4 co-located BANs can work without interfering with each other. Assuming a transmission range of few meters for BAN's devices and a small area like a train coach, a theater or an office, it is realistic to consider a scenario in which more than 4 co-located BANs have to operate. Thus, in order to cover a variety of situations, the approach has been validated with 2, 3 and 4 conflicting BANs.

Proposed Methodology The proposed methodology is a **BAN-BAN Interference Reduction System** (**B^{2}IRS**) that provides an effective way to reduce, and possibly eliminate, single channel BAN-BAN interference issues. B^{2}IRS is compatible with all the state-of-the-art approaches and is intended to be used on slotted, beacon-enabled, ZigBee networks when they operate in a context where there are no free channels (thus, channel switching can not be performed).

Overlapping occurs when a coordinator sends a beacon packet in the active period of another BAN. According to the standard, end devices initiate the transmission only after the reception of a beacon packet from their coordinator, following the structure in Figure 5.1. Usually, a coordinator ignores beacons received from the coordinators of another network causing CAP/CFP overlapping. To better understand this concept, an example with three co-located networks operating on the same channel is presented in Figure 5.4. In this situation, during the active period of N_0, both N_1 and N_2 start their active periods, causing a CAP/CFP overlapping between the networks. Since the networks have the same *Beacon Order*, CAP/CFP overlapping continuously occur.

The Proposed Idea To avoid overlapping, it is important to ensure that a BAN will never send a beacon in the active portion of another BAN. In order to do that, B^{2}IRS reschedules the beacon transmissions at the end of the active period of another BAN. To identify when beacons must be transmitted, it is important to gather all the beacon transmissions from co-located BANs since, according to the standard [18], a beacon includes the information about the structure of the superframe. In fact, when a coordinator receives a beacon from another coordinator, it determines if its next beacon will fall into an active period of that coordinator. To determine if the beacon must be rescheduled, it compares the time when the next beacon will be sent with the end of the active period of the other coordinator, given by 2^{SO}:

$$t_B < NSS * BSD * 2^{SO} \tag{5.1}$$

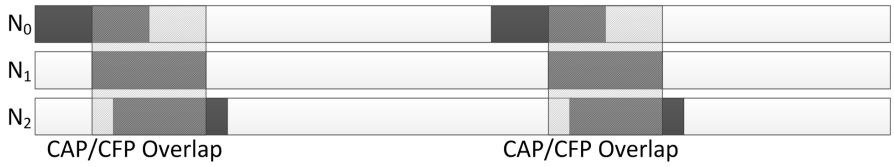


Figure 5.4: *This picture illustrates three conflicting BANs **without B^{2}IRS**. Dark areas represent active period and light areas inactive periods. With no B^{2}IRS, CAP and CFP will overlap continuously*

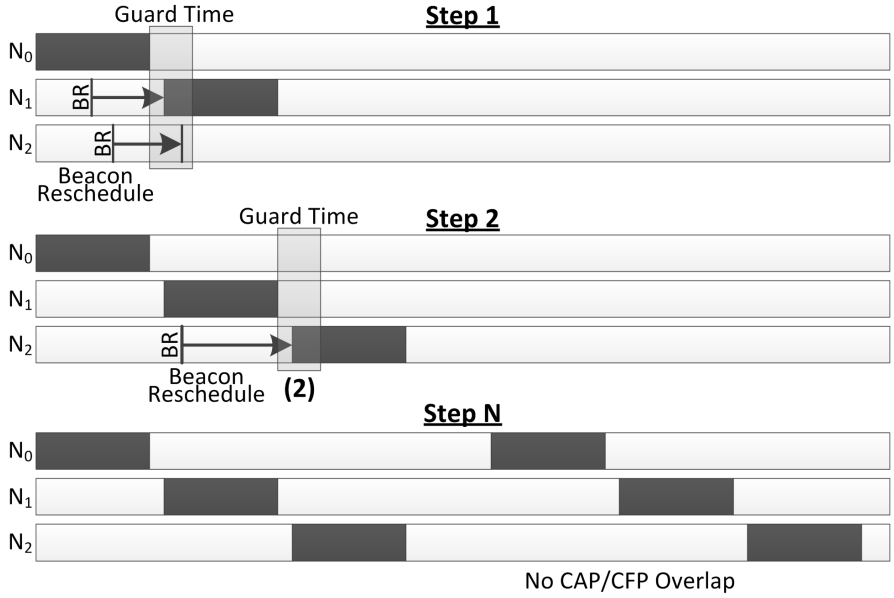


Figure 5.5: *This picture illustrates three conflicting BANs **with B^{2}IRS**. Dark areas represent active period and light areas inactive periods. B^{2}IRS reschedules the beacon twice: first for network N_1 (Step 1) then for N_2 (Step 2). At this point, CAP and CFP will not overlap anymore (Step N)*

where t_B is the time (in symbols) when the next beacon is scheduled, NSS is the *number of superframe slots* (16), BSD is the *base slot duration* (in symbols) and SO is the superframe order (contained into the beacon packet).

If Equation 5.1 is satisfied, the beacon must be rescheduled at the end of the active period, given by $NSS * BSD * 2^{SO}$. To avoid collisions with other BANs, that could reschedule its beacon at the end of the same active period, CSMA-CA is implemented. It means that the coordinator does not reschedule the beacons exactly at the end of the active period, but in a random instant in an interval of seconds (**Guard Time**) after the active period. Information about carrier sensing is retrieved from the radio and only if the channel is free, the beacon is transmitted. Otherwise, if the coordinator receives a beacon from another coordinator, the beacon's transmission is rescheduled again; empirical analysis show that if guard time and carrier sensing is not used, overall performances of B^{2}IRS can be seriously affected.

An example of the proposed idea is presented in Figure 5.5. When B^{2}IRS is not used, CAP/CFP overlap is not eliminated. On the other hand, B^{2}IRS reschedules beacon packets to avoid CAP/CFP overlapping. In the example, both N_1 and N_2 reschedule their beacons because both fall into the active period of N_0. Rescheduling occurs in the interval defined by the *Guard Time*. After the active period of N_0, N_1 transmits

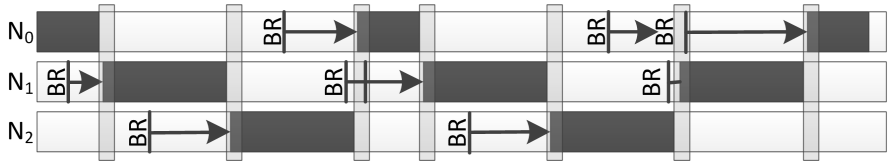


Figure 5.6: *This picture illustrates three conflicting BANs **with B²IRS** when Equation 5.2 is **satisfied**. Dark grey areas represent active period and light gray areas inactive periods, BR indicates that the current beacon is rescheduled and the arrow shows when it is rescheduled*

its beacon and, since N_2 beacon falls into the active period of N_1 it re-reschedules its beacon. After active period of N_1, N_2 can finally send a beacon and start its active period. From now on, CAP/CFP overlap does not occur anymore.

IEEE 802.15.4 defines the beacon interval as function of 2^{BO} and active periods as a function of 2^{FO}. Thanks to this definition, superframe duration and beacon interval cannot be prime to each other and, after few beacon rescheduling operations, CAP/CFP will not overlap anymore, ensuring maximum performances and reliability for the involved BANs. The only limitation of this approach is when an inactive period is not long enough to ensure beacon rescheduling. It happens when the sum of active periods of all the conflicting BANs is greater than the minimum inactive periods of all the BANs:

$$\sum_{i=0}^{N-1} (SD_i + GT) > \min_{i \in N} (BI_i - SD_i) \qquad (5.2)$$

where SD_i and BI_i represent the superframe duration and the beacon interval of node i respectively and GT is the Guard Time. Refer to [18] for details on how SD and BI are computed. According to Equation 5.2, the amount of BANs that do not cause overlapping depends on the structure of the superframe. Generally, the bigger the active periods are, the easier will be the rescheduling. In case Equation 5.2 is satisfied, B²IRS continually reschedules beacons. An example of how B²IRS works when Equation 5.2 is satisfied, is presented in Figure 5.6.

The Proposed Algorithm Algorithm 1 presents how B²IRS works. It must be included into the routine that handles incoming packets.

Algorithm 6:

1]Pseudo-code of B²IRS algorithm. [Coordinator]

```
1: P ← packet_received()
2: if P is BEACON then
3:    SO ← getSO(P)
4:    t_B ← getTimer(NEXT_BEACON)
5:    if t_B < NSS * BSD * 2^SO then
6:       t_B ← NSS * BSD * 2^SO + rand[0, GT]
7:       setTimer(t_B,NEXT_BEACON)
8:    end if
9: end if
```

When a beacon is received, SO is extracted from it. Then, Equation 5.1 is evaluated. If it is satisfied,the beacon is rescheduled according to the policies presented above,

otherwise the beacon is ignored.

Experimental Results To prove the effectiveness of the proposed approach, two different experiments has been conducted: the first where Equation 5.2 is not satisfied and the second where it is satisfied. These experiments show that, even if overlapping cannot be avoided, B^{2}IRS is able to substantially keep the amount of received packets close to the one of an ideal situation. In all these experiments, a BAN is composed of one coordinator and two end devices, with average distance of 1 mt (the distance between coordinators is 3 mt). The **ideal** solution has been calculated with a single, non-conflicting BAN. The simulation time for each test has been set to 300s and each value has been computed averaging 50 simulations. All the experiments were conducted using the OMNET++ Castalia simulator [9], that provides accurate wireless channel and radio models, with a realistic node behavior [38]. The IEEE 802.15.4 MAC layer implemented in Castalia has been extended to effectively support the proposed approach.

The first experiment has been performed on 2, 3 and 4 conflicting BANs, with and without B^{2}IRS, and varying the amount of packets per seconds. In this experiment, Equation 5.2 is not satisfied, and each BAN is equally configured: $SO = 4$, $BO = 6$, $GTS\_num = 4$, $TX_{out} = -15dBm$, $NSS = 16$, $BSD = 60$, $GuardTime = 7ms$ and packet size is 115 Bytes; to keep Equation 5.2 not satisfied, with 4 BANs, one network is configured with $SO = 3$. Figure 5.7 presents the results of the first set of experiments. Both the percentage of packets received (top plot) and the number of buffer overflows (middle plot) are very close to the ideal, while the number of interferences is slightly higher, but this is negligible because it does not affect the performance of the network. Energy consumption increases from 7% (2 BANs, 5 pkt/s) to 32% (4 BANs, 60 pkt/s) if B^{2}IRS is not used, and only 1.6% in the worst case (4 BANs, 60 pkt/s) when B^{2}IRS is employed. Thus, when Equation 5.2 is not satisfied, B^{2}IRS is able to eliminate BAN-BAN interference issues keeping the energy consumption very close to the ideal situation.

The second experiment has been performed on 3 and 4 conflicting BANs. BANs are configured as in the previous experiment, except for SO that is equal to 5 in two BANs and 4 in the other BANs; with this configuration, Equation 5.2 is satisfied. As previously explained, in this case it is not possible to ensure a perfect overlapping avoidance, and beacons are continuously rescheduled. The results of these experiments are depicted in Figure 5.8. Differently from the previous experiment, the number of buffer overflows is bigger, since it is not possible to avoid overlapping, which causes a reduction in the percentage of received packets. However, the percentage of packets received is significantly higher when B^{2}IRS is employed. Energy consumption is increased, with respect to the previous case, from 1.6% to 5.3% in the worst case (4 BANs, 60 pkt/s). An increase of 5.3% in energy consumption can be accepted, considering the noticeable improvement in terms of received packets, which directly affects the performance and the reliability of the system.

Summary This Section analyzes the effect of interferences of co-located BANs running on the same channel. Considering the amount of channels available after WiFi/Bluetooth interference reduction, channel switching can be difficult or even impossible to perform. To solve this issue, B^{2}IRS has been presented, which aims at reducing single-

channel interferences of multiple co-located networks, rescheduling beacon transmission. Experimental results show that B^{2}IRS effectively reduces the packet losses due to BAN-BAN interferences with a negligible increment in energy consumption in presence of conflicting BANs (lower than 5.3%), while classical ZigBee shows an energy consumption increment from 7 to 32%. Concluding, B^{2}IRS always improves the performance of conflicting BANs and, if no conflicts are present, it performs exactly like the IEEE 802.15.4 with a negligible overhead.

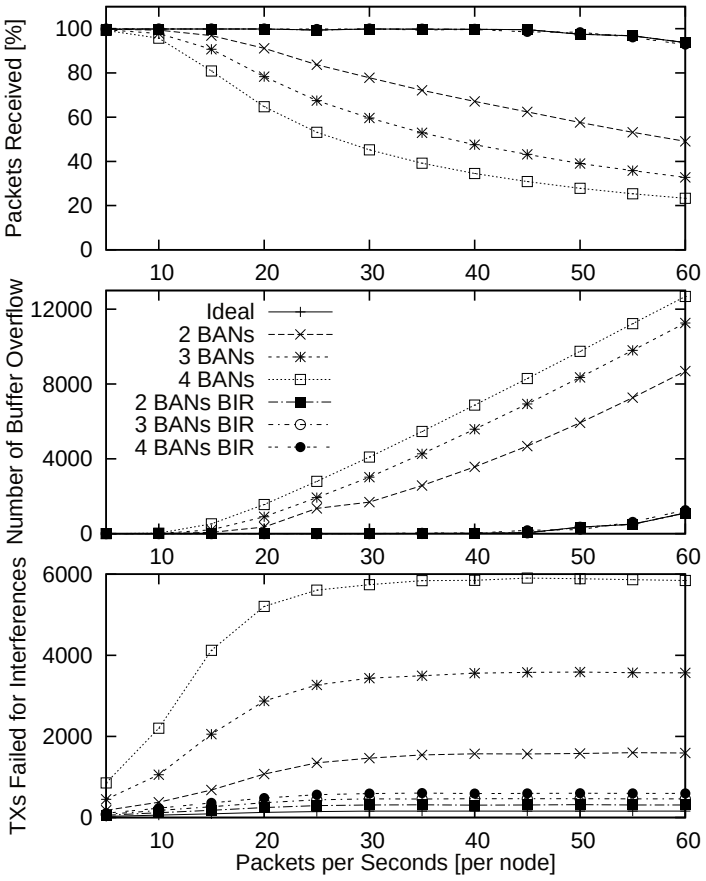


Figure 5.7: *Amount of packets received (**top** plot), buffer overflows (**middle** plot) and interferences (**bottom** plot) when **Equation 5.2 is not satisfied**. Experiments were conducted on 2, 3 and 4 conflicting BANs*

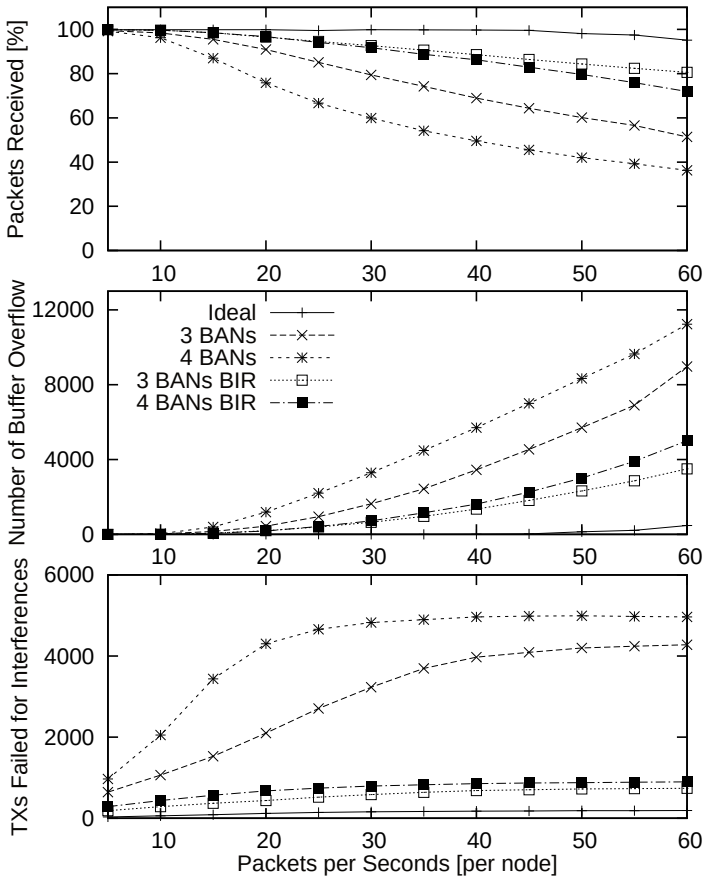


Figure 5.8: *Amount of packets received (**top** plot), buffer overflows (**middle** plot) and interferences (**bottom** plot) when **Equation 5.2 is satisfied**. Experiments were conducted on 3 and 4 conflicting BANs*

5.2 Tacit Consent: A Technique to Reduce Redundant Transmissions from Spatially Correlated Nodes in Wireless Sensor Networks

The nodes of a WSN can be densely deployed to satisfy fault-tolerance and coverage requirements. One of the most common techniques to organize these complex WSNs is *clustering*, where nodes are grouped into clusters that are managed by an elected node called *Cluster Head* (CH), which collects and aggregates data coming from the other cluster nodes. This technique is known for reducing both collisions and contention over the wireless channel, as transmissions from sensing nodes to the CHs, and from the CHs to the sink, can be locally scheduled to avoid packet collisions, hence sleeping periods are controlled to reduce energy consumption. The majority of cluster protocols is composed of two phases [19]: *intra-cluster collection*, *inter-cluster routing*. In the first phase, each member node sends its measurements to the CH, which collects and aggregates them. In the second phase, CHs route data through the network toward the sink.

Although clustering can improve the lifetime and the capacity of dense networks, performance are still constrained by the massive transmission of redundant data, which often appear in spatially-proximal observations [158]. However, it is well known that

highly-correlated streams of data [48] can be efficiently compressed according to the Slepian-Wolf theorem [49], thus improving the overall performance of the system. This redundancy can be effectively exploited especially in MAC and routing protocols to either reduce energy consumption [158] or improve network reliability [85]. In particular, the main objective of spatial-correlation-aware algorithms is to reduce energy consumption while maintaining a very good accuracy of the measurements [157].

This Section introduces a technique to exploit spatial correlation in an energy-efficient way. In particular, the clustered nodes are divided into two distinct groups: *representative* and *member* nodes. Representative nodes send data directly to the CH, while member nodes overhear the transmitted data and understand whether additional information is required, e.g., because the measure that was transmitted to the CH by a representative node is too different from the one that is sensed by the member node. Otherwise, the member node tacitly consents the data transmitted by the representative, and does not transmit any data.

In order to reconstruct the missing information, the sink uses a custom estimation function to determine the value coming from the member nodes by means of the values of the neighboring representatives. The precision of the reconstruction clearly increases with the number of representative nodes, which however leads to a reduction of the network lifetime [132] and of the capacity of the wireless channel [95]. Therefore, a trade-off between energy consumption and estimation error arises: more transmissions (i.e., more measurements) lead to a reduction of the estimation error and an increment of the energy consumption and, vice versa, few transmissions reduce energy consumption while decreasing the accuracy of the estimation.

With respect to the state-of-the-art approaches, the proposed technique achieves a considerable reduction of the energy consumption with high estimation accuracy. Moreover, differently from the existing works, the proposed approach employs custom estimation functions and is able to work even without absolute spatial information, i.e., the physical node position. Finally, the proposed technique is not limited to a specific network protocol and can be adapted to many kinds of wireless sensor networks.

State of the Art Clustering has been extensively studied in the literature, and different approaches have been proposed. CLUBS [120] is a clustering technique where every node belongs to a cluster, whose diameter is the same for all the clusters. Other approaches aim at defining a multi-tier hierarchical clustering strategy [30]. Sizes and degrees of overlap of the clusters are taken into account while grouping nodes and managing the hierarchy. To tackle energy efficiency of WSNs, several approaches have been presented. The Energy Efficient Hierarchical Clustering, proposed in [87], is a distributed and randomized clustering algorithm designed with the objective of optimizing the network lifetime. Similarly behaves the Low Energy Adaptive Clustering Hierarchy (LEACH) [69] approach, which is one of the most referenced and popular clustering algorithms for WSNs. The protocol creates clusters based on the strength of the received signal and uses the CH nodes - elected with probabilistic function - as routers to the BS for the communications coming from their cluster members. Several modifications of the LEACH strategy have been proposed, such as TEEN [106], APTEEN [107] and PEGASIS [97]. For further information about clustering algorithms, the reader may refer to the survey in [19].

Spatial Correlation is a unique characteristic of WSNs since it generates highly-correlated information, whose size can be reduced by following well-known results in the field of lossless transmission [48]. Differently from the above clustering/routing protocols, spatial-correlation-aware algorithms exploit this characteristic and the consequent data redundancy to improve the efficiency of the network. As redundancy is reduced, these algorithms aim at finding a good tradeoff between energy saving and loss of accuracy due to data aggregation policies.

Yoon et al. [165] proposed CAG, an aggregation technique that provides approximate results by exploiting spatial correlation in aggregate queries. CAG aggregates data at CHs according to the specific query and routes aggregated data to the sink. Liu et al. [98] presented a data collection method based on a careful analysis of surveillance data reported by sensors. By analyzing the spatial correlation of the sensed data, they dynamically partition the sensor network into cluster with similar time series.

The most recent and efficient algorithm for spatial-correlation-aware data collection is YEAST [157]. In fact, this algorithm has been shown to outperform many other state-of-the-art algorithms in terms of scalability, flexibility, accuracy, aggregation rate, overhead and energy consumption. In particular, YEAST reduces data redundancy by defining adaptive **correlation regions**, where sensed data is supposed to be uniform. Each correlation region elects a set of *representative* nodes that detect an event and report the gathered data to a coordinator. Routing toward the sink is performed using geographical information and in-network collection is performed. To estimate measurement's values of member nodes, an estimation function based on node's position is used. Nevertheless, YEAST is strongly limited by two assumptions: nodes need to know their (absolute) position in the network, and the sensed physical variable has to be uniform within the correlation region. Moreover, YEAST only works with event-based networks. On the other hand, this approach does not rely on the aforementioned assumptions, and it also supports periodic monitoring. In addition to this, while all the presented approaches work with specific, ad-hoc network protocols, this approach is not bound to a specific routing protocol, hence it can be adapted to a wide range of network architectures.

The Proposed Methodology The main issue of spatial-correlation-aware data transmission is the definition of the correct reporting rate, i.e., the percentage of transmitting nodes over the total number of nodes in the WSN. One the one hand, a low reporting rate leads to a limited energy consumption, but the accuracy may fall below an acceptable distortion bound [158]. On the other hand, a high reporting rate increases both the accuracy and the throughput of the network, thus leading to a higher energy consumption.

To reduce the network throughput, YEAST uses the so-called *correlation regions* to estimate measurements and keep the number of transmissions as low as possible. In a correlation region, only the representative nodes send their measurements, thus the sink receives only one measurement from each correlation region. This approach works well in scenarios where the physical variable does not significantly vary within the correlation region. However, this approach does not scale well with the size of the correlation region, as spatial correlation decreases when the distance among the nodes increases [158].

Before introducing the proposed approach, let us define the **relation graph** as a directed graph $G\langle N, A, M_n(t), \phi_{n,m}(M_n, t, P_{n,m}), t\rangle$ where N represents the nodes of the network, A the connections between nodes, $M_n(t)$ the measurements at time t of node n, ϕ the relationship between two nodes (n and m) at time t, and P_{n_m} a set of numerical parameters required by the estimation function. The estimation function (ϕ) depends on n and m (thus $\phi_{n,m}$) and is able to estimate M_n from M_m and $P_{n,m}$. An example of $P_{n,m}$ is the distance between measurements (offset) or a scaling factor.

This approach is more general than *correlation regions*, since a correlation region is an area where $\phi(M_m, t, P_{n,m}) = M_m$ for all t and $P_{n,m}$. Moreover, correlation regions have a predefined shape and position. A comparison between correlation regions and correlation graphs is illustrated in Figure 5.9. In YEAST, the topology is constrained by the size and the position of the correlation regions while, in the *relation graph*, the relationships between nodes are expressed according to the knowledge of the physical phenomenon.

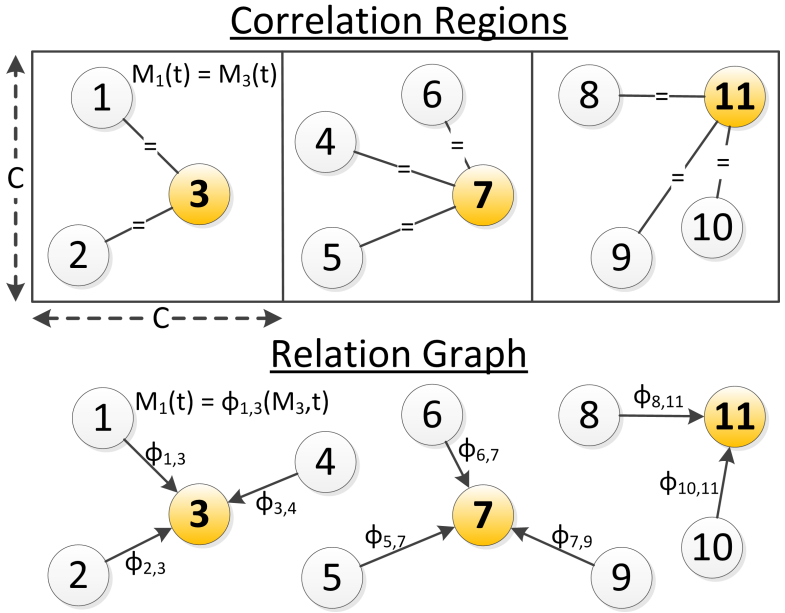


Figure 5.9: *Comparison between Correlation Regions (top) and Relation Graph (bottom). In a correlation region, the measurement of a node is assumed to be equal to the measurement of their representative node. In a relation graph, a relation between member and representative nodes is specified by the ϕ function*

The relations of a *relation graphs* have to be defined at design-time, since it is usually very complex to extract this information at runtime. On the other hand, the remaining parameters can be tuned at runtime in order to perform accurate estimations. For example, if the measurements are temperatures, it is well known that the difference between the temperatures of two nodes in the network is equal to the temperature gradient, but the specific gradient in each part of the network has to be estimated. In this example, $\phi(M_m, t, P_{n,m}) = M_m(t) + \Delta_{n,m}(t)$, where $\Delta_{n,m}(t)$ is a scalar that represents the temperature gradient between nodes n and m at time t.

The TaCo Algorithm The proposed algorithm is called **Tacit Consent** (***TaCo***), and it groups the nodes of the WSN into two classes:

- **representative Nodes**, which represent a specific area of the network. They act as normal nodes: when a new measure is available, they will transmit it immediately. They can be defined at design time or at runtime;
- **member Nodes**, which are represented by other nodes. A member node is represented by a single representative node. Member nodes communicate with the sink if and only if it is strictly required.

Two types of data messages can be sent by the nodes:

- `MEASURE_PKT`: used to communicate the current measurement (M_n) to the sink. It is composed of three fields: (`n`, `sensorID`, `M`). This packet is sent by representative nodes;
- `PHI_PKT`: used to communicate the parameters ($P_{n,m}$) of the estimation function to the sink. It is composed of five fields: (`n`, `m`, `sensorID`, `P`). This packet is sent by member nodes.

TaCo can be configured with the following set of parameters, which appear in Equation 5.3:

- *RSSI_threshold*: minimum amount of RSSI (Received Signal Strength Indication) to consider two nodes as neighbor. It affects the number of representative nodes;
- *E*: maximum estimation error ϕ tolerated by the application. Higher is E, less frequently will be the `PHI_PKT`s;
- δ: desired degree of member nodes for the representative nodes. It is used when representative nodes have not been defined at design time;
- w_1, w_2, w_3: positive numbers that are used to dynamically determine the representative.

TaCo consists of two phases: *setup phase* and *execution phase*. The **setup phase** is required only when the role of the nodes (representative or member) has not been specified at design time. This phase consists of three steps: in the first step, each node identifies its neighbors by broadcasting *HELLO* messages at a low transmission power and receiving sensitivity (in order to reach only spatially co-located nodes); only nodes with a RSSI greater than *RSSI_threshold* are chosen as neighbors. In the second phase, each node computes a **weight**, that depends on the desired degree of neighbors (δ), the average quality of signal ($RSSI$) of the neighbors set ($N(n)$) and the consumed energy:

$$W_n = w_1|d_n - \delta| - w_2 \sum_{v \in N(n)} \frac{RSSI_v}{|N(n)|} + w_3 E_n \tag{5.3}$$

where d_n is the number of neighbors of n, δ is the desired amount of neighbors and E_n is the consumed energy (w_1, w_2, w_3 are non-negative weights). Both the function and the weights are inspired to DWCA [47]. It has been observed that this function provides good clusters and fairness between nodes, but a different clustering technique

can be used as well. When weights are computed, they are broadcasted to the other nodes. Representative nodes are the ones with the lower weights among the neighbors. Thus, once the weights of all the neighbors have been received, a node becomes representative if no neighbor with a higher weight exists. As a third step, representative nodes communicate the role to the neighbors member nodes, and each member node selects the node with the highest RSSI as its representative.

Once the setup has been completed, TaCo enters in the **execution phase**, whose pseudo-code is presented in Algorithm 7. In this phase, each representative node operates as a normal node by transmitting a new measure toward the sink as soon as the new measure is available, thus, when a new measurement is sampled (`SAMPLE`), representative nodes transmit it immediately and member nodes store it. The member nodes silently listen to the transmission (**overhearing**) and compare the estimation of the measurement ($\phi(L, P)$) with the actual value (M). If the estimation error is greater than E, the specific member node transmits a `PHI_PKT` to update $P_{n,m}$ in order to keep the estimation error within the tolerance. Otherwise, if the estimation is correct, the member node **tacitly consents** the estimation and does not send any packet. To perform overhearing, member nodes must be awake when their representative node transmits. Since the underlying routing/MAC protocol may manage representative and member nodes in a different way, TaCo schedules its own `wake-up`'s regardless of the MAC protocol.

The estimation accuracy (e) is computed as the ratio between estimated value (α) and accurate measurement (β):

$$e(\alpha, \beta) = 100 - \left(\frac{\mid \alpha - \beta \mid}{\beta} \times 100 \right) \tag{5.4}$$

The same function has been used in [157] to validate YEAST.

Algorithm 7: Execution Phase of TaCo algorithm

```
input : Packet or sample (I)
if I is SAMPLE then
    if node is REPRESENTATIVE then
        M = readSample(I);
        sendMeasure(M);
    else if node is MEMBER then
        M = readSample(I);
    end
else if I is MEASURE_PACKET then
    if node is REPRESENTATIVE then
        // discard packet
    else if node is MEMBER then
        L = getMeasure(I);
        if e(M, φ(L, P)) > E then
            P = updateParams(L,M,E);
            sendParams(P);
        end
    end
end
```

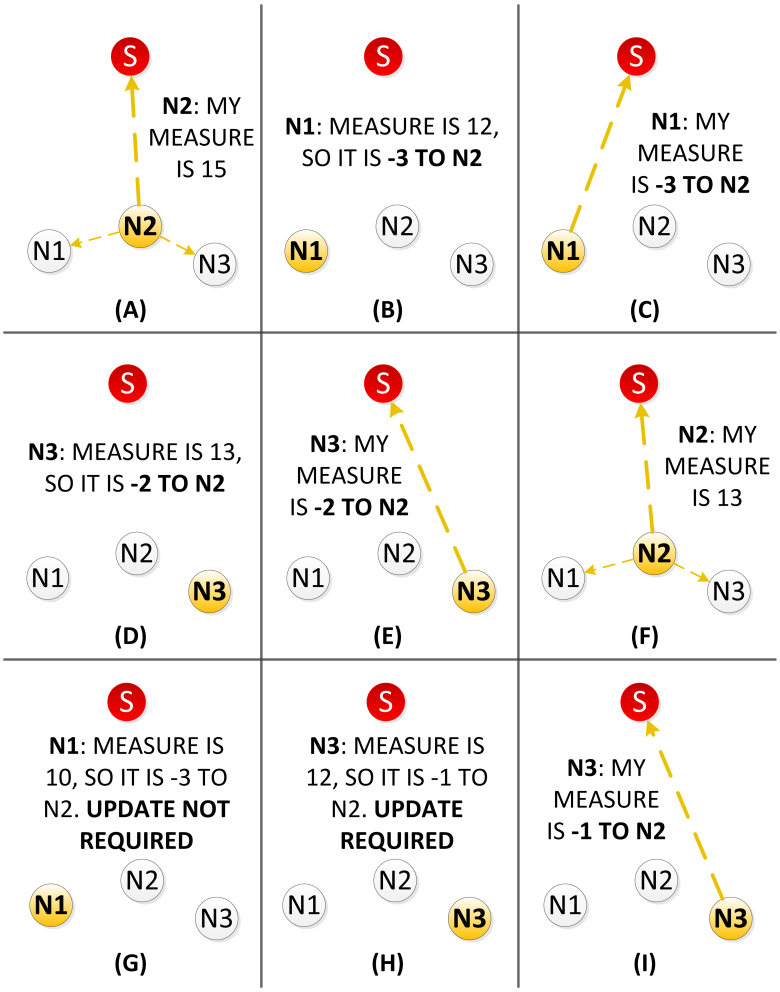


Figure 5.10: *An example of the proposed algorithm. $N2$ is a representative node while $N1$ and $N3$ are non-representative nodes. The red node, represented with S is the sink. The ϕ function used here is $M_n = M_m + \Delta$*

To better understand the idea of TaCo, an example is illustrated in Figure 5.10. In this example, the estimation function ϕ corresponds to $M_n = M_m + \Delta$. At the beginning (***A***), $N2$ reads and sends its measure (15) to the sink (the red node, represented with S). At the same time, $N1$ and $N3$ listen (overhear) the measure and store it. Then $N1$ reads 12 (***B***), computes Δ (offset), which is *-3 to N2*, and transmit Δ to the sink (***C***). It transmits Δ because it is the first data sent to the sink. Similarly, $N3$ reads data from its sensors and compares it to the value sent by $N2$ (***D***). Once Δ is determined, $N3$ transmits it to the sink (***E***). After a certain period of time that depends on the sampling period, $N2$ samples a new measure (13) and transmits it to the sink (***F***). Both $N1$ and $N3$ silently listen $N2$ measure and store it. Later, $N1$ reads the measure from its sensors and compare it to $N2$ measure (***G***). Since the measure is 10, Δ still remains the same, so no update is required and no transmission is required; $N1$ **tacitly consents** sink's estimation. Differently, $N3$ reads 12 from its sensors which means that Δ is different from the previous one (***H***). Finally, $N3$ retransmits an updated value of Δ (***I***).

Integration of TaCo with Existing MAC/Routing Algorithms TaCo can be easily integrated with existing MAC/Routing protocols, making it possible for the designers to identify the correct MAC/Routing layer according to system constraints and requirements (cost, availability, performances, etc.). This is possible since TaCo requires only single-hop broadcast communications (supported by many MAC layers) and the creation of the relation graph is not bound to the network architecture. As a matter of fact, the packets of a member node can follow a different path with respect to the one its representative, since the final estimation is performed directly by the sink. The example in Figure 5.10 presents a direct communication with the sink, but, since no feedback from the sink is required, the communication can exploit any kind of routing algorithm (single/multi-tier, opportunistic, probabilistic, multi-hop, etc.). In the experimental results, two different routing protocols has been used: Directed Diffusion [74] and a simple version of the Synopsis Diffusion [122]. Moreover, both have been tested with S-MAC [162] and B-MAC [135].

Figure 5.11 illustrates how to integrate TaCo with existing MAC/Routing layers. TaCo has been designed to interact with routing and MAC layers directly, and its execution can be controlled by the application to determine which measurements are handled by TaCo and which ones by the application.

Experimental Results The proposed approach has been validated by comparing it to YEAST in terms of both energy and accuracy. Experimental results have been conducted using Castalia [9], a popular simulator for Wireless Sensor Networks and Body Area Networks, based on the OMNeT++ framework [155]. Both TaCo and YEAST have been implemented in the same simulator in order to ensure a fair comparison. We use the *Customizable Physical Process* provided by Castalia to analyze the behavior of the two algorithms in different environmental conditions. This physical model is composed by a set of sources whose influence is diffused over the bi-dimensional space [39]. The effect of multiple points is additive, as shown by the following equation:

$$V(p,t) = \sum_{\forall i} \frac{V_i(t)}{(K d_i(p,t)+1)^a} + N(0,\sigma) \tag{5.5}$$

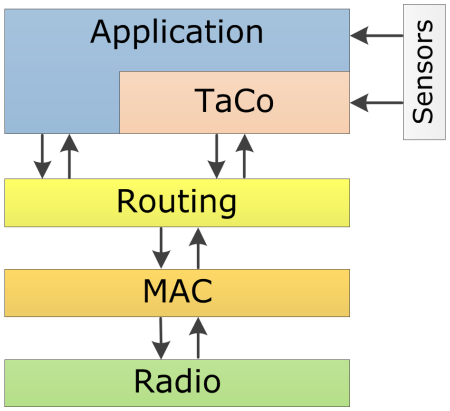


Figure 5.11: *Integration of TaCo with existing MAC/Routing layers. TaCo has been designed to interact with routing and MAC layer directly, with or without the application*

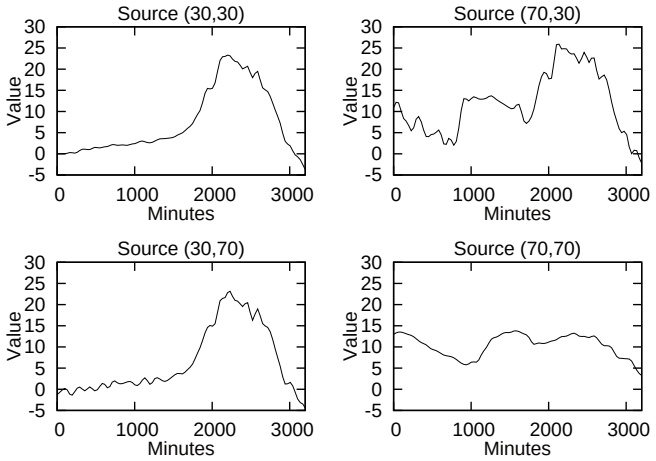


Figure 5.12: *Values of the four sources used in the Castalia's Customizable Physical Process in the experiments. Sources are located in (30,30),(70,30),(30,70),(70,70) [m,m]*

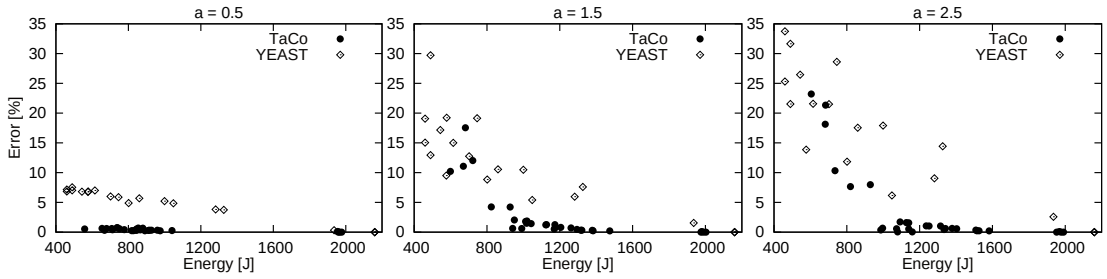


Figure 5.13: *Energy-Error tradeoffs obtained by **TaCo** and **YEAST** with $a = 0.5$, $a = 1.5$ and $a = 2.5$*

where $V(p,t)$ denotes the value of the physical process at point p at time t, $d_i(p,t)$ the value of the source i at time t, while K and a determine how the values are diffused, and $N(0,\sigma)$ is a zero-mean Gaussian random variable. In these experiments, we vary the value of a in order to determine if the proposed approach correctly works in different situations: higher values of a imply stronger gradients between points in the space. In these experiments, four sources, respectively located in $(30,30),(70,30),(30,70),(70,70)$, were used. The parameters used in the experiments are listed in Figure 5.12. Values in the other position in the network are computed with Equation 5.5. In all the experimental results, $k = 0.25$, $\sigma = 0$ and a varies from 0 to 2 according to the specific experiment.

The main parameters used in the simulation are presented in Table 5.1. Nodes are uniformly distributed according to $\sqrt{n\pi r_c^2/d}$ [157] where n is the number of nodes, r_c the communication radius and d the average degree of neighbors.

The first experiment highlights the different energy-accuracy tradeoffs that can be obtained using TaCo and YEAST, with different values of a. Figure 5.13 shows the different solutions, which correspond to different numbers of representative nodes: a higher number of representatives lead to a solution in the bottom-right corner of the energy/error graph (high energy consumption and low estimation error), whereas a low number of representatives lead to a solution characterized by low energy consumption and high estimation errors. The results show that in general TaCo outperforms YEAST

Table 5.1: *Simulation parameters with range of values*

Parameters	Values
Number of Nodes	(50,100,225,400)
Area $[m \times m]$	100×100
Sample Rate [sample/min]	1
Transmission power [mW]	57.42
Receiving power (Low Sensitivity) [mW]	32
Receiving power (High Sensitivity) [mW]	62
Length of the Simulation [min]	3200
Radio Module	CC2420
MAC Layer Module	(S-MAC, B-MAC)
Routing Layer Module	(Directed Diffusion, Synopsis Diffusion)
Simulation Run (for each configuration)	50

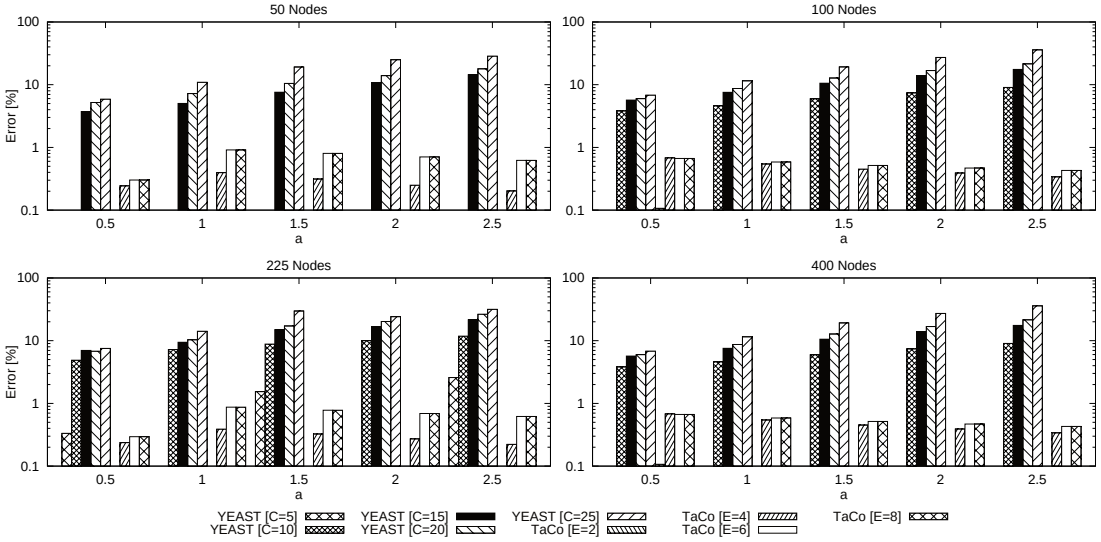


Figure 5.14: *Average estimation error of YEAST and TaCo for 50, 100, 225 and 400 nodes with S-MAC. YEAST was tested with $c = \{5, 10, 15, 20, 25\}$ and TaCo with $E = \{2, 4, 6, 8\}$. Error is computed with Equation 5.4 and Y axes are logarithmic*

both in terms of energy consumption and estimation accuracy, especially in the scenarios when the maximum acceptable error is below 5%.

The second experiment was conducted to determine the effectiveness of the proposed approach with respect to the estimation error. It has been conducted on both YEAST and TaCo equipped with S-MAC. In this case, routing is not considered since it is not relevant for estimation accuracy. YEAST has been tested with $C = \{5, 10, 15, 20, 25\}$ and TaCo with $E = \{2, 4, 6, 8\}$. Experimental results are presented in Figure 5.14. TaCo is able to keep the error below the desired error while YEAST has no control about the current estimation error. Please note that actual average error in TaCo is always lower than E, which represents the maximum tolerated error. In YEAST, data from member nodes are completely ignored and thus no control on the current error is performed. Even if *correlation regions* can be reconfigured at runtime, estimation error strongly depends on the correlation region width (C) and increases to more than 20%.

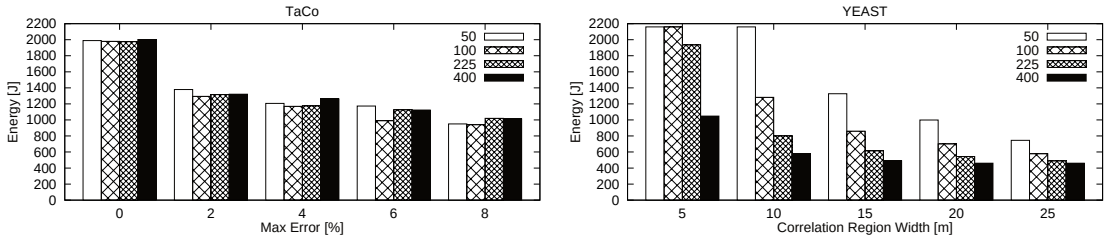


Figure 5.15: *Average Energy Consumption of **TaCo** for 50, 100, 225 and 400 nodes. RSSI threshold is equal to $-95dBm$ and $a = 1.5$ with **B-MAC** and **Synopsis Diffusion***

The third experiment was conducted on TaCo and YEAST with B-MAC and Synopsis Diffusion in order to determine the relationship between parameters and energy consumption. Two distinct tests the following values have been varied: in the first experiment the TaCo's error tolerance (E) and, in the second, the width of correlation regions (C) in YEAST. Figure 5.15 illustrates the energy consumption of TaCo and YEAST. In TaCo, increasing E reduces energy consumption. This result was expected since a greater tolerance reduces the amount of `PHI_PKT` that must be transmitted. On the other hand, correlation region width (C) strongly affects energy consumption such as higher values of C decreases the amount of representative nodes, reducing the energy consumption. This phenomenon confirm the experimental results in [157].

Summary This Section presented TaCo, a new technique to exploit spatial correlation and provide accurate measurements in dense wireless sensor networks with a limited energy consumption. Thanks to the high correlation between co-located nodes, it is possible to correctly predict the measurements from the values sensed by the neighboring nodes. This allows to drastically reduce the number of transmitted packets and hence the energy consumption of the whole network. Experimental results prove that TaCo is able to find better energy/accuracy tradeoffs with respect to YEAST, the most efficient spatial-correlation-aware algorithm that can be found in the literature. In particular, TaCo greatly reduces the energy consumption of all the solutions that guarantee a high accuracy (i.e., a low estimation error). Finally, TaCo allows the designer to freely select the MAC/Routing protocol, thus making it applicable to a broad class of WSNs.

5.3 Concluding Remarks

Design-time analysis and algorithms aim at identify the optimal network configuration for the given metrics, but runtime issues like interferences, node failures or node mobility can dramatically affect the performance of the network. The need to adapt the network at runtime is of extreme relevance to ensure the operational effectiveness and performance of the network as expected.

This Chapter presented two specific approaches that deal with online adaptivity of Wireless Sensor Networks: a technique to reduce BAN-BAN interferences in WSNs, and a technique to reduce the amount of transmissions in a cluster-based network. The first technique provided an effective way to reduce, or even eliminate, BAN-BAN interferences in IEEE 802.15.4-based networks. The second technique exploited the concept of spatial correlation to control and reduce the amount of required transmissions in a

cluster-based network.

6 Application Case Study

This Chapter details a complete application case study that aims at presenting how the proposed design flow can be applied to a real case study. The complexity and variety of Wireless Sensor Networks's applications make it impossible to create a comprehensive case study; as a consequence, certain processes can be trivial, while others require several hours or days to perform. However, the purpose of this chapter is not to present the effectiveness of specific techniques, rather an high-level view of how the proposed design flow can be used on a real-world case study.

The case study presented in this chapter has been inspired to the work contained in [63] and conducted in collaboration with the Hydrology Department of Politecnico di Milano. The network has not been deployed, thus all the measured data are collected with the OMNET++ Castalia simulator [9], enriched with realistic data from the datasheets of the platforms presented in Chapter 3.

6.1 Description of the Case Study

Consider an area of $5km$ per $3km$ as shown in Figure 6.1. This area is divided in three distinct zones by a lake and a river, that crosses the area from north to south. the first zone, located at west of the river, has a small hill covered by trees, the second zone, at the east of the river, is a plain and the third zone, located at the south between the rivers, is covered by a dense forest. The objective of the design phase is to place and configure a wireless sensor network in order to retrieve needed data from the environment.

There are four groups of variables to gather from the environment:

- **Water Level**: the actual water level of the lake;
- **Water Flow**: the actual water flow of the river;
- **Temperature, Humidity and Rainfall**: actual temperature, humidity and rainfall data gathered from the environment;
- **GAS**: actual value of gases like CO, CO_2, O_3, etc...

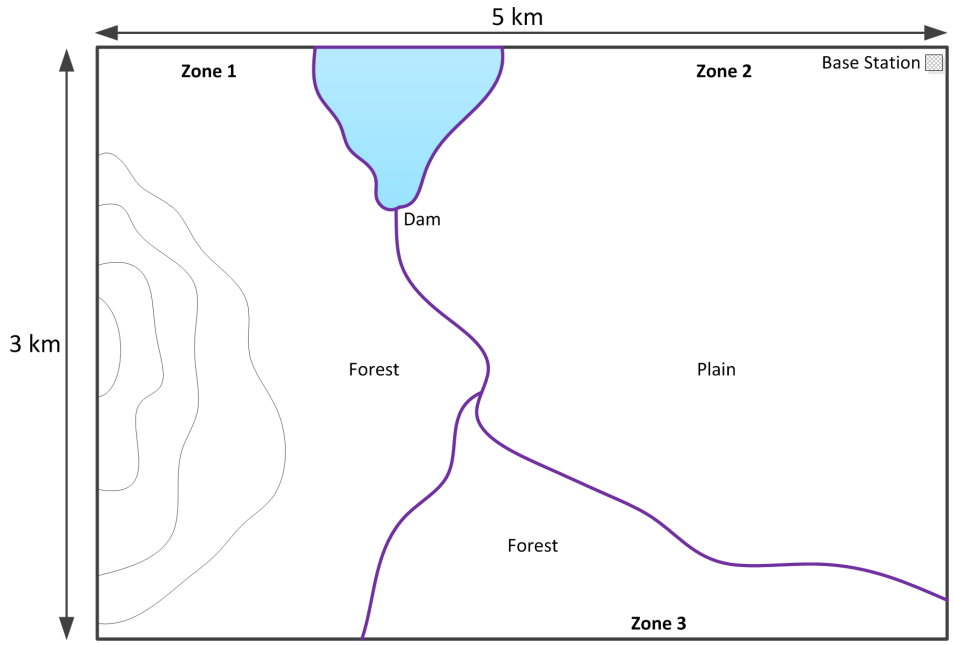


Figure 6.1: *Main Field of the Application Case Study*

METRICS REQUIREMENTS

The network must be able to collect the required data from the environment with a **Packet Receive Ratio** (number of packets received on the number of packets sent) **higher than** 95% and with the **maximum achievable lifetime**.

6.2 Design Phase

6.2.1 Sensing Coverage

According to the proposed design flow, the first step of the design is the definition of the position of the sensors (**Sensing Coverage**). In the specific case study, this position is defined **manually** by specialists, that identify the correct number, their type and position. The result of this first step is shown in Figure 6.2.

6.2.2 SW Development

The network of this case study acts as a *push network*, meaning that all the nodes periodically sends their data to the **Base Station** for further analysis. This type of application is extremely popular in WSN field, and it can be easily implemented on any platform and operating system. An example of a task in *push networks* is illustrated in Algorithm 8. This task, takes the sensor ID (s_{id}) as input parameter in order to identify which sensor is controlled by the actual task among the available sensors on the platform. It first reads a new sample from the assigned sensor and store it in memory $(x(s_{id}, t))$. Then, the function *reportRequired* is evaluated; this function evaluates if the actual value must be transmitted to the Base Station or not. Although in *push networks* data are always transmitted to the Base Station, energy saving policies could delay or

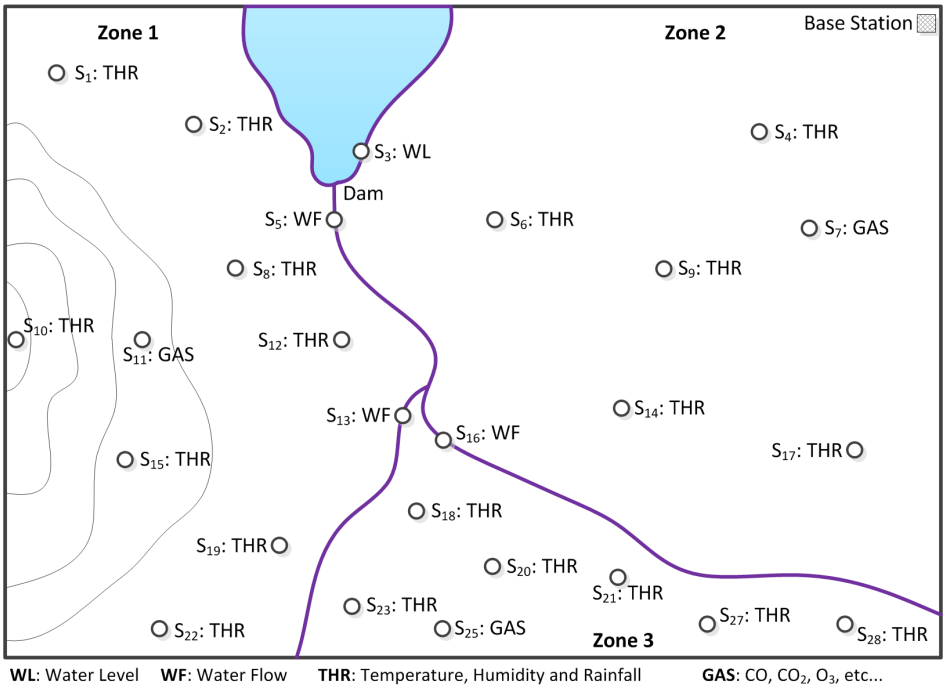


Figure 6.2: *Result of the Sensing Coverage Process as Specified by the Specialists (Manual Operation)*

block the transmission, similarly to what Tacit Consent algorithm does (see Section 5.2). In case *reportRequired* function returns *true*, the actual value is transmitted to the Base Station, otherwise sample is ignored.. At the end, the task set an interval for its re-activation; this specific function can differ according to the chosen operating system.

SW Development is typically performed **manually**, since the application hardly depends on the problem to solve.

Algorithm 8: Example of Sensing Task in Push Networks

input: s_{id}

```
1 x(s_id, t) = getSample(s_id)
2 if reportReqired(x(s_id,t)) then
3 |   send(x(s_id,t), BASE_STATION)
4 end
5 setInterval(T, SampleVariable(s_id))
```

6.2.3 Network Connectivity

Third step of the design flow is the *Network Connectivity* process, that inserts additional nodes, called **relay nodes** in order to create a connected network. Assuming an effective transmission range of 800 meters, the network connectivity of the actual network (before *Network Connectivity* process) is illustrated in Figure 6.3; all the nodes in the network cannot communicate with the Base Station, and most of them are isolated. To perform this process, the algorithm described in [150] has been used; the algorithm has been introduced in Section 4.3. The output of this algorithm has been **automatically**

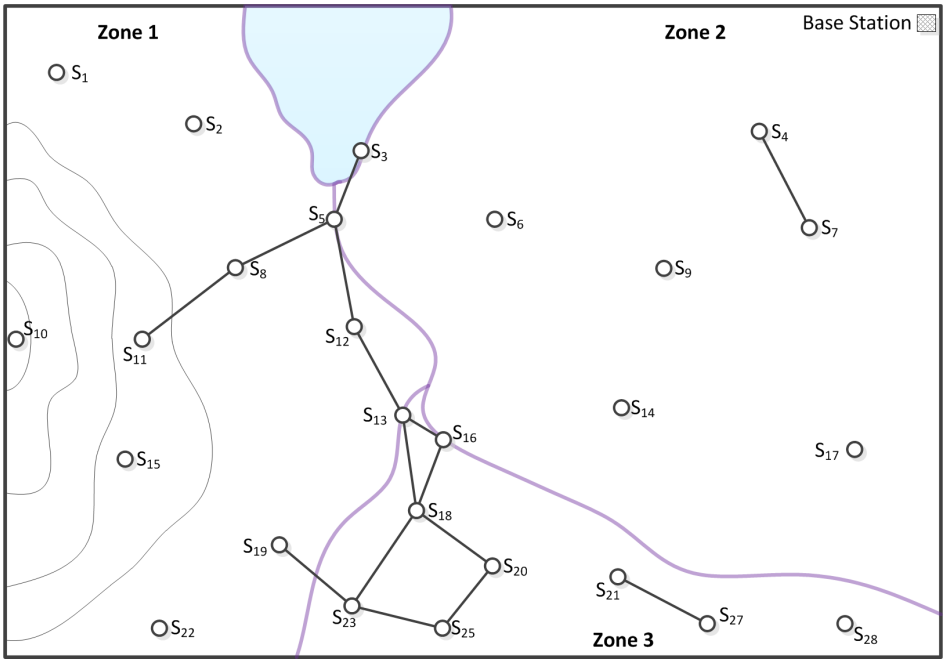


Figure 6.3: *Situation of the Network **Before*** Network Connectivity *Process*

computed and it is illustrated in Figure 6.4. At that point, all the nodes are able to communicate with the Base Station.

6.2.4 Hardware Design

The actual application case study does not require custom hardware platforms, and device size and costs are not constraints. For this purpose, according to the required sensors, **Wasp Mote** from **Libelium** (see Section 3.1) best fits the desired requirements. The company provides ready-to-use sensor boards as well as batteries and solar panels. Moreover, Over the Air Programming (OTA) is a desired feature in such applications, since direct programming could be difficult and complex. This process is perfomed manually.

6.2.5 OS Definition

Once hardware has been defined, the Operating System must be chosen. According to the analysis in Section 3.1.1, Libelium supports only **FreeRTOS**, thus the definition of the OS is very simple. However, FreeRTOS is a good operating system for this particular application; it supports *multi-threading*, *mutexes* and *semaphores*. Moreover, overhead of FreeRTOS is very limited. This process is perfomed manually.

6.2.6 Network Protocol Definition and Configuration

Except for *Network Connectivity* process, the other processes have been executed manually, performing only qualitative analysis. **The remaining processes will be executed automatically**, in order to optimize the lifetime of the network, keeping them fully functional. These two processes have the objective to identify which are the network

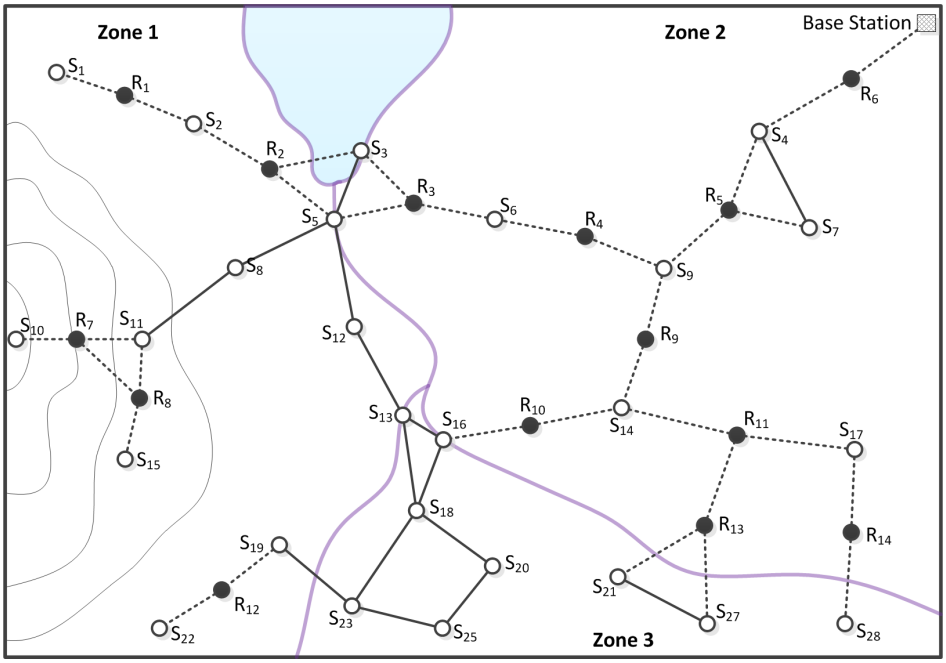


Figure 6.4: *Situation of the Network **After*** Network Connectivity *Process (Automatic Operation)*

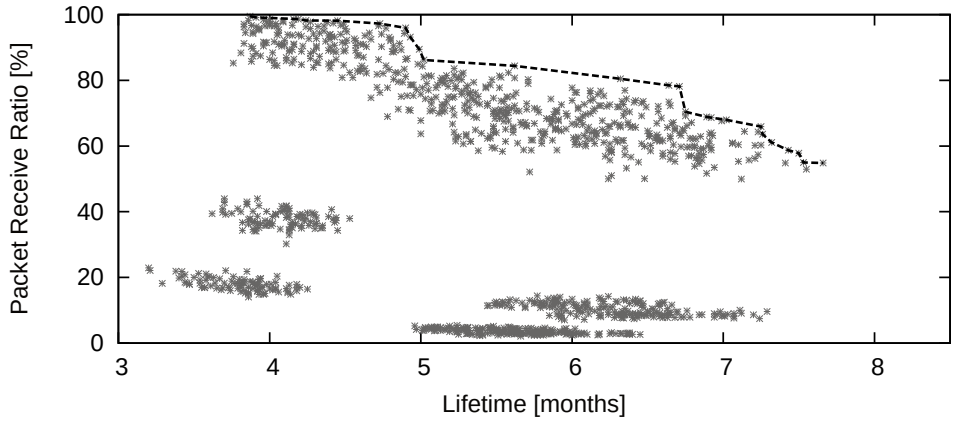
protocols (MAC and Routing) and how they must be configured in order to maximize the lifetime of the network, keeping the packet-receive-ratio at the Base Station higher than 95%, as specified in the requirements. In this study lifetime is considered as ***the time until ONE node stop working***. This analysis does not keep into consideration energy harvesing, so worst case scenario is always assumed.

Analyzing the topology of the network in Figure 6.4, it is possible to notice that the network is a single-tier network, then no cluster methods are suggested. It includes the use of the popular IEEE 802.15.4 protocol, that works in two-tier networks. Candidate MAC protocols (supported by the simulator) are:

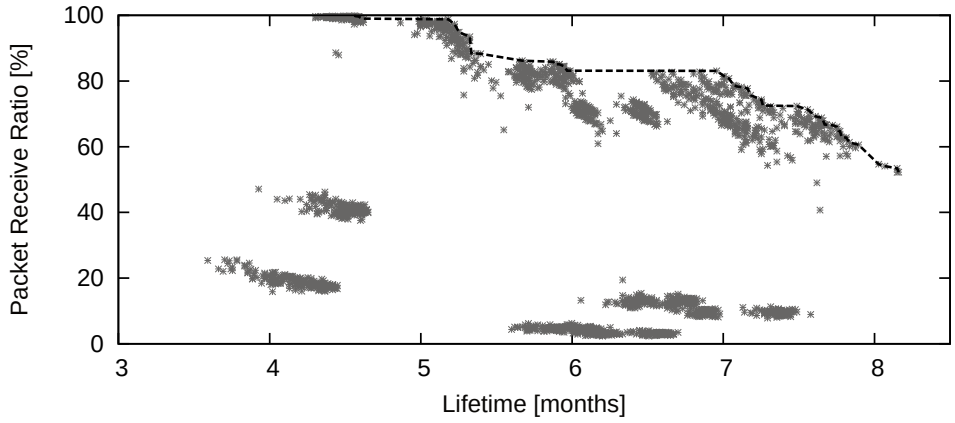
- **T-MAC**: a popular MAC for WSN, it adapts the duty cycle according to the traffic needs and uses SYNC, ACK, RTS and CTS packets for synchronization and to maintain low values of packet-error-rate;
- **S-MAC**: the predecessor of T-MAC, it uses a more rigid duty cycle;
- **Tunable MAC**: a complex and highly-customizable MAC layer with variable duty-cycle and no RTS/CTS packets. It only supports broadcast communications.

To define which protocol best fit the desired needs, **three distinct Design Space Explorations** have been conducted on the protocols. The objective is to detect an optimal configuration of the network, defined as the configuration with at least 95% of *Packet Receive Ratio* and the maximum *Lifetime*. To perform a realistic simulation of the applicaiton, a fixed routing has been used. For the explorations, NSGA-II, PMA and MOSA have been used; the results show the points identified by all the algorithms.

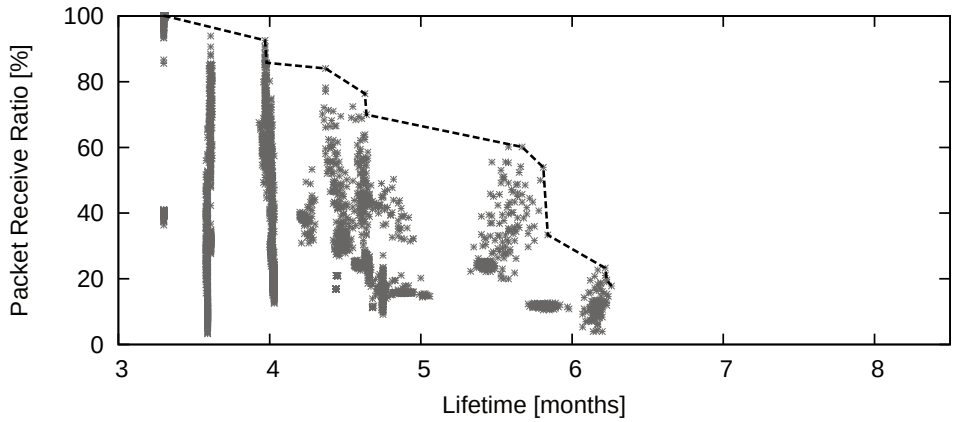
Figure 6.5 illustrates the points and the Pareto curves extracted by the three explorations. The metrics of interests are lifetime (in months) and packet-receive-ratio (in percentage) at the Base Station. The algorithms aim at maximizes both metrics. Pareto



(a) SMAC



(b) TMAC



(c) Tunable MAC

Figure 6.5: *Design Space Exploration of SMAC, TMAC and Tunable MAC (Automatic Operation)*

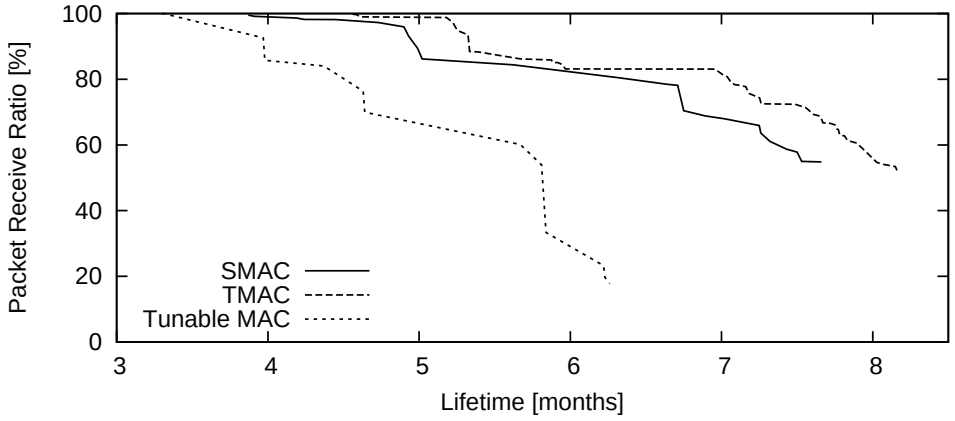


Figure 6.6: *Comparison of the Pareto Frontier of SMAC, TMAC and Tunable MAC (Automatic Operation)*

frontiers of SMAC, TMAC and Tunable MAC are illustrated in Figure 6.6 for comparison. According to the initial requirements, a Packet Receive Ratio of 95% is needed, thus all the solutions below this threshold are excluded. Experimental Results show that *Tunable MAC* is dominated by both SMAC and TMAC, thus it is not the right choice for this application. On the other hand, TMAC dominates SMAC and has a better lifetime for solutions with Packet Receive Ratio higher than 95%, thus TMAC is chosen as MAC layer in this application.

At this point, routing protocol must be defined. According to topology, similarly to what has been done for MAC layer, the protocols available are:

- **Ad Hoc On-Demand Distance Vector** (**AODV**): a reactive routing protocol that creates custom paths from source to destination on-demand. It is capable to deal with topology changes and faults in the network;
- **Destination-Sequenced Distance Vector (DSDV**): table-driven routing protocol, derived from the Bellmann-Ford algorithm. Differently from AODV, it is more efficient but requires constant update in the routing tables;
- **Directed Diffusion**: scalable and robust routing protocol for WNS, it routes data from source to Base Station by computing the gradient descend.

These routing protocols have few parameters to set, thus an exhaustive search is preferred. The results, given in Table 6.1 have been **automatically** computed using the MAC layer defined in the previous step and configured to give a Packet Receive Ratio of 95% and the maximum achievable lifetime (that is about 5.2 months). According to the results, **Directed Diffusion** dominates both AODV and DSDV and improves both

	AODV	DSDV	Directed Diffusion
Lifetime [months]	5	5.2	5.8
Packet Receive Ratio [%]	95	92	97

Table 6.1: *Results of Routing Protocols' Definition Process (Automatic Operation)*

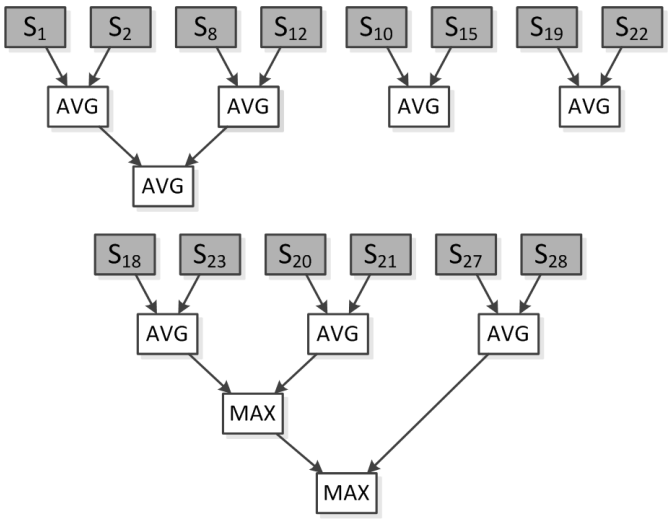


Figure 6.7: *Definition of the Data Flow Graph (Manual Operation)*

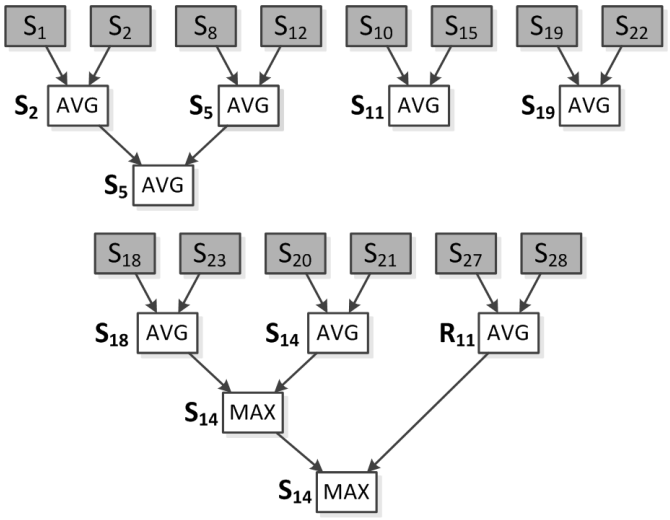


Figure 6.8: *Partitioned and Mapped Data Flow Graph (Automatic Operation)*

the lifetime and the packet receive ratio with respect to the configuration with fixed routing, thus it has been identified as candidate routing protocol for this application.

6.2.7 SW Partition and Mapping

A very popular technique to reduce the number of transmission in the network is to aggregate the data during the routing. Many routing protocols already implement this functionality, but for this application it is preferred to specify the aggregation policies **manually**, while *Partiion* and *Mapping* will be executed **automatically**. The resulting Data Flow Graphs (DFGs) are illustrated in Figure 6.7 for Zone 1 (top) and 3 (bottom). In all the DFGs, gray nodes are sampling tasks from the corresponding nodes, thus they are contrained to that node, while white tasks perform aggregating operations, thus can be executed on every node in the network, relay nodes included.

According to the results presented in Section 4.5.1, **GA+Tournament** algorithm has

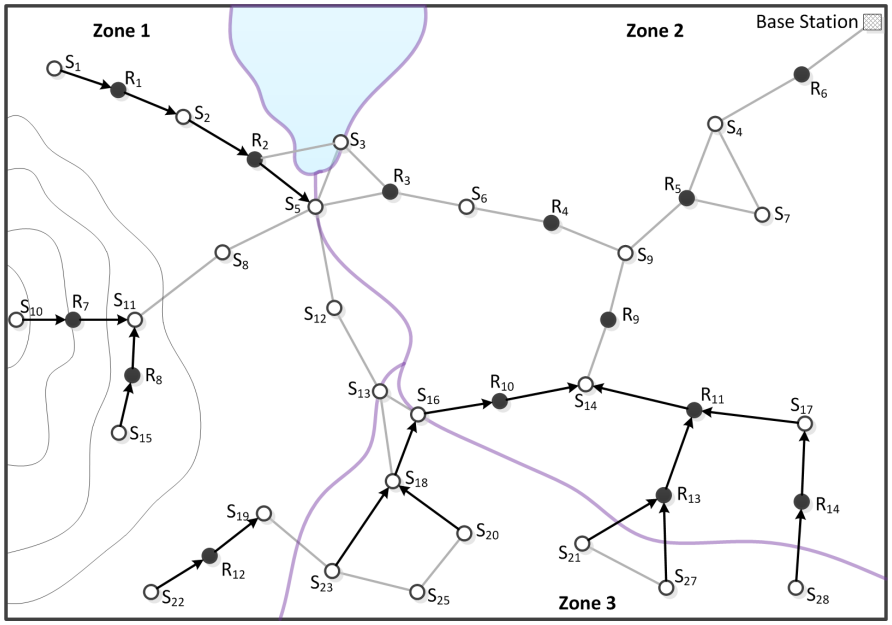


Figure 6.9: *Aggregation Routes After Partition and Mapping Process (Automatic Operation)*

been used. The best solution identified by the algorithm is illustrated in Figure 6.8. To better appreciate and understand the effectiveness of the algorithm, aggregating routes are plotted in Figure 6.9.

6.3 Concluding Remarks

This Chapter illustrated a complete design problem from the high-level specification given by the customer to the final deployment, through all the processes of the proposed Design Flow. The objective of this Chapter was to show the reader an example of how to apply the proposed design flow to a real-world case study. In this chapter became clear that all the processes are important in the identification of the deployed solution, and some of them could be easier and faster than others, depending on the case study. In the example presented in this Chapter, hardware design process and OS definition were extremely fast, since there are different platforms that are able to support the desired application; in Wireless Body Sensor Networks application, on the other case, hardware design is extremely important to guarantee low-energy, high-reliable systems.

7 Concluding Remarks

The optimal design of a Wireless Sensor Network (WSN) is extremely complex to obtain, since it involves the identification and configuration of several aspects regarding hardware platforms, network protocols and software development. All these aspects are inter-dependent, thus their definition cannot be performed without considering the design process as a whole thing. Unfortunately, Design Space of WSNs is extremely complex and huge, thus an exhaustive Design Space Exploration cannot be employed. Moreover, considering that its size increases exponentially as the number of nodes involved increases, efficient DSE techniques are required as well as effective CAD tools.

Wireless Sensor Networks are becoming a mature field for research, with dozens of platforms, network protocols and interfaces, several routing techniques and energy saving policies, operating systems, and so on. At that moment, WSN design is mainly performed manually, leaving the user to decide which protocol best fit its needs, which hardware platform should be used, how nodes should be programmed, including the definition and configuration of the operating system. The identification of the right combination of these things is extremely hard and could lead to non-optimal or non-working devices if performed manually. Considering that a good design of a WSN require knowledge in the field of telecommunication, computer science and electronic as well as problem-specific notions (i.e. in medics, geophysics, natural sciences, etc..), multi-disciplinary teams must be created, since a single person does not own all the required skills. Moreover, manual WSN design could leave the designer to *use a technology since he knows how to use it,* that is not acceptable for strong constrained networks.

This thesis presented a comprehensive study about methodologies and techniques for Wireless Sensor Network design. This thesis introduced a complete and generic design flow for Wireless Sensor Networks as well as innovative methodologies and techniques to perform specific design processes. The thesis started by introducing some basic concept about WSNs and Design Space Exploration (DSE) such as models or exploration techniques. Then it presented a review and comparison of commercial microcontroller-based platforms and a novel FPGA-based platform. The fourth Chapter is the core of the work, and introduced a general-purpose design flow as well as innovative model-based and hybrid exploration techniques such as the promising Markov Decision Process (MDP). The fifth Chapter presented two online techniques to: reduce single-channel interferences in IEEE 802.15.4 based networks and increase

energy efficiency in spatially-correlated cluster networks. Lastly, Chapter six presented a complete design example to show how to effectively use the proposed design flow in a real-world example.

This thesis introduced several innovations:

- A general-purpose **design flow** for Wireless Sensor Networks. To the best of my knowledge, this is the first design flow specifically designed for Wireless Sensor Networks;
- A comparative study between Flash and SRAM FPGAs and the **design of a novel FPGA-based WSN node** for high throughput networks with strong energy requirements;
- Two **models** for model-based DSE created for **Wireless Body Sensor Network** field;
- An innovative **hybrid DSE algorithm** based on the **Markov Decision Process (MDP)** and effectively applied to IEEE 802.15.4 networks;
- An innovative **technique to reduce BAN-BAN interference** in IEEE 802.15.4 based Body Area Networks;
- An innovative **technique to reduce transmissions** in cluster-based networks using **spatial correlation** information.

Future Works and Promising Trends

Optimal design of Wireless Sensor Networks is complex, due to the singularity of their applications and the large number of available hardware, network and software architectures. In a very near future, more and more applications and devices will come into the market, thus quick and effective CAD tools will be required. However, these tools should not deploy the design into the target network, rather they must support the user in all the aspects related to the design of WSNs. At the same time, they must drive the design process by evaluating and comparing alternative implementations.

Future works include:

- **Development of design frameworks**: the framework developed in this thesis was designed only to verify and test the proposed methodologies and techniques but it is not able to deploy the final implementation. Future developers should create platform-independent CAD tools able to target specific platforms; they should include: accurate models of target platforms, compilers, custom configuration tools, etc...;
- **Development of accurate and efficient simulators**: simulators provide a good compromise between accuracy and evaluation speed. To design efficient and reliable sensor networks, better simulators must be developed. They should include: realistic models of real commercial hardware platforms, routing protocols, MAC protocols, OS emulation, obstacle and environment simulation, etc...;

- **Development of both general-purpose and problem-specific DSE tools**: DSE tools are extremely important to support the designer during the identification of the optimal solutions, thus both general-purpose (less effective but reusable) and problem-specific (more effective but less reusable) must be developed.

WSNs have been widely studied in the last decade, and several innovative applications are being envisioned and implemented. This thesis could affect and influence several application fields including (but not limited to):

- **Smart Building and Smart Cities**: this is an extremely hot topic today: *"How to make our buildings and cities smarter? How to improve energy efficiency of our cities without reducing their comfort? etc..."*. Despite application specific algorithm and techniques to solve these problems, it is evident that, once effective solutions will be identified, their implementation could not be trivial. Imagine a building with hundreds of sensor/actuator wireless devices interconnected to monitor and control energy, light and security, or offering various media services. Design techniques and tools will decrease the time-to-market, increase the quality and reliability of the system, and help the designer to test and compare various design alternatives before the final deployment;

- **Wireless Body Sensor Networks**: these systems do not have a large number of devices, but energy and performance constraints make their design extremely difficult. For that reason, very often, the hardware is designed ad-hoc for the specific application, and that design must be verified with network protocols and operating system. Wireless Body Sensor Networks are usually employed in the medical field, thus reliability must be guaranteed over any reasonable optimization. Similarly to Smart Buildings, efficient CAD tools will decrease the time-to-market, increase the performance and lifetime of the system, and help the designer to verify and test its algorithms and hardware platforms;

- **Environmental Monitoring**: this was one of the first application of WSNs. It includes the deployment of a WSN in hostile places to gather environmental data periodically in order to monitor and control the area of interest. These type of WSNs can be used to monitor an inhabitated area to check if catastrofic events are occurring (fire), or just to gather information about the habits of the local fauna. An example of design of these networks has been given in the previous chapter. Although these types of networks do not have strong reliability and safety constraints, their management is extremely difficult, thus an optimal design is required to reduce maintainance costs. In this application field, lifetime of the network should be typically be higher than three months;

- **Multimedia Sensor Networks**: multimedia sensor networks are characterized by extremely high throughputs and high computation requirements. They are used for security or entertainment applications with strong performance and cost requirements. An effective definition of hardware plaforms as well as network protocols is required to guarantee throughtput requirements. Similarly to the other fields, verification and testing of these systems extremely important and could make the difference between a winning or a failure product.

Bibliography

[1] Avrora: The AVR simulation and analysis framework (last visit on 9th september 2012). http://compilers.cs.ucla.edu/avrora/ (last visit on 9th September 2012).

[2] The FreeRTOS project. http://www.freertos.org (last visit on 9th September 2012).

[3] INDRIYA: a wireless sensor network testbed. http://indriya.comp.nus.edu.sg (last visit on 9th September 2012).

[4] ISIS - JProwler. http://w3.isis.vanderbilt.edu/projects/nest/jprowler/index.html (last visit on 9th September 2012).

[5] MiXiM project. http://mixim.sourceforge.net/ (last visit on 9th September 2012).

[6] MOMH: multiple objective meta heuristics. http://home.gna.org/momh/ (last visit on 9th September 2012).

[7] MoteLAB. http://motelab.eecs.harvard.edu/ (last visit on 9th September 2012).

[8] NS-2. http://nsnam.isi.edu/nsnam/index.php/User_Information (last visit on 9th September 2012).

[9] OMNET++ castalia simulator. http://castalia.npc.nicta.com.au/ (last visit on 9th September 2012).

[10] SENS: a sensor, environment and network simulator. http://osl.cs.uiuc.edu/sens/ (last visit on 9th September 2012).

[11] Senslab: Very large scale open wireless sensor network testbed. http://www.senslab.info/ (last visit on 9th September 2012).

[12] TKN wireless indoor sensor network testbed. http://www.twist.tu-berlin.de (last visit on 9th September 2012).

[13] UWSim: the UnderWater simulator. http://www.irs.uji.es/uwsim/ (last visit on 9th September 2012).

[14] VisualSense: visual editor and simulator for wireless sensor network systems. http://ptolemy.eecs.berkeley.edu/visualsense/ (last visit on 9th September 2012).

[15] WISENES simulator. http://www.tkt.cs.tut.fi/research/daci/daci_wsn_wisenes.html (last visit on 9th September 2012).

[16] The WUSTL wireless sensor network testbed. http://wsn.cse.wustl.edu/index.php/The_WUSTL_Wireless_Sensor_Network_Testbed (last visit on 9th September 2012).

[17] 21 ideas for the 21st century. pages 78–167, Aug. 1999.

[18] IEEE standard for information technology. 802.15.4 standard specification, 2006.

[19] A. A. Abbasi and M. Younis. A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, 30(14-15):2826–2841, 2007.

[20] Actel. Core8051 - direct core datasheet.

[21] Actel. IGLOO low power flash FPGAs datasheet.

[22] Actel. Actel's Flash*Freeze technology and low power modes, 2008.

[23] Actel. Competitive programmable logic power comparison (white paper), 2008.

[24] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325–349, May 2005.

[25] I. F. Akyildiz, D. Pompili, and T. Melodia. Underwater acoustic sensor networks: research challenges. *Ad Hoc Networks*, 3(3):257–279, May 2005.

[26] M. I. Ali, B. M. Al-Hashimi, J. Recas, and D. Atienza. Evaluation and design exploration of solar harvested-energy prediction algorithm. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '10, pages 142–147, Dresden, Germany, 2010. European Design and Automation Association.

[27] C. Alippi, R. Camplani, C. Galperti, A. Marullo, and M. Roveri. An hybrid wireless-wired monitoring system for real-time rock collapse forecasting. In *2010 IEEE 7th International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 224–231. IEEE, Nov. 2010.

[28] O. Aziz, B. Lo, R. King, A. Darzi, and G.-Z. Yang. Pervasive body sensor network: an approach to monitoring the post-operative surgical patient. In *Wearable and Implantable Body Sensor Networks (BSN)*, pages 4–18. IEEE, Apr. 2006.

[29] L. S. Bai, R. P. Dick, P. H. Chou, and P. A. Dinda. Automated construction of fast and accurate system-level models for wireless sensor networks. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pages 1–6. IEEE, Mar. 2011.

[30] S. Banerjee and S. Khuller. A clustering scheme for hierarchical control in multi-hop wireless networks. In *IEEE INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, volume 2, pages 1028–1037 vol.2. IEEE, 2001.

[31] H. Belhadj, V. Aggrawal, A. Pradhan, and A. Zerrouki. Power-aware FPGA design, 2009.

[32] G. Beltrame, L. Fossati, and D. Sciuto. Decision-theoretic design space exploration of multiprocessor platforms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(7):1083–1095, July 2010.

[33] R. Benzid, F. Marir, A. Boussaad, M. Benyoucef, and D. Arar. Fixed percentage of wavelet coefficients to be zeroed for ECG compression. *Electronics Letters*, 39(11):830 – 831, May 2003.

[34] A. P. Bhondekar, R. Vig, M. L. Singla, C. Ghanshyam, and P. Kapur. Genetic algorithm based node placement methodology for wireless sensor networks. *Computer*, I:7, 2009.

[35] B. W. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5):61 –72, May 1988.

[36] A. Bonivento, L. P. Carloni, and A. Sangiovanni-Vincentelli. Platform based design for wireless sensor networks. *Mob. Netw. Appl.*, 11(4):469–485, Aug. 2006.

[37] A. Boukerche. *Algorithms and Protocols for Wireless Sensor Networks*. Wiley-IEEE Press, 2008.

[38] A. Boulis. Castalia: revealing pitfalls in designing distributed algorithms in WSN. In *Embedded networked sensor systems*, pages 407–408, New York, NY, USA, 2007. ACM.

[39] A. Boulis. Castalia, a simulator for wireless sensor networks and body area networks, user's manual (Version 3.2), Mar. 2011.

[40] C. Buratti. A mathematical model for performance of IEEE 802.15.4 beacon-enabled mode. In *Proceedings of the International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, IWCMC '09, pages 1184–1190, New York, NY, USA, 2009. ACM.

[41] T. Burd and R. Brodersen. Energy efficient CMOS microprocessor design. In *System Sciences, 1995. Proceedings of the Twenty-Eighth Hawaii International Conference on*, volume 1, pages 288 –297 vol.1, Jan. 1995.

[42] A. Burns, B. R. Greene, M. J. McGrath, T. J. O'Shea, B. Kuris, S. M. Ayer, F. Stroiescu, and V. Cionca. SHIMMER - a wireless sensor platform for non-invasive biomedical research. *IEEE Sensors Journal*, 10(9):1527–1534, Sept. 2010.

[43] E. Casilari, J. M. Cano-García, and G. Campos-Garrido. Modeling of current consumption in 802.15.4/ZigBee sensor motes. *Sensors*, 10(6):5443–5468, June 2010.

[44] B. G. Celler, T. Hesketh, W. Earnshaw, and E. Ilsar. An instrumentation system for the remote monitoring of changes in functional health status of the elderly at

home. In *Engineering in Medicine and Biology Society*, pages 908–909 vol.2. IEEE, 1994.

[45] J.-H. Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. *Networking, IEEE/ACM Transactions on*, 12(4):609 – 619, Aug. 2004.

[46] M. Chen, S. Gonzalez, A. Vasilakos, H. Cao, and V. C. Leung. Body area networks: A survey. *Mobile Networks and Applications*, 16(2):171–193, Apr. 2011.

[47] W. Choi and M. Woo. A distributed weighted clustering algorithm for mobile ad hoc networks. In *International Conference on Internet and Web Applications and Services/Advanced International Conference on Telecommunications, 2006. AICT-ICIW '06*. IEEE, Feb. 2006.

[48] C.-Y. Chong and S. P. Kumar. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247– 1256, Aug. 2003.

[49] T. Cover. A proof of the data compression theorem of slepian and wolf for ergodic sources (Corresp.). *IEEE Transactions on Information Theory*, 21(2):226–228, Mar. 1975.

[50] B. de Silva, A. Natarajan, and M. Motani. Inter-user interference in body sensor networks: Preliminary investigation and an infrastructure-based solution. In *Wearable and Implantable Body Sensor Networks (BSN)*, pages 35–40. IEEE, June 2009.

[51] Digi. XBee and XBee-PRO DigiMesh 2.4 embedded RF modules.

[52] E. Duarte-Melo and M. Liu. Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks. In *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, volume 1, pages 21 – 25 vol.1, Nov. 2002.

[53] E. K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J. C. Burgelman. ISTAG - scenarios for ambient intelligence in 2010. *Office for Official Publications of the European Communities*, 2001.

[54] L. M. Feeney. An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks. *Mobile Networks and Applications*, 6(3):239–249, 2001.

[55] S. Fekete, A. Kroller, S. Fischer, and D. Pfisterer. Shawn: The fast, highly customizable sensor network simulator. In *Fourth International Conference on Networked Sensing Systems, 2007. INSS '07*, page 299, June 2007.

[56] K. P. Ferentinos and T. A. Tsiligiridis. Adaptive design optimization of wireless sensor networks using genetic algorithms. *Computer Networks*, 51(4):1031–1051, Mar. 2007.

[57] J. Ferrante, K. J. Ottenstein, and J. D. Warren. The program dependence graph and its use in optimization. *ACM Transactions on Programming Languages and Systems*, 9(3):319–349, July 1987.

[58] H. Ghasemzadeh, R. Jafari, and B. Prabhakaran. A body sensor network with electromyogram and inertial sensors: Multimodal interpretation of muscular activities. *IEEE Transactions on Information Technology in Biomedicine*, 14(2):198 –206, Mar. 2010.

[59] L. Girod, N. Ramanathan, J. Elson, T. Stathopoulos, M. Lukac, and D. Estrin. Emstar: A software environment for developing and deploying heterogeneous sensor-actuator networks. *ACM Trans. Sen. Netw.*, 3(3), Aug. 2007.

[60] A. J. Goldsmith and S. B. Wicker. Design challenges for energy-constrained ad hoc wireless networks. *IEEE Wireless Communications*, 9(4):8– 27, Aug. 2002.

[61] O. Goussevskaia, R. Wattenhofer, M. M. Halldorsson, and E. Welzl. Capacity of arbitrary wireless networks. In *IEEE INFOCOM 2009*, pages 1872–1880. IEEE, Apr. 2009.

[62] P. R. Grassi, I. Beretta, V. Rana, D. Atienza, and D. Sciuto. Knowledge-based design space exploration of wireless sensor networks. Tampere, Finland, Oct. 2012.

[63] P. R. Grassi, A. Ceppi, F. Cancarè, G. Ravazzani, M. Mancini, and D. Sciuto. Automatic identification and placement of measurement stations for hydrological discharge simulations at basin scale. volume 14, page 5826, Apr. 2012.

[64] V. Gutnik and A. Chandrakasan. Embedded power supply for low-power DSP. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 5(4):425 –435, Dec. 1997.

[65] P. S. Hall and Y. Hao. Antennas and propagation for body centric communications. In *First European Conference on Antennas and Propagation, 2006. EuCAP 2006*, pages 1 –7, Nov. 2006.

[66] X. Han, X. Cao, E. L. Lloyd, and C.-C. Shen. Fault-tolerant relay node placement in heterogeneous wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9(5):643–656, May 2010.

[67] J. He, Z. Tang, H.-H. Chen, and Q. Zhang. An accurate and scalable analytical model for IEEE 802.15.4 slotted CSMA/CA networks. *IEEE Transactions on Wireless Communications*, 8(1):440–448, Jan. 2009.

[68] Z. He, J. Eggert, W. Cheng, X. Zhao, J. Millspaugh, R. Moll, J. Beringer, and J. Sartwell. Energy-aware portable video communication system design for wildlife activity monitoring. *Circuits and Systems Magazine, IEEE*, 8(2):25 – 37, 2008.

[69] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660– 670, Oct. 2002.

[70] J. L. Hennessy and D. A. Patterson. Computer architecture, fourth edition: A quantitative approach. Sept. 2006.

[71] T. P. Hong, G. N. Shiu, and Y. C. Lee. A PSO heuristic algorithm for basestation locations. Japan, 2006.

[72] N. Horton. Commercial wireless sensor networks: Status, issues and challenges. Santa Clara, California, Oct. 2004.

[73] W. Hu, V. N. Tran, N. Bulusu, C. T. Chou, S. Jha, and A. Taylor. The design and evaluation of a hybrid sensor network for cane-toad monitoring. In *Fourth International Symposium on Information Processing in Sensor Networks, 2005. IPSN 2005*, pages 503– 508. IEEE, Apr. 2005.

[74] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1):2– 16, Feb. 2003.

[75] E. Jovanov and A. Milenkovic. Body area networks for ubiquitous healthcare applications: Opportunities and challenges. *Journal of Medical Systems*, 35(5):1245–1254, Feb. 2011.

[76] E. Jovanov, A. Milenkovic, C. Otto, and P. C. d. Groen. A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. *Journal of NeuroEngineering and Rehabilitation*, 2(1):6, Mar. 2005.

[77] K. Kavi, B. Buckles, and U. Bhat. A formal definition of data flow graph models. *IEEE Transactions on Computers*, C-35(11):940 –948, Nov. 1986.

[78] I. Khan, Y. I. Nechayev, K. Ghanem, and P. S. Hall. BAN-BAN interference rejection with multiple antennas at the receiver. *IEEE Transactions on Antennas and Propagation*, 58(3):927–934, Mar. 2010.

[79] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *6th International Symposium on Information Processing in Sensor Networks, 2007. IPSN 2007*, pages 254 –263, Apr. 2007.

[80] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[81] H. Koc, O. Ozturk, M. Kandemir, and E. Ercanli. Minimizing energy consumption of banked memories using data recomputation. In *Low Power Electronics and Design, 2006. ISLPED'06. Proceedings of the 2006 International Symposium on*, pages 358 –361, Oct. 2006.

[82] M. Kohvakka, M. Kuorilehto, M. Hannikainen, and T. D. Hamalainen. Performance analysis of IEEE 802.15.4 and ZigBee for large-scale wireless sensor network applications. In *Performance evaluation of wireless ad hoc, sensor and ubiquitous networks*, pages 48–57, New York, NY, USA, 2006. ACM.

[83] A. Koubaa, M. Alves, and E. Tovar. GTS allocation analysis in IEEE 802.15.4 for real-time wireless sensor networks. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, Apr. 2006.

[84] B. Krishnamachari, D. Estrin, and S. Wicker. Modelling data-centric routing in wireless sensor networks. New York, USA, June 2002.

[85] B. Krishnamachari and S. Iyengar. Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *IEEE Transactions on Computers*, 53(3):241– 250, Mar. 2004.

[86] L. Krishnamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *22nd International Conference on Distributed Computing Systems Workshops, 2002. Proceedings*, pages 575– 578. IEEE, 2002.

[87] D. Kumar, T. C. Aseri, and R. Patel. EEHC: energy efficient heterogeneous clustered scheme for wireless sensor networks. *Computer Communications*, 32(4):662–667, Mar. 2009.

[88] G. Kumar, G. Manimaran, and Z. Wang. End-to-end energy management in networked real-time embedded systems. *Parallel and Distributed Systems, IEEE Transactions on*, 19(11):1498–1510, Nov. 2008.

[89] M. Kuorilehto, M. Hännikäinen, and T. D. Hämäläinen. Rapid design and evaluation framework for wireless sensor networks. *Ad Hoc Networks*, 6(6):909–935, Aug. 2008.

[90] M. Kuorilehto, M. Kohvakka, M. Hännikäinen, and T. D. Hämäläinen. High abstraction level design and implementation framework for wireless sensor networks. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, T. D. Hämäläinen, A. D. Pimentel, J. Takala, and S. Vassiliadis, editors, *Embedded Computer Systems: Architectures, Modeling, and Simulation*, volume 3553, pages 384–393. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[91] B. Latre, B. Braem, I. Moerman, C. Blondia, and P. Demeester. A survey on wireless body area networks. *Wireless Networks*, 17(1):1–18, Nov. 2010.

[92] Lattice. LatticeXP family data sheet, 2007.

[93] U. Lee, B. Zhou, M. Gerla, E. Magistretti, P. Bellavista, and A. Corradi. Mobeyes: smart mobs for urban monitoring with a vehicular sensor network. *IEEE Wireless Communications*, 13(5):52 –57, Oct. 2006.

[94] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, page 126–137, New York, NY, USA, 2003. ACM.

[95] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *Mobile computing and networking*, pages 61–69, Rome, Italy, 2001. ACM.

[96] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. In P. Dini, P. Lorenz, and J. de Souza, editors, *Service Assurance with Partial and Intermittent Resources*, volume 3126 of *Lecture Notes in Computer Science*, pages 239–254.

[97] S. Lindsey and C. S. Raghavendra. PEGASIS: power-efficient gathering in sensor information systems. In *IEEE Aerospace Conference Proceedings, 2002*, volume 3, pages 3–1125– 3–1130 vol.3. IEEE, 2002.

[98] C. Liu, K. Wu, and J. Pei. An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation. *IEEE Transactions on Parallel and Distributed Systems*, 18(7):1010–1023, July 2007.

[99] Y. Liu, W. Zhang, and K. Akkaya. Static worst-case energy and lifetime estimation of wireless sensor networks. In *Performance Computing and Communications Conference (IPCCC), 2009 IEEE 28th International*, pages 17 –24, Dec. 2009.

[100] L. Lo Bello and E. Toscano. Coexistence issues of multiple co-located IEEE 802.15.4/ZigBee networks running on adjacent radio channels in industrial environments. *IEEE Transactions on Industrial Informatics*, 5(2):157–167, May 2009.

[101] K. Lorincz, B.-r. Chen, G. W. Challen, A. R. Chowdhury, S. Patel, P. Bonato, and M. Welsh. Mercury: A wearable sensor network platform for high-fidelity motion analysis. *Embedded Networked Sensor Systems*, pages 183–196, 2009.

[102] G. Lu, B. Krishnamachari, and C. S. Raghavendra. Performance evaluation of the IEEE 802.15.4 MAC for low-rate low-power wireless networks. In *2004 IEEE International Conference on Performance, Computing, and Communications*, pages 701– 706. IEEE, 2004.

[103] S. Luke. *Essentials of Metaheuristics*. Lulu, 2009.

[104] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Wireless sensor networks and applications*, pages 88–97, Atlanta, Georgia, USA, 2002. ACM.

[105] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst. Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes. *Biomedical Engineering, IEEE Transactions on*, 58(9):2456 –2466, Sept. 2011.

[106] A. Manjeshwar and D. P. Agrawal. TEEN: a routing protocol for enhanced efficiency in wireless sensor networks. In *Parallel and Distributed Processing Symposium, International*, volume 3, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[107] A. Manjeshwar and D. P. Agrawal. APTEEN: a hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. In *Parallel and Distributed Processing Symposium, International*, volume 2, page 0195b, Los Alamitos, CA, USA, 2002. IEEE Computer Society.

[108] K. Martinez, R. Ong, and J. Hart. Glacsweb: a sensor network for hostile environments. In *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004*, pages 81– 87. IEEE, Oct. 2004.

[109] C. . Mathúna, T. O'Donnell, R. V. Martinez-Catala, J. Rohan, and B. O'Flynn. Energy scavenging for long-term deployable wireless sensor networks. *Talanta*, 75(3):613–623, May 2008.

[110] A. McGibney, A. Gurnard, and D. Pesch. Wi-design: A modelling and optimization tool for wireless embedded systems in buildings. In *2011 IEEE 36th Conference on Local Computer Networks (LCN)*, pages 640–648. IEEE, Oct. 2011.

[111] K. Mechitov, W. Kim, G. Agha, and T. Nagayama. High-frequency distributed sensing for structure monitoring. In *in Proc. First Intl. Workshop on Networked Sensing Systems (INSS 04*, 2004.

[112] E. Meshkova, J. Riihijarvi, A. Achtzehn, and P. Mahonen. Exploring simulated annealing and graphical models for optimization in cognitive wireless networks. In *IEEE Global Telecommunications Conference, 2009. GLOBECOM 2009*, pages 1–8. IEEE, Dec. 2009.

[113] S. Misra, S. D. Hong, G. Xue, and J. Tang. Constrained relay node placement in wireless sensor networks to meet connectivity and survivability requirements. In *IEEE INFOCOM 2008. The 27th Conference on Computer Communications*, pages 281–285. IEEE, Apr. 2008.

[114] M. Motani, V. Srinivasan, and P. S. Nuggehalli. PeopleNet: engineering a wireless virtual social network. In *Proceedings of the 11th annual international conference on Mobile computing and networking*, MobiCom '05, page 243–257, New York, NY, USA, 2005. ACM.

[115] L. Mottola and G. P. Picco. Programming wireless sensor networks: Fundamental concepts and state of the art. *ACM Comput. Surv.*, 43(3):19:1–19:51, Apr. 2011.

[116] P. Muralidhar and C. B. Rao. Reconfigurable wireless sensor network node based on nios core. In *Fourth International Conference on Wireless Communication and Sensor Networks, 2008. WCSN 2008*, pages 67–72. IEEE, Dec. 2008.

[117] R. Musaloiu-E. and A. Terzis. Minimising the effect of WiFi interference in 802.15.4 wireless sensor networks. *International Journal of Sensor Networks*, 3(1):43–54, Jan. 2008.

[118] S. Nabar, J. Walling, and R. Poovendran. Minimizing energy consumption in body sensor networks via convex optimization. In *Body Sensor Networks (BSN), Int. Conf. on*, pages 62 –67, June 2010.

[119] M. Nabi, M. Blagojevic, T. Basten, M. Geilen, and T. Hendriks. Configuring multi-objective evolutionary algorithms for design-space exploration of wireless

sensor networks. In *Proceedings of the 4th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pages 111–119, Tenerife, Canary Islands, Spain, 2009. ACM.

[120] R. Nagpal and D. Coore. An algorithm for group formation in an amorphous computer. *Proceedings of the Tenth International Conference on Parallel and Distributed Systems*, 1998.

[121] D. Nam and C. H. Park. Multiobjective simulated annealing: A comparative study to evolutionary algorithms. *International Journal of Fuzzy Systems*, 2(2):87–97, 2000.

[122] S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. *ACM Trans. Sen. Netw.*, 4(2):7:1–7:40, Apr. 2008.

[123] D. Navarro, F. Mieyeville, W. Du, M. Galos, and I. O'Connor. Towards a design framework for heterogeneous wireless sensor networks. In *2011 1st International Symposium on Access Spaces (ISAS)*, pages 83–88. IEEE, June 2011.

[124] T. Okabe, Y. Jin, and B. Sendhoff. A critical survey of performance indices for multi-objective optimisation. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03*, volume 2, pages 878– 885 Vol.2. IEEE, Dec. 2003.

[125] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with COOJA. In *Proceedings 2006 31st IEEE Conference on Local Computer Networks*, pages 641 –648, Nov. 2006.

[126] G. Palermo, C. Silvano, and V. Zaccaria. ReSPIR: a response surface-based pareto iterative refinement for application-specific design space exploration. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(12):1816 –1829, Dec. 2009.

[127] A. Pantelopoulos and N. G. Bourbakis. A survey on wearable sensor-based systems for health monitoring and prognosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40(1):1–12, Jan. 2010.

[128] P. Park, C. Fischione, and K. Johansson. Performance analysis of GTS allocation in beacon enabled IEEE 802.15.4. In *6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09*, pages 1 –9, June 2009.

[129] P. Park, C. Fischione, and K. H. Johansson. Performance analysis of GTS allocation in beacon enabled IEEE 802.15.4. In *6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09*, pages 1–9. IEEE, June 2009.

[130] M. A. Pasha, S. Derrien, and O. Sentieys. System-level synthesis for wireless sensor node controllers. *ACM Transactions on Design Automation of Electronic Systems*, 17(1):1–24, Jan. 2012.

[131] M. Patel and J. Wang. Applications, challenges, and prospective in emerging body area networking technologies. *Wireless Communications, IEEE*, 17(1):80–88, Feb. 2010.

[132] S. Pattem, B. Krishnamachari, and R. Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. page 28. ACM Press, 2004.

[133] B. Paul, M. A. Matin, M. J. Showkat, and Z. Rahman. Optimal placement of base stations in a two tiered wireless sensor network. *WTOC*, 9(1):43–52, Jan. 2010.

[134] F. Philipp and M. Glesner. Mechanisms and architecture for the dynamic reconfiguration of an advanced wireless sensor node. In *2011 International Conference on Field Programmable Logic and Applications (FPL)*, pages 396–398. IEEE, Sept. 2011.

[135] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 95–107, New York, NY, USA, 2004. ACM.

[136] S. Pollin, M. Ergen, M. Timmers, A. Dejonghe, L. van der Perre, F. Catthoor, I. Moerman, and A. Bahai. Distributed cognitive coexistence of 802.15.4 with 802.11. In *Cognitive Radio Oriented Wireless Networks and Communications*, pages 1–5. IEEE, June 2006.

[137] D. Pompili, T. Melodia, and I. F. Akyildiz. Deployment analysis in underwater acoustic wireless sensor networks. In *Proceedings of the 1st ACM international workshop on Underwater networks*, WUWNet '06, pages 48–55, New York, NY, USA, 2006. ACM.

[138] F. Rincon, P. R. Grassi, N. Kahled, D. Atienza, and D. Sciuto. Automated real-time atrial fibrillation detection on a wearable wireless sensor platform. 2012.

[139] K. Romer and F. Mattern. The design space of wireless sensor networks. *IEEE Wireless Communications*, 11(6):54– 61, Dec. 2004.

[140] S. Russel and P. Norvig. *Artificial Intelligence: Modern Approach*. 1st ed. englewood cliffs, NJ: prentice hall edition, 1995.

[141] I. Sanchez-Tato, J. C. Senciales, J. Salinas, L. Fanucci, G. Pardini, F. Costalli, S. Dalmiani, J. M. de la Higuera, Z. Vukovic, and Z. Cicigoj. Health @ home: A telecare system for patients with chronic heart failure. In *2010 Fifth International Conference on Broadband and Biomedical Communications (IB2Com)*, pages 1–5. IEEE, Dec. 2010.

[142] C. Schurgers, V. Raghunathan, and M. B. Srivastava. Power management for energy-aware communication systems. *ACM Trans. Embed. Comput. Syst.*, 2(3):431–447, Aug. 2003.

[143] S. Y. Shin, H. S. Park, and W. H. Kwon. Mutual interference analysis of IEEE 802.15.4 and IEEE 802.11b. *Computer Networks*, 51(12):3338–3353, Aug. 2007.

[144] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 1–12, Baltimore, MD, USA, 2004. ACM.

[145] M. Srivastava, R. Muntz, and M. Potkonjak. Smart kindergarten: sensor-based wireless networks for smart developmental problem-solving environments. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, pages 132–138, Rome, Italy, 2001. ACM.

[146] P. Suarez, C.-G. Renmarker, A. Dunkels, and T. Voigt. Increasing ZigBee network lifetime with x-MAC. In *Proceedings of the workshop on Real-world wireless sensor networks*, REALWSN '08, pages 26–30, New York, NY, USA, 2008. ACM.

[147] Y. Sun, L. Li, and H. Luo. Design of FPGA-Based multimedia node for WSN. In *2011 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, pages 1–5. IEEE, Sept. 2011.

[148] H. Sundani, H. Li, V. Devabhaktuni, M. Alam, and P. Bhattacharya. Wireless sensor network simulators: A survey and comparisons. *International Journal of Computer Networks (IJCN)*, 2(6):249, Feb. 2011.

[149] C. Tachtatzis, F. Di Franco, D. Tracey, N. Timmons, and J. Morrison. An energy analysis of IEEE 802.15.6 scheduled access modes. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 1270 –1275, Dec. 2010.

[150] J. Tang, B. Hao, and A. Sen. Relay node placement in large scale wireless sensor networks. *Computer Communications*, 29(4):490–501, Feb. 2006.

[151] N. F. Timmons and W. G. Scanlon. Analysis of the performance of IEEE 802.15.4 for medical sensor body area networking. pages 16– 24. IEEE, Oct. 2004.

[152] W. Tsujita, A. Yoshino, H. Ishida, and T. Moriizumi. Gas sensor network for air-pollution monitoring. *Sensors and Actuators B: Chemical*, 110(2):304–311, Oct. 2005.

[153] S. Ullah, H. Higgins, B. Braem, B. Latre, C. Blondia, I. Moerman, S. Saleem, Z. Rahman, and K. Kwak. A comprehensive survey of wireless body area networks. *Journal of Medical Systems*, 36(3):1065–1094, 2012.

[154] S. Ullah and K. S. Kwak. Throughput and delay limits of IEEE 802.15.6. In *Wireless Communications and Networking Conference (WCNC), 2011 IEEE*, pages 174 –178, Mar. 2011.

[155] A. Varga. The OMNET++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference*, pages 319–324. SCS – European Publishing House, 2001.

[156] H. Viittala, M. Hamalainen, J. Iinatti, and A. Taparugssanagorn. Different experimental WBAN channel models and IEEE802.15.6 models: Comparison and effects. In *Applied Sciences in Biomedical and Communication Technologies, 2009. ISABEL 2009. 2nd International Symposium on*, pages 1 –5, Nov. 2009.

[157] L. Villas, A. Boukerche, H. A. de Oliveira, R. B. de Araujo, and A. A. Loureiro. A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks. *Ad Hoc Networks*, In Press, Accepted Manuscript, 2011.

[158] M. C. Vuran and I. F. Akyildiz. Spatial correlation-based collaborative medium access control in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 14(2):316– 329, Apr. 2006.

[159] Y. Wang. Topology control for wireless sensor networks. In Y. Li, M. T. Thai, and W. Wu, editors, *Wireless Sensor Networks and Applications*, Signals and Communication Technology, pages 113–147. Springer US, 2008.

[160] D. West. *Introduction to Graph Theory (2nd Edition)*. Prentice Hall, 2000.

[161] G. Wu, J. Ren, F. Xia, L. Yao, and Z. Xu. DISG: decentralized inter-user interference suppression in body sensor networks with non-cooperative game. In *Ubiquitous Intelligence & Computing and Autonomic & Trusted Computing (UIC/ATC)*, pages 256–261. IEEE, Oct. 2010.

[162] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *IEEE INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, volume 3, pages 1567– 1576 vol.3. IEEE, 2002.

[163] C.-t. Yeh, Z. Fan, and R. Gao. Energy-aware data acquisition in wireless sensor networks. In *Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE*, pages 1 –6, May 2007.

[164] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, 2008.

[165] S. Yoon and C. Shahabi. Exploiting spatial correlation towards an energy efficient clustered aggregation technique (CAG) [wireless sensor network applications]. In *2005 IEEE International Conference on Communications, 2005. ICC 2005*, volume 5, pages 3307– 3313 Vol. 5. IEEE, May 2005.

[166] M. Younis and K. Akkaya. Strategies and techniques for node placement in wireless sensor networks: A survey. *Ad Hoc Networks*, 6(4):621–655, June 2008.

[167] C. H. Zhiyong, L. Y. Pan, Z. Zeng, and M. Q.-H. Meng. A novel FPGA-based wireless vision sensor node. In *IEEE International Conference on Automation and Logistics, 2009. ICAL '09*, pages 841–846. IEEE, Aug. 2009.

[168] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, MobiSys '04, pages 125–138, New York, NY, USA, 2004. ACM.

[169] Y. Zou and K. Chakrabarty. Sensor deployment and target localization based on virtual forces. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 1293– 1303 vol.2. IEEE, Apr. 2003.

[170] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004*, pages 517– 526. IEEE, Oct. 2004.