Platform Based Design for Wireless Sensor Networks

Alvise Bonivento

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-85
http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-85.html

June 20, 2007

Platform Based Design for Wireless Sensor Networks

by

Alvise Bonivento

Laurea (University of Padua, Italy) 2002
M.S. (University of California, Berkeley) 2004

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Science

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Alberto Sangiovanni-Vincentelli, Chair
Professor Jan Rabaey
Professor Paul Wright

Fall 2007

The dissertation of Alvise Bonivento is approved.

Chair Date

Date

Date

University of California, Berkeley

Fall 2007

Platform Based Design for Wireless Sensor Networks

by

Alvise Bonivento

Abstract

Platform Based Design for Wireless Sensor Networks

by

Alvise Bonivento

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Science

University of California, Berkeley

Professor Alberto Sangiovanni-Vincentelli, Chair

The increasing complexity, heterogeneity and reliability requirements of wireless sensor networks is posing major challenges to the capability of developing effective designs. The lack of a system level approach is significantly slowing the adoption of this technology and limiting it to marginal markets.

This Dissertation proposes a new methodology for the system level design of wireless sensor networks. This methodology is based on the Platform Based Design (PBD) methodology that was originally developed for classical embedded systems, and it is here revisited and applied to the wireless sensor networks domain. According to PBD, a design is obtained as a sequence of refining steps that take guide the designer from the initial specification all the way down to a physical implementation. To support this process, a set of intermediate abstraction layers and platforms are identified.

When applying this methodology to wireless sensor networks, three layers of abstraction and relative platforms are identified: a service platform at the applciation layer, a protocol platform to describe the protocol stacks, and an implementation

platform for the hardware nodes. Two different strategies to refine applications into implementations using these three platforms are proposed. This differentiation of the mapping strategies is essential to accomodate different application categories. The proposed methodology is validated using case studies from building monitoring and industrial monitoring applications.

Professor Alberto Sangiovanni-Vincentelli
Dissertation Committee Chair

To my family

Contents

List of Figures

List of Tables

Acknowledgements

Getting a PhD is not only an academic achievement, but it is most of all a completion of a maturation process both at the technical and personal level, and it is very difficult to distinguish these two aspects. There are many things that contribute to this process, some of which I learned at Cory Hall, some important others I learned during five years of day to day life in this wonderful and diverse environment called Berkeley. With this perspective, there are many people that impact my work in these five years and that I am grateful to.

First and foremost, I would like to thank my adviser Professor Alberto Sangiovanni Vincentelli for the great opportunity that he offered me by supporting my graduate student career at Berkeley since the application process. Naming his worldwide known qualities would take hundreds of pages. His great vision and intuition inspired most of my work, and the continuous feedback, suggestions, support during times of frustration (that inevitably happen in five years of day and night commitment to research) as well as joy during the times of success, is something that I will never forget and that I will be thankful for the rest of my life.

An important role in my graduate career was played by Professor Jan Rabaey. Collaborating with Jan and his students was a fundamental piece of my professional development. I cannot even number the times that out of conversations with him, a new interesting idea came out. But the most important thing is that those conversations were not only at the BWRC, but also in the ski trips that he organizes for his group !

I am also very thankful to the outside member of my committee, Professor Paul Wright. It was a honor to have him in my committee because he is a great person that I deeply admire and who taught me a lot, especially at a personal level. During

these years I really enjoyed our friendship and I also had the privilege to be hooded by him. I am sure this friendship will last in the future.

I also had the privilege of having one of the greatest motivators of our times in my Qualification exam committee, Professor Adam Wolisz. Every time I am in a meeting where he is speaking (phone conference works just as fine) I see people walking out of it with a renew passion and willingness to take new challenges. He certainly had this impact on me and his continuous support during these last four years was a blessing for me.

When I first started my grad school here, my first real mentor was a senior graduate student in my group, who eventually became Professor Luca Carloni at Columbia University, and with whom I continued to collaborate throughout the years even after his appointment at Columbia. Luca was not only an academic mentor, but also a life mentor. Our discussions were spanning from sport to music and politics, and our senior/junior relationship (that was actually reversed when we were playing basket together) is still very important to me as I am sure it is for him.

Besides the professors, a special thank goes to my "little brother" Dragan Petrovic. I met him during my first class here in Berkeley, and we became inseparable friends and colleagues since then. We have been roommates, coauthors, and football teammates, he taught me many things (among which how to play Go) and continuously motivated me and supported me during all these years.

I would also like to thank Alessandro Abate, Alberto Diminin, and Alessandro Pinto, three very important Italian friends of mine whose inputs and feedbacks (again in both research and life) were always very valuable and with whom I shared very important moments of my life.

A special thank also to the various students who visited U.C.Berkeley and collab-

orated in my projects: Carlo Fischione, Fernando Pianegiani, Luca Necchi, Davide Gasperini, and Michele Comin.

A very important ingredient to the development of my research was the relationship with industry. The continuous feedbacks and sanity checks of my industry contacts have been invaluable. In this context, I would like to thank Fulvio Rusina‘, Renzo Calcagno, and the people involved in the Telecom-Pirelli Lab projects: Marco Sgroi, Giorgio Audisio, Marco Sabatini, Fabio Bellifemmine, Filippo Tempia, Claudio Borean, and Roozbeh Jafari.

When looking back at five years of my life, I cannot mention all the great people that I met and that made this Berkeley experience so unique. In particular, there are three partially overlapping groups that I would like to recall: the members and friends of Italian International Student Association (IISA), the Italian Family of the I-House, and last but not least, the people belonging to the famous Gert's IHouse Mailing List.

Finally, I would like to thank with all my heart the people that most of all had to make immense sacrifices for me. My father Sandro and my mother Annalisa, for always supporting my adventure in California despite the knowledge of being far away from their only child, and of course Tanja, who made a lot of sacrifices for me and whose love and patience carried me through the tough times and sleepless nights.

Chapter 1

Introduction

Ad-hoc wireless sensor networks have the definite potential to change the operational models of traditional businesses in several application domains, such as the building industry [1], power delivery [2], environmental control [3], and industrial automation [4, 5, 6]. Sensor networks are already the essential backbone of the "ambient intelligence" paradigm, which envisions smart environments aiding humans to perform their daily tasks in a non-intrusive way.

This revolution has not escaped the attention of both academia and industry and has led to a flurry of activities such as the exploration of new applications and the development of new radio architectures, low-power wireless sensor nodes [7, 8, 9, 10], low-date rate wireless protocols, and ad-hoc multi-hop routing algorithms [11, 12, 13, 14, 15]. The creation of forms of interoperability between the myriad of hardware components and software protocols is essential for the full potential of these technologies to be achieved. In this context, a number of new wireless standards such as 802.15.4 [16] and Zigbee [17] are under development. Yet, these efforts created in a bottom-up fashion do not fully address the essential question of how to allow interoperability across the many sensor network operational models

that are bound to emerge. In fact, different operational scenarios lead to different requirements, and hence different implementations, in terms of data throughput and latency, quality-of-service, use of computation and communication resources, and network heterogeneity. These requirements ultimately result in different solutions in terms of network topology, protocols, computational platforms, and air interfaces.

Another important drawback of bottom-up driven solutions is the incapability of develping systems that offer end to end communication guarantees. Most of the current approaches offer best effort quality of service and are optimized for single hop performance. Because of this system level unreliability, it is not possible to develop control applications on top of such a network architecture. Consequently, the combination of lack of reliability and support for heterogeneity are significantly slowing the commercial exploitation of this technology, especially in those markets, like industrial automation and healthcare monitoring, where meeting these requirements ia a must.

To support reliable system design and true interoperability between different applications as well as between different implementation platforms, I advocate the use of a rigorous design methodology based on a set of appropriate abstraction layers. The proposed approach is an application of Platform Based Design (PBD) [18, 19, 20]. PBD is a "meet-in-the-middle" design methodology, where system constraints are refined top-down, while implementation characteristics including performances such as delay and power consumption are abstracted bottom-up (see Fig. 1.1). The two parts are essential for selecting a good implementation via a design exploration phase that meets the constraints while estimating the performance of the candidate implementations. PBD relies on a clear identification of layers of abstraction, on a modeling strategy that captures uniformly functionality and architecture of the design, and on tools that map two contiguous layers, verify that the selected architectures satisfy constraints, and identify drawbacks and strengths of the design.

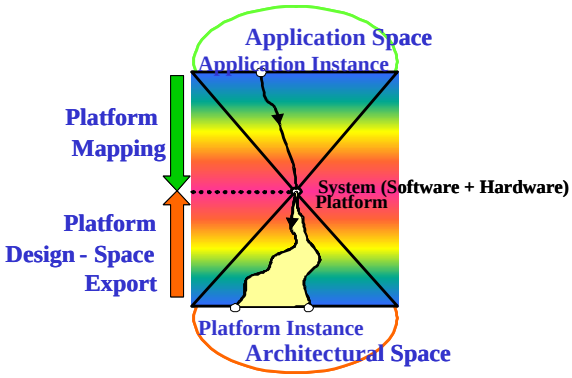

Figure 1.1. The Platform Based Design Approach

Although PBD was initially developed for classical embedded systems its principles can be applied to any communication problem including wireless sensor networks. PBD exploits the similarities between different communication problems to develop an effective methodology that can be applied in different domains [21]. In this dissertation, I specialize the general PBD methodology to the WSN case. This methodology is based on three abstraction layers, two mapping strategies, and the relative supporting tools.

The top and the bottom layers have been introduced before; the intermediate layer is novel. The first layer is an application interface called Sensor Network Service Platform (SNSP) [22]. The SNSP defines a set of services available to the end user to specify the target application formally without dealing with the details of a particular network implementation. While the SNSP description suffices to capture the interaction between controllers, sensors and actuators, it is a purely functional description, which does not prescribe how and where each of these functions will be implemented. Hence, information such as communication protocols, energy, delay, cost, and memory size, are not available. The lowest abstraction layer is the Sensor Network Implementation Platform (SNIP), which is a library of different hardware platforms that can be used to create the topology that supports the application. The abstraction layer in the middle is the Sensor Network Ad-hoc Protocol Platform

(SNAPP)and it is a library of communication protocols that can be optimized to be deployed on the given topology to satisfy end to end communication constraints while optimizing for energy consumption.

There are two mapping strategies that are targeted to two different application categories. The first one is called Static Mapping, and it addresses all those applications that can be modeled like a cyclic control routine and where a WSN can be deployed on demand to support that specific application. A typical example are temperature monitoring applications for buildings, or vibration monitoring of machines in a manufacturing plant. The term *Static* comes from the fact that the mapping of the application onto the network resources can be done offline at commissioning time, without the need of extensive run time reconfiguration of the system.

The other mapping strategy is called Dynamic Mapping and it addresses all those problems that do not fall into the static category. The dynamicity may be due to either the charachteristics of the application, such as continuous and unpredictable changes of traffic patterns (tipical of a network with strong human interaction), or the charachteristics of the network infrastructure, such as the scenarios in which the application has to be supported by an existing and predeployed network whose services are shared among different independent applications. In the dynamic case, an offline design approach is not enough, and a middleware was developed to allow the controllers to dynamically select the optimum run time mapping strategy.

As a side result, in this dissertation some novel communication protocols and aggregation algorithms are proposed. Specifically, RAND and SERAN are two new communication protocols that exploit network density to provide system level reliability and energy efficiency, while EERINA is robust data aggregation algorithm targeted to applications where cluster of sensors are deployed to monitor homogeneous

phenomena. I provide several case studies on building and industrial automation to validate both our methodology as well as the proposed protocols and algorithms.

In the next sections, I overview the different application domains that are of particular interest for wireless sensor networks, outlining their typical reqirements in terms of communication and standardization as well as cost of transitioning to wireless technology, then I discuss some of the previous attempts of defining system level design methodologies for wireless sensor networks. Finally, I give the outline of the rest of the dissertation.

1.1 Applications, Requirements, and Legacies

In this section, I discuss different application domains that are generally considered important targets for wireless sensor networks research, and describe the impact and benefit that WSN technology could potentially bring as well as the relative system level requirements and cost of transitioning to this new technology. In Table 1.1, the findings are summarized.

The first and presently most important driver for the adoption of WSN are building monitoring applications. Wireless, battery powered nodes can be easilily deployed on the walls of a building to monitor temperature, humidity, ventilation, and more recently gas or chemical leakage. Data is then conveyed to a central base station that decides on the opportune actuation (i.e. turn on air conditioning). The benefit of a wireless solution is that it is much easier to deploy wireless sensors (no need for rewiring cables through the walls) and in many situations this leads to much finer grane monitoring and consequent improved service levels and energy savings [1]. At the same time, the system level requirements for these applications are not too strict in terms of real time and reliability. Since most of these monitoring applications have

deadlines of several minutes, current sensor nodes can be turned off most of the time, and the relative power saving allows sufficiently high network lifetimes. Because of the clear value proposition and the current maturity of the technology, this category of applications is the one that has been penetrated the most by this technology. Several companies (Dust, CrossBow, Ember [23, 24, 25]) offer turn key solutions based on either proprietory protocol stack or open platforms. These solutions can be easily deployed on new buildings, but also on existing buildings to cooperate and improve existing systems. For these reasons the cost of adopting WSN technology is reasonably low. The next challenge in this domain is to raise the level of abstraction and have the WSN in a building to interoperate with the other wireless infrastructures such as WiFi and Bluetooth to create more added value services to the users. Several projects on ambient intelligence, both at the academic and industrial level [26], are targeted exactly to fill this gap. It is not clear at this point what the emerging solution will be, and what level of standardization will be required. But the capability of supporting different protocols with different QoS at the same time is a must in this domain, and a methodology to support such a development is required. I believe that the design methodology descibed in this dissertation, specially for the dynamic mapping problem in Chapter 5 is a step into this direction.

The second important application domain is sensor networks for manufacturing plants. Manufacturing plants are charachterized by a huge amount of sensors deployed in different locations of the manufacturing cells and most of them part of real time control routines for the automation line. The introduction of WSN has the potential of dramatically reducing installation and maintenance cost of these networks [6], as well as allowing for easy reconfiguration of existing manufacturing lines with great improvement in manufacturing productivity. Although wireless received a lot of attention from companies that operate in this area [27], still these sensor networks are mostly wired. There are two main application subdomains in this domain.

The first is what I called *manufacturing monitoring* applications. These are applications where wireless sensors are deployed on different machines of the manufacturing line to monitor the state of the machine and report it to a central station for preemptive maintenance. A typical example is wireless sensors deployed on robots to observe their vibration patterns and report suspect malfunctions to a central controller. Although from a requirement perspective, these applications are very similar to the building monitoring problems and some early deployments are visible [28], still more has to be done to ensure at the same time reliability and capability of supporting different networks.

The second subdomain is the one involving sensors that are part of the main control routines (i.e. proximity sensors). These sensors represent the majority of the sensors in a manufacturing line, and these are the ones to address to maximize the impact in this industry. However, the requirements in terms of real time and reliability are very strict, and current technology struggles to meet them. This is because nodes should be active and transmiting all the time, and this would limit their lifetime in case of battery operated platforms. For this reason, a key enabler for WSN in this domain is the capability of effectively scavenging energy from the environment, and some interesting efforts have recently been presented [27]. Even if the power issue were solved, still the relibility and heterogeneity requirements have to be addressed by an effective methodology, and in this area this is even more evident because the communication infrastructure has to satisfy strict regulatory standards in order to be usable.

Another interesting application domain is the capability of replacing the sensor networks inside the cars (or any other vehicle) using WSN. Except for some non safety critical applications such as tyre pressure monitoring, this domain is even more complex than the control systems in manufacturing plants. That is due not only to the extreme level of reliability and real time reactivity of these systems, but also to

Application	R.T.	Rel.	Het.	Lay.	Stand.	Trans. Cost
Building	Low	Mid	High	Mid	Low	Low
Manufacturing (monitoring)	Low	Mid	High	Mid	Low	Low
Manufacturing (control)	High	High	High	High	High	High
Automotive	High	High	Low	High	High	High
Logistics	Low	Mid	High	High	Mid	Mid
Healthcare	Mid	High	High	High	High	High

Table 1.1. Summary of requirements for different application domains.

the difficulty in operating in environments that are charachterized by a lot of metal such as car bodies or engines.

An interesting and emerging application field for WSN is logistic applications. Although warehouse monitoring with RFIDs was the initial driver in this domain, more and more WSN are being adopted in supply chain management issues, such as the control of the quality of transportation for delicate products (i.e. pharmaceutical) or to support human operators in logistic nodes, such as ports, airports, or cross docking stations. Because of the relatively mild requirements, this field has the opportunity to become an important driver for the adoption of WSN technology. Since these systems have to work together with expensive software infrastructure for company management, the capability of offering clear interfaces while supporting different communication protocols and hardware platforms will be a key to offer successfull solutions.

Finally, one the greatest opportunity is represented by healthcare monitoring applications. This is a booming new field and several conferences on WSN for healtcare monitoring have recently been organized. Although the opportunity of improving service levels and decreasing costs of healthcare is huge, cooping with difficult regulations may be difficult. However, there are important non safety critical applications in this domain, specially for home assistance such as injury recovery and rehabilitation, that are certaintly important avenues to gain momentum into this market.

1.2 Methodologies and Architectures

Because of the increasing design complexity and reliability requirements for WSN systems, there is a great focus on the development of effective design methodologies, both at the academic and industry level. In this section, I overview some of these efforts.

Historically, the most common design methodology for WSNs starts with the description of the protocol specifications using the NesC/TinyOS stack [29]. The NesC/TinyOS platform, developed at U.C. Berkeley, leverages a "method call" model of computation. It was designed to describe component-based architectures using a simple event-based concurrency model. This platform has then been enriched with a simulation environment called TOSSIM [30]. Its success is also related to the wide spreading of the hardware platforms of the Mica family [8]. Remarkably, the combination of Mica and TinyOS allowed for the development of many WSN applications.

Alternatively, protocol solutions are simulated using environment such as OMNET++ [31] or VisualSense [32] and then implemented in NesC/TinyOS. Omnet++ is a discrete event simulator developed by Andras Varga at the Technical University of Budapest. Although not specifically targeting the WSN domain, Omnet++ is widely used within the communication community for protocol simulations.

Visualsense is a modeling framework for WSN developed as part of the Ptolemy project at U.C.Berkeley [33]. It is an extension of a discrete-event model with an extra capability of describing properties of the wireless connectivity. Visualsense is a powerful tool to model and evaluate protocol solutions under different scenarios. Although an effort to move to a higher layer of abstraction is visible, especially with Visualsense, the current design flows are too oriented toward a bottom-up approach.

An attempt of raising the level of abstraction is presented in [34], where a classi-

fication for node communication mechanisms is introduced to allow for a higher level description of the network algorithms. In [35], a design methodology is presented. That methodology is based on a bottom-up part for the description of network algorithms, a top-down part to describe the application, and a mapping process to deploy the software code onto the nodes. The overall method fits with the Platform Based Design paradigms advocated in this dissertation, but leverages different layers of abstraction. My approach emphasizes the control based nature of WSN applications and offers a clear semantics and set of primitives to interpret timing issues at a very high level, hence providing a clear level of abstraction for the application designer.

Two approaches that have been very succesfull at an academic level are Directed Diffusion [36] and TinyDB [37]. Both of them are examples of query-based methods for high-level programming of WSNs where users specify the application as a set of tasks or queries (i.e. monitoring tasks) that the WSN must continuously perform. The request for data floods the network following a tree-based routing strategy until it reaches the desired node. Similarly, the query response is routed back to the originator of the query. On top of this, both approaches offer data-aggregation capabilities. Although I share the same vision of a WSN as a large distributed computing system where users describe the applications with simple and intuitive queries, I believe that a separation should be provided between the design of the aggregate computation and the design of the communication infrastructure, thereby reducing the design complexity and allowing for different communication protocols that could offer tailored energy-performance trade offs useful to address specific application domains.

A system level approach to the design of WSNs was recently presented in [38]. A platform called SP is proposed between the link and the network layer. The SP should provide the adequate modularity for the nodes to support different MAC and Routing layers. The philosophy is similar to the Internet "everything over IP", where in this case it would be "everything over SP". Although this is a very interesting architecture

for best effort networks, I believe it is not appropriate for control applications where end to end guarantees are required.

Several standardization proposals have been made for sensor networks in general and WSN in particular. The IEEE 1451.2 [39] standardizes both the key sensors (and actuators) parameters and their interface with the units that read their measures (or set their values). In particular, the standard defines the physical interface between the Smart Transducer Interface Module (STIM), which includes one or more transducers, and the Transducer Electronic Data Sheet (TEDS) containing the list of their relevant parameters, and the Network Capable Application Processor (NCAP), which controls the access to the STIM. IEEE 1451 was defined to improve the reusability of the network and component solutions for sensor networks within manufacturing plants. Although the initial targets were wired networks, the applicability of its concepts appeals to a wireless solution. IEEE 1451 presents already the concept of logical components (e.g, a sensor identifies a group of sensing devices rather than a single hardware component). Nevertheless, the IEEE 1451 standards are specifically targeted to the design of interfaces and they can hardly be generalized to capture application characteristics.

In [22], a service-based architecture is introduced. In that work two platform are presented. At the application level there is the Sensor Network Service Platform (SNSP), which is a collection of services that can be composed to specify different applications. At the lower level there is the Sensor Network Implementation Platform (SNIP) that describes the network infrastructure to support the application. Although that work describes a clear set of interfces, there is no hint on how the different services should be mapped onto the network infrastructure. This dissertation is exactly targeted to fill this gap for different application categories.

A number of standards for open communication in sensor networks have been pro-

posed. The most well-known are the BACnet and LonWorks standards, developed for building automation [1]. These standards are geared toward well-defined application areas, and are built on top of fairly well defined network structures. Hence, many of the exciting new developments that are emerging from the wireless sensor network community cannot be accommodated within these frameworks. At the same time, the application-specific functionality of both BACnet and LonWorks can easily be overlaid on top of the service-based model proposed in [22] .

There is a number of standards in the making at the ad-hoc wireless network layer. An important effort is represented by Zigbee advocated by a consortium of companies [17]. ZigBee defines an open standard for low-power wireless networking of monitoring and control devices. It works in cooperation with the IEEE 802.15.4 standard, which focuses on the lower protocol layers (physical and MAC). Instead, ZigBee defines the upper layers of the protocol stack, from network to application, including application profiles. From our perspective, Zigbee represents only one possible way to realize a network. The services proposed in [22] can be easily deployed in and on top of a Zigbee realization or alternative implementations such as Bluetooth Scatternets.

There are several commercial solutions that different vendors are now presenting on the market. Some of them are vertical solutions that use proprietory hardware, communication protocol and application interface in a turn key solution. This strategy charachterizes the systems offered by Dust Inc. and Bosch Inc. and so far have used mostly for building monitoring applications. Although these systems are in general charachterized by good performance and reliability, still they are "closed" solutions that do not allow the user to select different protocols. This may be problematic in domains such as industrial atomation where heterogeneity and standardization are strong requirements. On the other side of the spectrum are "open" solutions, like the one offered by CorssBow Inc.,where off-the-shelf hardware nodes can be matched

with a communication protocol and a user interface that can be purchased as a single vertical solution or as a set of layers interfaced by an evolution of TinyOS.

In summary, there are many different application domains that are potential important targets for WSN technology. Although a great advancement in the hardware platforms and energy scavenging techniques allowed for some initial success, still to be able to succefully address the challenges of these applications, an improvement on the system level design side is required. Specifically, issues such as reliability and support for heterogeneity must be addressed at a system level and a new methodology is needed to reach these goals. In this dissertation, I propose a design methodology based on the Platform Based Design and that continues the effort initiated in [22] to offer different synthesys strategies to design reliable WSN systems.

1.3 Outline

The rest of this dissertation is organized as follows: in Chapter 2, the generalities of the Platform Based Design are introduced, and the methodology is formalized using the mathematical framework of agent algebra.

In Chapter 3, the three platforms that carachterize the methodology are presented together with examples of platform instances. In this chapter, two new communication protocols for wireless sensor networks, called RAND and SERAN, are introduced and discussed.

In Chapter 4 the static mapping strategy for cyclic control applications is presented together with a framework, called Rialto, that is used to capture system specifications and produce a set of constraints for the design of the network infrastructure. Two case studies on building monitoring and industrial automation are presented in this chapter.

In Chapter 5, the tdynamic mapping strategy and the middleware that implements it are presented and validated with a case study on building monitoring.

In Chapter 6, two aggregation algorithms are presented that can be used to improve the designs produced by the proposed methodology in case of clustered topologies

In Chapter 7 there are some concluding remarks, an analysis of the impact of this work, and an indication of future avenues of research.

Chapter 2

Background: Platform Based Design

In this chapter, I present an overview the Platform Based Design methodology and focus on its application on communication synthesis problems, providing a formalization using agent algebras of the different steps of a design flow.

The increasing complexity of embedded electronic systems, together with shorter time to market and tighter correctness requirements, call for the development of efficient system level design methodologies that are able to:

1. Raise the level of abstraction, so that designers can easily describe the application independently from the final implementation.

2. Ensure correct by construction design, so that the final implementation is guaranteed to satisfy the initial requirements without the need of extensive verification.

3. Maximize component reuse, so that the hardware and software blocks developed in previous designs can be used in new designs to speed up the design process.

The Platform Based Design (PBD) [18] is a methodology that was developed to address these very issues.

According to the PBD, a design is composed by a sequence of steps that lead the initial high level description, all the way down to the implementation. Each step is a refinement process that takes the design from a higher level description to a lower level description that is progressively closer to the final implementation. This refinement step is obtained by replacing each block of the higher level description, with blocks (or composition of blocks) from a lower level description. Among the possible lower level implementations, the methodology selects one that satisfies the constraints coming from the higher level description, while optimizing for some cost function (i.e. chip area, or power consumption). For each layer of abstraction, these design blocks, together with a description of their interfaces and performance, are stored in a library, called *platform*. The higher the initial level of abstraction, the easier is expressing the functionality and constraints as well as catching design errors early, but the more difficult is to reach quickly a high-quality implementation due to the semantic gap between specification and implementation. Differently from classical top-down or bottom-up approaches, in PBD each step is a combination of both approaches where application constraints are refined in a top-down fashion. architecture performance are abstracted in a bottom up fashion, and a *meet in the middle* phase decides the final implementation solving a constrained optimization problem.

The PBD methodology was further specialized for communication synthesis problems [21]. As in any design problem, to perform communication design I need first to specify the functionality and constraints that I have to satisfy. Assume I want to interconnect a set of nodes (e.g., computers) so that every node in the set can access every other node. Our initial specification includes the quality of service that each connection must be able to support, such as the required bandwidth and the maximum latency of the communication. I can solve this problem by constructing a network

made of several different components such as routers, hubs, modems, protocol stacks, and links of different nature. The resources must be sized to satisfy the required constraints. However, the gap between our original, high–level, specification, and the implementation is clearly too large to be bridged in a single synthesis step: clearly, enumerating all possible topologies and interconnections is not practical, even for networks of modest complexity. A better way of approaching this problem is to divide this gap in several layers, where each layer focuses on a particular design choice. The question is then whether this division is optimal and, more importantly, how much of the entire design space can be explored. Answering these questions gives us an idea of the quality of the solutions that I obtain. The PBD approach consists of quantifying the design exploration process by relating the levels of abstractions corresponding to different layers. If two layers are too far apart then performance estimation will likely be poor and will not provide the necessary support for the synthesis algorithms.

In this context, a *platform* consists of a set of library elements, or resources, that can be assembled and interconnected according to predetermined rules to form a platform *instance*. One step in a platform-based design flow involves mapping a function or a specification onto different platform instances, and evaluating its performance. By employing existing components and interconnection resources, reuse in a platform-based design flow shifts the functional verification problem from the verification of the *individual elements* to the verification of *their interaction* [40, 41]. In addition, by exporting an abstracted view of the parameters of the model, the user of a platform is able to estimate the relevant performance metrics and verify that they satisfy the design constraints. The mapping and estimation step is then repeated at increasingly lower levels of abstraction in order to come to a complete implementation.

Crossing the boundaries between abstraction levels, i.e., the process of abstracting or refining a specification, is often non-trivial. The most common pitfalls include

mishandling of corner cases and inadvertently misinterpreting changes in the communication semantics.

These problem arise because of the poor understanding and the lack of a precise definition of the abstraction and refinement maps used in the flow. In addition, abstraction and refinement should be designed to preserve, whenever possible, the properties of the design that have already been established. This is essential to increase the value of early, high-level models and to guarantee a speedier path to implementation.

2.1 Formalizing Platform Based Design

The formalization of the platform based design methodology is based on the framework of Agent Algebra [42]. Informally, an agent algebra $\mathcal{Q}$ is composed of a domain D that contains the agents under study for the algebra, and of certain operators that formalize the most common operations of the models of computation used in embedded system design. Different models of computation are constructed by providing different definitions for the domain of agents and the operators. The algebra also includes a master alphabet $\mathcal{A}$ that is used as the universe of "signals" that agents use to communicate with other agents.

Definition 2.1.1 *An* agent algebra $\mathcal{Q}$ *has a domain* $\mathcal{Q}.D$ *of* agents, *a* master alphabet $\mathcal{Q}.\mathcal{A}$, *and three operators:* renaming, projection *and* parallel composition, *denoted by* $\text{rename}(r)$, $\text{proj}(B)$ *and* $\|$. *Each agent* $p \in \mathcal{Q}.D$ *is associated with an* alphabet $A \subseteq \mathcal{A}$.

The operators of the algebra are partial functions on the domain D and have an intuitive correspondence with those of most models of concurrent systems. The operation of renaming, which takes as argument a renaming function r on the alphabet, corre-

sponds to the instantiation of an agent in a system. Projection corresponds to hiding a set of signals, and takes the set B of signals to be retained as a parameter. Hence it corresponds to an operation of scoping. Finally, parallel composition corresponds to the concurrent "execution" of two agents. It is possible to define other operators. I prefer to work with a limited set and add operators only when they cannot be derived from existing ones. In particular, in this work I will be mainly concerned with the operator of parallel composition. The operators must satisfy certain axioms that formalize their intuitive behavior and provide some general properties that I want to be true regardless of the model of computation. For example, parallel composition must be associative and commutative. The definition of the operators is otherwise unspecified, and depends on the particular agent model being considered.

The notion of refinement in each model of computation is represented by adding a preorder (or a partial order) on the agents, denoted by the symbol $\preceq$. The result is called an *ordered agent algebra*. I require that the operators in an ordered agent algebra be monotonic relative to the ordering. This is essential to apply compositional techniques. However, since these are partial functions, this requires generalizing monotonicity to partial functions. This generalization is however beyond the scope of this paper. The interested reader is referred to [42] for more details.

It is easy to construct an agent algebra $\mathcal{Q}$ to represent the interface that components expose to their environment. In this case, the set D consists of the agents of the form $p = (I, O)$ where $I \subseteq \mathcal{Q}.\mathcal{A}$ is the set of input ports of the components and $O \subseteq \mathcal{Q}.\mathcal{A}$ the set of output ports. The alphabet of an agent p is simply $A = I \cup O$, and I require that the set of inputs and outputs be disjoint, i.e., $I \cap O = \emptyset$. The parallel composition $p = p_1 \parallel p_2$ is defined only if the sets O_1 and O_2 are disjoint, to ensure that only one agent drives each port. When defined, a port is an output of the parallel composition if it is an output of either agent. Conversely, it is an input if it is an input of either p_1 or p_2, and it is not concurrently an output of the other

agent. Thus $O = O_1 \cup O_2$ and $I = (I_1 \cup I_2) - (O_1 \cup O_2)$. Given the definitions, it is clear that in this example connections are established by name.

The model can be enriched with information about the nature of the signals used by the agents. For instance, in the case of agents that describe communication topologies, signals can be distinguished between those that belong to a link, denoted by the symbol l, and those that belong to a component, denoted by the symbol n (non-link). I call this a *typed IO agent algebra*. The sets I and O of an agent p thus become sets of pairs of signals together with their type, i.e., $I \subseteq \{(a,t) : a \in \mathcal{Q}.\mathcal{A} \wedge t \in \{l,n\}\}$ and similarly for the output ports. Parallel composition can also be modified so that the operation is defined only if the ports of the agents being connected are *not* of the same type, i.e., a link must be used to connect two regular ports. Hence, $p_1 \parallel p_2$ is defined if and only if for all $i \in I_1$ and for all $o \in O_2$, if $i.a = o'.a$ then $i.t \neq o'.t$, and viceversa for p_2 and p_1.

Different agent algebras are related by means of conservative approximations. A conservative approximation from $\mathcal{Q}$ to $\mathcal{Q}'$ is a pair $\Psi = (\Psi_l, \Psi_u)$, where Ψ_l and Ψ_u are functions from $\mathcal{Q}.D$ to $\mathcal{Q}'.D$. The first mapping is an upper bound of the agent relative to the order of the algebra: for instance, the abstract agent represents all of the possible behaviors of the agent in the more detailed domain, plus possibly some more. The second is a lower bound: the abstract agent represents only possible behaviors of the more detailed one, but possibly not all. Formally, a conservative approximations is an abstraction that maintain a precise relationship between the orders in the two agent algebras.

Definition 2.1.2 *Let $\mathcal{Q}$ and $\mathcal{Q}'$ be ordered agent algebras, and let Ψ_l and Ψ_u be functions from $\mathcal{Q}.D$ to $\mathcal{Q}'.D$. I say that $\Psi = (\Psi_l, \Psi_u)$ is a* conservative approximation from $\mathcal{Q}$ to $\mathcal{Q}'$ *if and only if for all agents p and q in $\mathcal{Q}.D$,*

$$\Psi_u(p) \preceq \Psi_l(q) \Rightarrow p \preceq q.$$

Thus, when used in combination, the two mappings allow us to relate refinement verification results in the abstract domain to results in the more detailed domain. Hence, the verification can be done in $\mathcal{Q}'$, where it is presumably more efficient than in $\mathcal{Q}$. The conservative approximation guarantees that this will not lead to a false positive result, although false negatives are possible depending on how the approximation is chosen.

To define the inverse Ψ_{inv} of an approximation, I investigate whether there are agents in $\mathcal{Q}.D$ that are represented exactly by Ψ_u and Ψ_l rather than just being bounded. I do so by only considering those agents p for which $\Psi_l(p)$ and $\Psi_u(p)$ have the same value p'. Intuitively, p' represents p exactly in this case, and I therefore define $\Psi_{inv}(p') = p$. If $\Psi_l(p) \neq \Psi_u(p)$, then p is not represented exactly in $\mathcal{Q}'$. In this case, p is not in the image of Ψ_{inv}.

Definition 2.1.3 *Let $\Psi = (\Psi_l, \Psi_u)$ be a conservative approximation from $\mathcal{Q}$ to $\mathcal{Q}'$. For $p' \in \mathcal{Q}'.D$, the inverse $\Psi_{inv}(p')$ is defined and is equal to p if and only if $\Psi_l(p) = \Psi_u(p) = p'$.*

If the algebra $\mathcal{Q}$ is partially ordered (as opposed to preordered), the inverse of the conservative approximation is uniquely determined. Otherwise, a choice may be possible among order equivalent agents. In all cases, however, because of the defining properties of a conservative approximation, Ψ_{inv} is one-to-one, monotonic, and inverse of both Ψ_l and Ψ_u.

Assume now that for an agent p, $\Psi_{inv}(\Psi_l(p))$ and $\Psi_{inv}(\Psi_u(p))$ are both defined, It is easy to show that $\Psi_{inv}(\Psi_l(p)) \preceq p \preceq \Psi_{inv}(\Psi_u(p))$. This fact makes precise the intuition that $\Psi_l(p)$ and $\Psi_u(p)$ represent a lower and an upper bound of p, respectively.

I can us agent algebras to describe formally the process of successive refinement in a platform-based design methodology. There, refinement is interpreted as the con-

cretization of a *function* in terms of the elements of a *platform*. The process of design consists of evaluating the performance of different kinds of instances in the platform by mapping the functionality onto its different elements. The implementation is then chosen on the basis of a cost function. I use three distinct domains of agents to characterize the process of mapping and performance evaluation. The first two are used to represent the platform and the function, while the third, called the *common semantic domain*, is an intermediate domain that is used to map the function onto a platform instance.

A platform, depicted in Figure 2.1 on the right, corresponds to the implementation search space.

Definition 2.1.4 *A platform consists of a set of elements, called the* library elements*, and of* composition rules *that define their admissible topologies of interconnection.*

To obtain an appropriate domain of agents to model a platform, I start from the set of library elements D_0. The domain of agents D is then constructed as the closure of D_0 under the operation of parallel composition. In other words, I construct all the topologies that are admissible by the composition rules, and add them to the set of agents in the algebra. Each element of the architecture platform is called a *platform instance*.

Performance evaluation usually requires that the elements of a platform include information regarding their internal structure. Thus, an algebra such as the typed IO agent algebra described is not suitable for this purpose, since composition doesn't retain the structure of the agent. The IO agents can, however, be used as library elements D_0. A new domain of agents D can then be constructed as follows. If $p_0 \in D_0$ is a library element, I include the *symbol* $\mathbf{p_0}$ in the set of agents $\mathcal{Q}.D$. I then close the set D under the operation of parallel composition. However, I represent a

composition $p = p_1 \| p_2$ in $\mathcal{Q}$ as the *sequence of symbols* $\mathbf{p_1} \parallel \mathbf{p_2}$. By doing so, I retain the *structure* of the composite, since all the previous composition steps are recorded in the representation. I call this process a *platform closure*.

Definition 2.1.5 *Given a set of library elements D_0 and a composition operator $\|$, the* platform closure *is the algebra with domain*

$$D = \{\mathbf{p} : p \in D_0\} \cup \{\mathbf{p_1} \parallel \mathbf{p_2} : \mathbf{p_1} \in D \wedge \mathbf{p_2} \in D\} \tag{2.1}$$

where $p_1 \| p_2$ is defined if and only if it can be obtained as a legal composition of agents in D_0.

The construction outlined above is general, and can be applied to building several different platforms, as will be shown later. The result is similar to a term algebra with the "constants" in D_0 and the operation of composition. Unlike a term algebra, however, our composition is subject to the constraints of the composition rules. For example an "architecture" platform may provide only one instance of a particular processor. In that case, topologies that use two ore more instances are ruled out. In addition, the final algebra must be taken up to the equivalence induced by the required properties of the operators. For example, since parallel composition must be commutative, $\mathbf{p_1} \parallel \mathbf{p_2}$ should not be distinguished from $\mathbf{p_2} \parallel \mathbf{p_1}$. This can be accomplished by taking the appropriate quotient relative to the equivalence relation. The details are outside the scope of this paper.

On the other hand, the function, depicted in Figure 2.1 on the left, is represented in an agent algebra called the *specification domain*. Here the desired function may be represented denotationally, as the collective behavior of a composition of agents, or may retain its structure in terms of a particular topology of simpler functions. The denotational representation is typically used at the beginning of the platform-based design process, when no information on the structure of the implementation

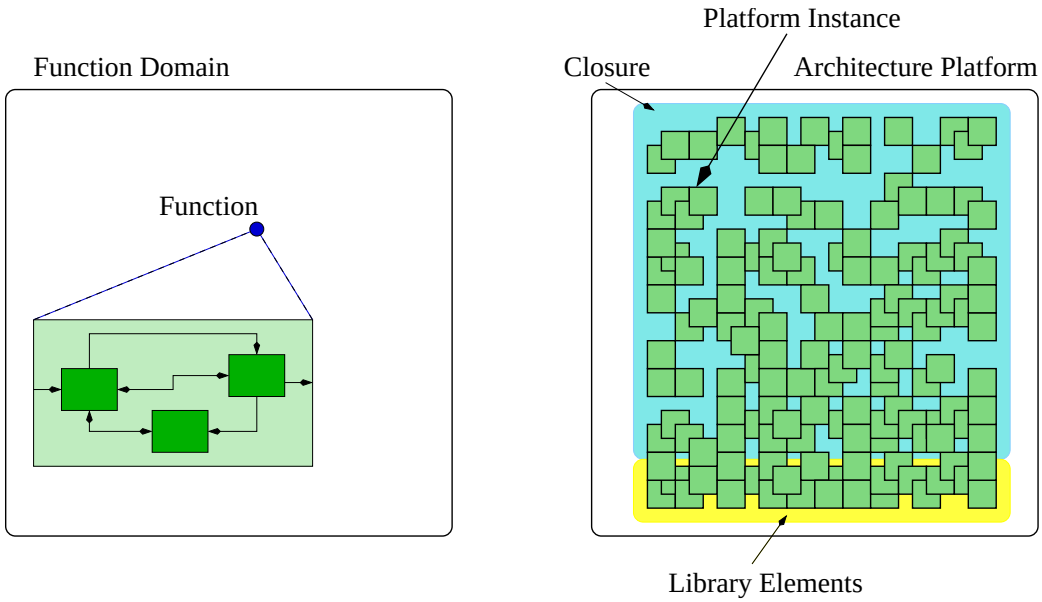

Figure 2.1. Architecture and Function Platforms

is available. Conversely, after the first mapping, the subsequent refinement steps are started from the mapped platform instance, which is taken as the specification. Thus, a *common semantic domain*, described below, is used as the specification domain. However, contrary to the mapping process that is used to select one particular instance among several, when viewed as a representation of a function the mapped instance is a specification, and it is therefore fixed.

The function and the platform come together in an intermediate representation, called the *common semantic domain*. This domain plays the role of a common refinement and is used to combine the properties of both the platform and the specification domain that are relevant for the mapping process. The domains are related through conservative approximations.

Definition 2.1.6 *Given a platform* $\mathcal{Q}^P$ *and specification domain* $\mathcal{Q}^S$*, a* common semantic domain *is an agent algebra* $\mathcal{Q}^C$ *related to* $\mathcal{Q}^P$ *and* $\mathcal{Q}^S$ *through conservative approximations* Ψ^P *and* Ψ^S*, respectively.*

In particular, I assume that the inverse of the conservative approximation is defined at the function that I wish to evaluate. The function therefore is mapped onto the common semantic domain as shown in Figure 2.2. This mapping also includes all the refinements of the function that are consistent with the performance constraints, which can be interpreted in the semantic domain.

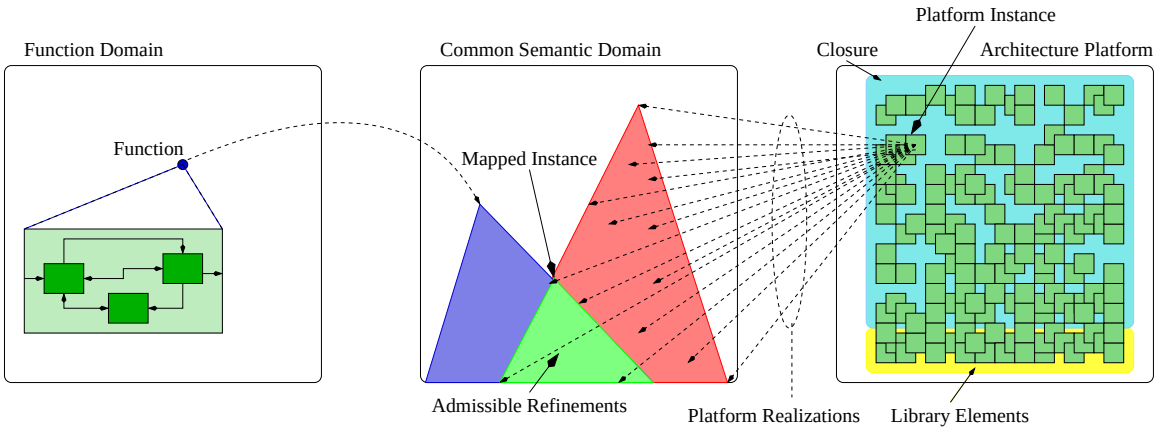

Figure 2.2. Mapping of function and architecture

If the platform includes programmable elements, the correspondence between the platform and the common semantic domain is typically more complex. In that case, each platform instance may be used to implement a variety of functions, or behaviors. Each of these functions is in turn represented as one agent in the common semantic domain. A platform instance is therefore projected onto the common semantic domain by considering the collection of the agents that can be implemented by the particular instance. This projection, represented by the rays that originate from the platform in Figure 2.2, may or may not have a greatest element. If it does, the greatest element represents the non-deterministic choice of any of the functions that are implementable by the instance.

The common semantic domain is partitioned into four different areas. I am interested in the intersection of the refinements of the function and of the functions that are implementable by the platform instance. This area is marked “Admissible

Refinements" in Figure 2.2. Each of the admissible refinements encodes a particular mapping of the components of the function onto the services offered by the selected platform instance. These can often be seen as the *covering* of the function through the elements of the platform library. Of all those agents, those that are closer to the greatest element are more likely offer the most flexibility in the implementation. Once a suitable implementation has been chosen (by possibly considering different platform instances), the same refinement process is iterated to descend to an even more concrete level of abstraction. The new function is thus the intersection of the behavior of the original function and the structure imposed by the platform. The process continues recursively at increasingly detailed levels of abstraction to come to the final implementation.

Chapter 3

Platforms and Instances

In this chapter, I introduce the different platforms that were developed to support the methodology and I provide some examples of instances of the different platforms.

Specifically, I indentify three layers of abstractions and define a platform for each of them. At the highest level there is the Sensor Network Service Platform (SNSP) that is used by the end user to describe the application. At the lowest level, there is the Sensor Network Implementation Platform (SNIP) that is used to describe the different hardware nodes, and in the middle I introduced the Sensor Network Adhoc Protocol Platform (SNAPP) that is used to describe the different communication protocol solutions. While the first two platforms were initially introduced in [22], their formalization in an algebraic framework as well as the SNAPP is a novel contribution.

In the rest of the chapter, I introduce the SNSP and SNIP, and then focus on the SNAPP giving examples of different protocols and how they are charachterized. Specifically, I describe and charachterize two protocols, RAND and SERAN, that I developed respectively for uniform and clustered topologies, then I consider a standardized protocol such as the IEEE 802.15.4 and describe how it can be charachterized

to be included in the SNAPP. For each platform, first I give an intuitive description, then I define it using the algebraic framework introduced in the previous chapter.

3.1 The Sensor Network Service Platform

The definition of sockets in the Internet has made the use of communication services independent from the underlying protocol stack, the communication media and the various possible operating systems. The goal of the SNSP is to introduce a similar abstraction layer.

A properly defined application interface captures all the possible services that can be used by any sensor network application and supported by any sensor network platform. A WSN is composed of controllers, sensors and actuator. To perform its functionality, a controller (algorithm) has to be able to read and modify the state of the environment. In a WSN, controllers do so by relying on communication and coordination among a set of distinct elements that are distributed in the environment to complete three different types of functions: sensing, control and actuation. The role of the Sensor Network Services Platform (SNSP) is to provide a logical abstraction for these communication and coordination functions. This approach allows the user to specify the application while abstracting away the specific details of the communication mechanisms (routing strategies, MAC protocols, physical channel characteristics) thereby making possible for the application designer to focus on the task of developing the control algorithms for the WSN application.

In particular,the SNSP is a collection of data processing functions (e.g. aggregation) and I/O functions (sensing, actuation) that cooperate in order provide the following services:

- *query service (QS)* used by controllers to get information from other components;
- *command service (CS)* used by controllers to set the state of other components;
- *timing/synchronization service (TSS)* used by components to agree on a common time;
- *location service (LS)* used by components to learn their location;
- *concept repository service (CRS)* which maintains a map of the capabilities of the deployed system and it is used by all the components to maintain a common consistent definition of the concepts that they agreed upon during the network operation.

The CRS is quite novel in the WSN community, but is deemed essential if a true ad-hoc realization of the network is to be obtained. The repository includes definitions of relevant global concepts such as the attributes that can be queried (e.g. temperature, pressure), or the regions that define the scope of the names used for addressing. It further allows collecting information about the capabilities of the system (i.e. which services it provides and at which quality and cost) and provides the application with a sufficiently accurate description. The repository is dynamically updated during the network operations. Access to the SNSP services is provided to the application through a set of primitives, combined in the *application interface (AI)*.

Following the algebraic approach introduced in the previous chapter, the SNSP can be charachterized by defining a set of agents and how they can be composed to create an instance. An instance of this platform is also called Application.

There are three types of agents:

1. Services: these are the previously described macroinstructions used to achieve a

specific goal within an algorithm (i.e. Query Service to ask for data, Command Service to force an actuation).

2. Conditional Blocks: used to relate time triggered or data triggered events to specific decision on actuations or further service requests.

3. Directional Links: links abstract the sequentiality of two services or conditional blocks.

Services or conditional statements can be composed only if a directional link is decleared between the two. Consequently, in its general conception, an application can be described using a simple flowchart. I believe that this approach is fundamental to allow this methodology to be succesfull in different application classes. However, further restrictions on the compositions can be imposed depending on the application domain as it will be shown in the next two chapters. In those chapters, I will also present effective methodologies to capture specifications in those specific domains.

3.2 The Sensor Network Implementation Platform

The Sensor Network Implementation Platform (SNIP) is a library of physical nodes that can be used to support the application. A physical node is a collection of physical resources such as:

- clocks and energy sources;
- processing units, memory, communication, and I/O devices;
- sensor and actuator devices.

In particular, the main physical parameters of a node are:

- list of sensors and actuators attached to node;
- memory available for the application;
- clock frequency range;
- clock accuracy and stability;
- level of available energy;
- cost of computation (energy);
- cost of communication (energy);
- transmission rate (range).

Example of physical nodes are the commercial hardware platforms such as Mica [8] and Telos motes [10], as well as Base Stations such as the Stargate [24].

Using the algebraic approach, the SNIP can be defined as a library whose agents are the hardware nodes, the base stations and bidirectional links. The hardware nodes and base stations are characterized not only by their physical resources, but also by their location. An instances of this platform is called a Topology.

In a topology, physical components can be connected only using links. A link represents the capability of communication between two physical components. Restrictions on the possibility of linking directly two components reflect the reachability due to their radio interface and distance. For example MicaZ and Telos motes can be linked since they both are ZigBee compatible, while Mica2 and MicaZ, that are not radio compatible, cannot be directly linked, but a path between the two can exist only if there is also a third component (i.e. a base station or a node with a reconfigurable radio) that is able to support both radio interfaces.

A partial order can be defined for this platform such as $v_1 \preceq v_2$ if and only if for each component in v_2 there is a correspondent component in v_1 and for each link in v_2 there is a corresponding link in v_1. In other words higher elements are the ones representing minimum topologies.

3.3 The Sensor Network Ad-hoc Protocol Platform

To choose the architecture of the SNIP and to map the functional specification of the system onto it are critical steps in sensor network design. To facilitate the process I created an intermediate level of abstraction called Sensor Network Ad-hoc Protocol Platform (SNAPP).

The SNAPP is a library of MAC and routing protocols. Because of the strict energy requirements of wireless sensor networks, many protocol solutions address MAC and routing in a monolithic fashion. Consequently, the agents of this platform are MAC protocols, routing protocols, or integrated MAC+Routing protocols. Composition of these elements is obviously limited to single MAC solutions and single routing solutions, whenever they are compatible (i.e. IEEE 802.15.4 MAC and ZigBee network layer).

These protocols are "parametrized protocols", meaning that their structure is specified, but their working point is determined by a set of parameters. In general, these parameters are the free parameters of the protocol that can be easily tuned by the application developer. For example, the access probability in a p-persistent CSMA scheme, or the wake up rate of the nodes for the unbeaconed version of the IEEE 802.15.4. As it will be shown in the next chapter, the value of these parameters is obtained as the solution of a constrained optimization problem, where the constraints

are derived from the latency, error rate, sensing requirements of the application while the cost function is the energy consumption. The energy consumption is estimated based on an abstraction of the physical properties of the candidate hardware platform.

Although I specifically developed some of these protocols, such as RAND and SERAN that are described later in this chapter, any protocol can be included in the SNAPP as long as the end to end delay distribution and energy consumption performance of the protocol are charachterized. An important and usually non trivial step is the charachterization of the end-to-end (E2E) delay distribution. In general, this modeling effort is divided in two steps. The first step is aimed at the analysis of the single hop perfomance of the protocol. Specifically, I consider parameters such as the number of nodes in the neighborhood, the wake up rate of the receiving nodes and the distribution of the number of transmission attempts before observing a succefull packet exchange.

Once I have the delay distribution for the single hop, this result must be extended to the multihop chain. Depending on the networking protocol, different techniques can be used for this step:

- Deterministic Routing. These are protocols where the overall routing structure is mostly constant and the transmissions are scheduled. In this case, the modeling is simple since I just need to evaluate the schedule. This is the case of protocols like SMAC [11] or SERAN [43].

- Geographic Routing. This is the class of protocols for which the next hop is selected randomly among the nodes that belong to a target region that gives an appropriate progress toward the final destination [14, 44]. In this case, given the initial topology it is possible to estimate a priori the expected number of hops in the multihop chain. Consequently, using the mean and and variance

of the single hop delay distribution, I model the E2E delay distribution as a Normal variable using the Central Limit Theorem.

- Dynamic Routing. This is the case of the routing protocols where the next hop is decided at run time based on the current estimated connectivity with the neighbors [45]. This is an important class of protocols and the most difficult to develop models for because it is not possible to estimate at design time the number of hops in the multihop chain. However, in most cases, an upper bound on the number of expected hops can be inferred from the initial topology, and that number can be used to extend the single hop delay distribution to the E2E distribution using the Central Limit Theorem.

The mathematical framework allows us to capture the requirements of the design functionality and performance as a constrained optimization problem. The solution to this problem provides the parameters to derive the final protocol implementation. Once the trade-off equations are derived and solved as an optimization problem, all the protocol parameters are automatically synthesized.

The use of parameterized protocols allows us to effectively restrict the large design space to a few parameters. In addition, since the protocols are developed with a specific mathematical model in mind, I can easily gouge the effects of changing these parameters on the overall network performance. This predictive ability prevents the need for extensive simulation and allows for the ease of comparison with other protocols.

In the next sections I introduce and charachterize two protocols that I developed for different WSN topologies, called RAND and SERAN, and then I show how to charachterize a standardized protocol such as IEEE 802.15.4. While RAND is provided of a distributed run time optimization algorithm that allows the nodes to adapt to the optimal working point for the network, SERAN and 802.15.4 require off-line

computation to optimize the relative parameters. In the next chapter, I present two case studies where this off-line optimization process is performed for these two protocols.

3.4 Example: RAND

The Randomized Protocol (RAND) is a MAC and Routing solution for scenarios charachterized by dense and uniforms topologies. An example of such a topology could be an indoor application (i.e. a room or a corridor) where many cheap wireless nodes are embedded in the floor, or in the walls to carry on localized sensing tasks (such as intrusion detection) and report the data to a sink in a low power multihop fashion with latency requirements to satisfy.

Although wireless nodes miniaturization is enabling these types of applications, the difficulty in developing reliable protocols is delaying the commercial blossom of this technology. Random behaviors like nodes malfunctioning and failure, challenge communication performance and robustness are typical of these scenarios. Nevertheless, decaying costs will allow to deploy high densities and I believe that leveraging this resource is the key to ensure reliable communication out of unreliable components.

I believe that protocol design should adapt to the inherent randomness of these systems. Furthermore, high node densities allow to characterize functionalities for group of nodes instead of single nodes [46] [15], hence ensuring a much higher robustness.

Consequently, I developed a protocol solution with a randomized routing, a randomized MAC and a randomized sleeping discipline that leverage node density and are jointly optimized for energy consumption. I also introduce a completely distributed adaptation algorithm that allows the network to adapt to traffic variations

and synthesize the protocol configuration parameters to reach the optimal working point without communication or state overhead.

Many solutions for sleeping disciplines have been recently proposed. Most of them try to put the nodes to sleep while preserving a connectivity graph and rely upon strong synchronization in the network [15] [13]. I believe that a randomized approach where only group properties are preserved should be followed, as proposed in [46]. According to this discipline, each node goes to sleep for an amount of time that is a random variable whose parameters are a function of traffic and network conditions. Our duty cycle solution can be considered an extension of [46] where the node wake up rate adaptation algorithm is further refined and distributed.

3.4.1 The Problem

There is a Source that sends packets to a Destination at a rate λ. Between Source and Destination a high density of nodes is uniformly deployed to relay these packets. These nodes are placed in a planar surface (i.e. a wall or ceiling). The communication infrastructure must offer the following services:

1. End-to-End (E2E) delay guarantee. The two sigma distribution of the E2E packet delay must stay within τ seconds: $P[E2E \leq \tau] \geq 0.96$.

2. Error Rate guarantee Each packet must arrive to D with probability at least Ω: $P[correct] \geq \Omega$.

I assume that each node knows its location This information can be either hard-coded in the node when deployed, or it can be obtained running a locationing algorithm on the network right after deployment. I further assume that nodes have tunable transmitting power, and packets piggyback informations on the traffic rate λ and the position of S and D.

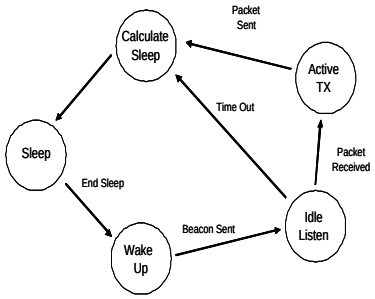

Figure 3.1. Protocol

3.4.2 The Randomized Protocol

Since the proposed randomized protocol is a cross-layer solution, I present the MAC, routing and duty-cycle algorithms all together.

The behavior of a node can be explained considering the state machine of Figure 3.1.

- **SLEEP STATE**: the node turn off its radio and starts a grenade timer whose duration is an exponentially distributed random variable of intensity μ. When the timer expires, the node goes to the WAKE UP state.
- **WAKE UP STATE**: the node turn its radio on and broadcasts a message indicating its location and that it is ready to receive (Beacon message). The node goes to the IDLE LISTEN state.
- **IDLE LISTEN STATE**: the node starts a grenade timer of a fixed duration that must be long enough to completely receive a packet. If a packet is received, the timer is discarded and the node goes to the ACTIVE TX state. Otherwise if the timer expires before any packet is received, the node goes to the CALCULATE state.
- **ACTIVE TX STATE**: the node calculates the size of the forwarding region (FwR). The FwR is the region between the maximum and minimum distance

(d_{max}, d_{min}) at which the next hop must be. The node waits for the first Beacon coming from a node within the FwR and forwards the packet to it. After the transmission is completed it goes to the CALCULATE state.

- **CALCULATE STATE**: the node calculates the intensity parameter μ for the next sleeping time and generates an exponentially distributed random variable of mean $1/\mu$. After this the node goes back to the SLEEP state.

Consequently:

1. The selection of the next hop is a random choice among nodes of a calculated region.
2. The duty-cycling algorithm is randomized.
3. The MAC is random based and does not implement any acknowledgment and retransmission scheme.
4. The working point of each node is determined by the size of the FwR and the wake up intensity μ.

I now show how to adaptively tune these parameters to satisfy delay and error rate constraints and optimize for power consumption.

3.4.3 Mathematical Model

To set the wake up and FwR parameters to an optimal working point, I model the network performance as a constrained optimization problem, where constraints are the E2E delay and error rate requirements and the cost function is the energy consumption of the network.

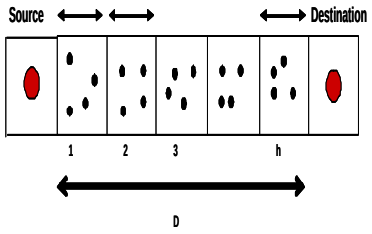

Figure 3.2. Block abstraction

Similarly to [46] and [47], I introduce the block abstraction to build a mathematical model of the protocol (Figure 3.2).

I consider the nodes layout as divided in h blocks. These blocks represent the forwarding regions. Consequently, a node in block i can forward his packets only to a node in block i+1. The number and size of these blocks, together with the wake up rate of the nodes within the same block, are the parameters that I optimize.

Note that the block abstraction is only used to create a model and it is not implemented.

The E2E delay is given by the sum of the delays at each hop. At each hop there are two sources of delay:

1. Time to wait before the first wake up of a node in the FwR. Since the inter wake up time of each node is an exponentially distributed random variable, the time to wait before the first wake up in a block is an exponentially distributed random variable whose intensity is the sum of the intensities of the single nodes. I call $\mu_{c,i}$ the cumulative wake up rate of block i.

2. Time to forward a packet once the connection is established. I call this time F.

Assuming h hops, the E2E delay constraint becomes:

$$P[hF + \sum_{i=1}^{h} \alpha_i \leq \tau] \geq 0.96 \tag{3.1}$$

where $\alpha_i \in Exp(\mu_{c,i})$

Since I do not implement contention or acknowledgment and retransmission schemes, a packet can be lost at each hop because of a collision or because of a bad channel during transmission.

1. Collisions: consider the case of a node in block i that has to send a packet to a node in block i+1. A collision occurs if another node in block i receives a packet before a node in block i+1 has broadcasted a beacon. Modeling the incoming traffic of block i as a Poisson process of intensity λ, this event happens with probability

   $P[coll] = \frac{\lambda}{\lambda + \mu_{c,i}}$.

2. Bad channel: to obtain a tractable model, I consider the probability of having a good channel during a single transmission as a Bernoulli variable of parameter p. In the simulations, I test the robustness of our protocol with more realistic channel models.

Consequently, assuming h hops, the error rate constraint becomes:

$$\prod_{i=1}^{h} \frac{p\mu_{c,i}}{\lambda + \mu_{c,i}} \geq \Omega \tag{3.2}$$

The energy consumption is given by the sum of two ingredients: the energy for transmission and reception of packets, and the energy to wake up and beaconing.

1. Transmission and Reception. I model the energy consumption involved in each packet transmission $E_{TX} = \rho d^{\beta}$, where ρ is the required transmitted energy if

the receiver was at 1m distance, d is the average transmission distance (size of next block), and β the roll-off factor $2 \leq \beta \leq 6$. For each reception I model a fixed cost R due to the RF circuit at the receiving node. Assuming h hops, and recalling that in a time T the Source emits $T\lambda$ packets, the energy consumption associated to correctly received these packets is $E_{pck} = T\lambda \sum_{i=1}^{h} (\rho d_i^{\beta} + R)$.

2. Wake up and beaconing. Each time a node wakes up, it contributes a fixed "idle" mode energy consumption that includes also the cost of a beacon message. Call this cost E_{id}. Since I consider a probability p of a good channel for each transmission, nodes have to wake up on average 1/p times to create the effect of a single wake up. Assuming h hops and a cumulative wake up rate per block $\mu_{c,i}$, in a time T the total cost for wake ups becomes $E_{WU} = \frac{1}{p} h \mu_{c,i} E_{id}$.

Consequently, the energy consumption of a network becomes:

$$E_{tot} = T(\lambda(hR + \sum_{i=1}^{h} \rho d_i^{\beta}) + \frac{1}{p} h \mu_{c,i} E_{id}) \tag{3.3}$$

Although some of the packets are lost for collisions, in equation 3.3 I implicitly assumed all the packets getting to destination. Consequently, equation 3.3 gives a higher bound on energy consumption. As it will be clearer in Section 3.4.4, this approximation allows a manageable solution of the optimization problem.

The constrained optimization problem that I want to solve becomes:

$$Argmin_{(h,d_1,\ldots,d_h,\mu_{c,1},\ldots,\mu_{c,h})} E_{tot} \tag{3.4}$$

such that inequalities 3.1 and 3.2 are satisfied.

3.4.4 Solving the Problem

Consider equation 3.3. Since the optimization variables d_i and $\mu_{c,i}$ are separated, a first simplification to the problem can be made.

Assume I have h variables, $x_1, ..., x_h$ strictly positive and subject to the constraints $\sum_{i=1}^{h} x_i = const.$ Then, the minimum of $f(x_1, ..., x(h)) = \sum_{i=1}^{h} x_i^a$ for $a > 1$ is obtained when $x_1 = x_2 = ... = x_h$ (the level curve of $f(\underline{x})$ are tangent to the constraint plane in that point). Consequently, since $\beta \geq 2$, for any given h, the optimum block size is the same for every block. That is $d_1 = d_2 = ... = d_h = D/h$.

Furthermore, I restrict the problem to the case where the cumulative wake up rate is the same for every block. That is $\mu_{c,1} = \mu_{c,2} = ... = \mu_{c,h} = \mu_c(h)$. The intuition under this choice is that since all blocks see the same incoming traffic, having a block with a lower cumulative wake up rate, would create a bottleneck in the communication infrastructure.

These observations lead to the following consequences:

1. Consider the E2E delay constraint in equation 3.1 and call $A(h) = \sum_{i=1}^{h} \alpha_i$. Since the α_i's are i.i.d., I can apply the central limit theorem and approximate $A(h)$ with a Gaussian random variable. That is $A(h) \in N(\frac{h}{\mu_c(h)}, \frac{h}{(\mu_c(h))^2})$. Consequently, the E2E delay two sigma constraint becomes:

$$\mu_c(h) \geq \frac{h + 2\sqrt{h}}{\tau - hF} \tag{3.5}$$

   Call $Lc(h) = \frac{h+2\sqrt{h}}{\tau - hF}$, consequently $\mu_c(h) \geq Lc(h)$. Inequality 3.5 introduces the constraint $h \leq \frac{\tau}{F}$.

2. The error rate constraint becomes $(\frac{p\mu_c(h)}{\mu_c(h)+\lambda})^h \geq \Omega$. The constraint on the wake

up can be expressed as $\mu_c(h) \geq \frac{\lambda\Omega^{1/h}}{p-\Omega^{1/h}}$. Using Taylor expansion on h, the constraint on the wake up rate can be approximated as:

$$\mu_c(h) \geq \frac{\lambda}{h\ln p - \ln\Omega}h \tag{3.6}$$

Note that inequality 3.6 introduces the constraint $h \leq \frac{\ln\Omega}{\ln p}$.

Define $Ec(h) = \frac{\lambda}{h\ln p - \ln\Omega}h$, the constrained optimization problem 3.4 becomes:

$$Argmin_h T(\lambda(hR + \rho D^{\beta}h^{1-\beta}) + \frac{h}{p}max\left\{Lc(h), Ec(h)\right\}E_{id}) \tag{3.7}$$

Proposition

The optimization problem in 3.7 is convex.

Proof

Both $hLc(h)$ and $hEc(h)$ are strictly convex for $0 \leq h \leq min\left\{\frac{\tau}{F}, \frac{\ln\Omega}{\ln(p)}\right\}$ (the second derivative is strictly positive). Consequently, $max\left\{hLc(h), hEc(h)\right\}$ is convex. Furthermore, hR is always convex and $h^{1-\beta}$ is convex for $\beta \geq 2$. Consequently, E_{tot} is a convex combination in the domain $0 \leq h \leq min\left\{\frac{\tau}{F}, \frac{\ln\Omega}{\ln(p)}\right\}$, and as such it is convex. QED.

3.4.5 Algorithm

Although E_{tot} is a convex function, it is in general non differentiable and finding a closed solution is not always possible. Anyway, since I am interested in the optimal integer value of h, I can use a simple iterative algorithm.

Initialize: Evaluate $Res = E_{tot}(1)$, set $h = 2$
Step: if $(E_{tot}(h) < Res) \wedge \left(h \leq min\left\{\frac{\tau}{F}, \frac{\ln \Omega}{\ln(p)}\right\}\right)$
$Res = E_{tot}$;
$h++$
Go to **Step**;
else
Return $Res, h--$;
End;

The worst case number of iterations is $min\left\{\frac{\tau}{F}, \frac{\ln \Omega}{\ln p}\right\} - 1$.

In practice, I noticed that with 6 or less iterations the optimal number of hops is reached.

3.4.6 Distributed Adaptation Protocol

In the previous section, I showed how to determine the optimal forwarding region and cumulative wake up rate. In this section, I present a distributed algorithm that each node has to run to correctly determine its forwarding region and wake up rate so that the overall network operates at the optimal working point calculated in 3.4.5. The proposed algorithm is completely local and allows to adapt to change in the traffic rate of the application and change in the channel conditions without message overhead.

I assume that all the physical layer abstraction values are known. Consequently, for a node to solve the optimization problem in 3.7 it must know the traffic λ and the average channel condition p. These two quantities cannot be locally estimated. However, the Source knows λ and that value can be piggybacked on packets. Furthermore, if the packets are numbered, the Destination can estimate p and the value can be piggybacked on beacons.

Assume N nodes in a block. Ideally, I favor the solution of distributing the cumulative wake up rate equally between all the nodes. Calling μ_i the wake up rate of

node i, the fair solution is $\mu_i = \frac{\mu_c}{N}, \forall i = 1, ..., N$. However, a node does not know and cannot efficiently estimate the number of nodes in its block. This problem was successfully addressed in [46]. In that paper, a parallel was drawn between this problem and the fair bandwidth allocation for TCP flows and it was shown how implementing an Additive Increase and Multiplicative Decrease (AIMD) algorithm of the wake up rate of each node leads to a fair distribution of the wake up duties within a single block. I decided to follow this approach. Specifically, each node that is waiting to forward a packet, it observes the time before the first wake up in the forwarding region. Starting from this observation, it estimates the cumulative wake up rate of the forwarding region and it compares it with optimal value calculated through the iterative algorithm outlined in the previous section. If the estimated value is less than or equal to the optimal value, it communicates to the next hop to additively increase its wake up rate, otherwise it orders the next hop to multiplicatively decrease its wake up rate. The command on the wake up rate variation is piggybacked on the packet and it does not require any additional message.

INIT STATE: the node sets its $FwR = [0, Max_{Range}]$, $\mu = \mu_0$, $p = p_0$. When a packet is received, the node goes to the OP state.

OP STATE: Run the Iterative Algorithm and determine d and μ_{copt}. Set $FwR = [\frac{d}{2}, \frac{3d}{2}]$. Wait for the reception of a packet or of a beacon.

If a beacon is received, retrieve information on p, estimate μ_c of the forwarding region, if $\mu_c < \mu_{copt}$ send Additive Increase (AI) command, else send Multiplicative Decrease (MD)command. Go back to INIT STATE.

If a packet is received, retrieve information on λ, check information on wake up rate update, if AI then $\mu = \mu + \Delta$, else $\mu = \frac{\mu}{2}$. Go back to INIT STATE.

I followed the approach of [46] and implemented an exponential filter to estimate the next region wake up rate. Consequently, calling α the time observed before the

first wake up in the next region, the new estimated wake up rate is $\mu_{cNew} = b\mu_c + \frac{1-b}{\alpha}$. Simulations showed that having $b = 0.6$ is a good choice.

The introduction of the Distributed Algorithm allows each node to work independently from its neighborhood and to select the forwarding region according only to its position and the positions of Source and Destination. Consequently, each node will see its own forwarding region and the block abstraction is relaxed. Furthermore, nodes are not required to maintain a neighbor list and the death of a node is met with an individually determined increase in all its neighbors activity. Consequently, the protocol is extremely robust against topology changes such as node failures and introduction of new nodes.

3.4.7 Simulations

To validate our solution, I implemented our randomized protocol in Omnet++ [31]. I considered a scenario of a Source and a Destination placed 50 meters apart in an indoor environment. Between the two, 100 nodes are placed uniformly at random. I modeled the nodes using the physical layer parameters of the PicoRadio motes [7]. I modeled the channel behavior using the Chaotic Maps model [48]. Since I am analyzing an application where the nodes are deployed with high density and over a plane surface, I expect very good channel conditions since line of sight is usually available. I run a set of simulations where the Source was sending packets periodically at different rates with different latency and error rate requirements. The time to transfer a packet (F) was assumed $10\mu s$. I observed how the network was adapting and compared the final results to the optimal conditions calculated from 3.7.

Note that when the most stringent constraint is the error rate (as in the results in Table 3.1), the optimum number of hops does not depend on λ (see equation 3.7). Consequently, if an adequate number of nodes is placed, traffic rates on the order

λ $kpps$	τ sec	Ω	Opt h	Opt μ_c $\frac{w.u.}{\mu s}$	Obs. μ_c $\frac{w.u.}{\mu s}$	D.C.
0.1	0.1s	0.8	3	0.0022	0.0024	0.07%
1	0.1s	0.8	3	0.022	0.023	0.7%
10	0.1s	0.8	3	0.22	0.24	7%
100	0.1s	0.8	3	2.2	2.2	70%

Table 3.1. Steady state performance for different traffic rates. The last column represents the observed node duty-cycle

of several kilo packets per second can be supported. Furthermore, the deployment a high number of nodes, automatically ensures very good duty cycle performance and consequently long network lifetime. The limit to the supported traffic rate is given by the time to forward a packet, hence it is a function of the node radio bit-rate and the packet length. I believe this opens important opportunities for the introduction of this network architecture for a wide range of indoor applications from typical sensing and control networks for temperature and light management to more complex multimedia and entertainment applications. However, the periodic traffic generation and simulation framework is realistic for sensing and actuation applications, while for multimedia more complex traffic models and packet structures should be considered.

I also observed a good matching between the simulated performance and the ones predicted by our mathematical model. Specifically, no packets arrived over the deadline and the error rate constraint was always satisfied. This is an important result since it validates our methodology and allows to set the network parameters using the simplified mathematical model instead of performing extensive simulations.

To evaluate the adaptation performance of our protocol, I analyzed the transient behavior and the speed of the protocol to reach a stable solution. In Fig. 3.3, I plot the observed end to end delay for three different sets of starting conditions. An optimal one, where the initial sleeping parameters where the optimal ones, a super optimal where the nodes were initially waking up more often than necessary and a sub optimal

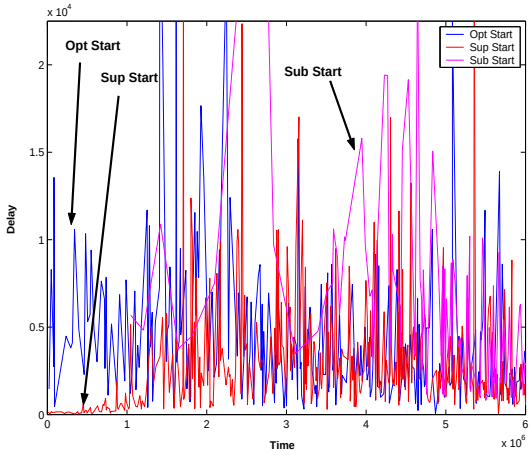

Figure 3.3. Adaptation performance for different starting conditions (time is in μs)

one where the nodes were initially lazy. Notice that the adaptation works better in the super optimal case. In the case of sub optimal starting conditions it takes some time before adaptation and the first set of packets are lost in collisions. Consequently, I suggest to set the initial μ higher than necessary during a real deployment and then let the network adapt to the optimal solution.

3.5 Example: SERAN

SERAN stands for SEmi-RANdom communication protocol, that I developed for low power clustered wireless sensor network topologies.

Most of the sensoring applications for industrial plants are characheterized by clustered topologies. For example in building automation applications, groups of sensors are deployed in specific rooms to observe quantities like temperature, humidity, or chemical leakage and report to a remote central station in a multihop fashion. In manufacturing lines, sensors are typically grouped around specific points of interest in a cell like the end of a rail or around some robots.

From a network perspective, these are all clustered topologies, and although the

size and the position of these clusters can vary significantly for different applications, this similarity in the high level structure allows us to create protocols that can be very effective over all these applications. Our approach is to leverage the only resource that is usually available in these systems and that is a reasonably high node density. In clustered topologies, different sensors are deployed to monitor the same phenomenon, and this space diversity can be exploited with targeted routing and MAC algorithms.

Without loss of generality, I present SERAN using the clustered topology of Fig. 3.4, where five clusters of sensors are deployed to perform periodic sensing and report to the Controller with a delay constraint D_{max}. The goal is to design a routing and MAC protocol that:

1. Satisfies system requirements;
2. Ensures robustness to environment variability;
3. Is energy and storage efficient;
4. Can be implemented on a large set of existing hardware platforms;
5. Has self-configuration capabilities;
6. Supports the addition of new nodes;
7. Can be extended with data aggregation algorithms.

SERAN approach can be summarized as follows:

1. I start from solutions at different layers developed for classical WSN applications (environmental and habitat monitoring) and modify them to suit our class of applications.
2. I join the different layers to create a complete protocol stack and characterize the delay and power performance of the solution with a mathematical model.

3. I use this mathematical model to set the protocol parameters so that latency requirements of the application are satisfied and energy consumption is minimized.

4. I develop a set of algorithms for initialization and maintenance of the network.

To motivate our design, I discuss the different alternatives at every step of the design flow and indicate how our solution is positioned with respect to previous work.

I assume that the Controller knows a priori the number, the position of the clusters, and how many nodes are in each of the clusters. Furthermore, the Controller has a good estimation of the amount of data generated by each cluster since the number of sensors in each cluster is known. I assume all nodes share the same communication channel and each node knows to which cluster it belongs. I believe this is a reasonable assumption because this information is available and can be hard coded in the nodes.

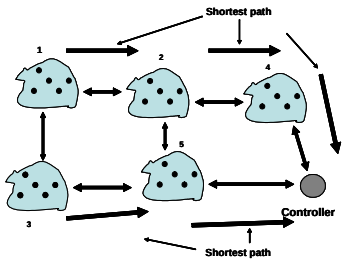

Figure 3.4. Connectivity Graph.

In [48], a characterization of wireless links in industrial environments for a 802.11b MAC link is presented that shows evidence of a bursty behavior of point-to-point links. Usually, these bursts are described using a Markovian model, but in those experiments, this class of models does not capture residual time correlation. To do so, a model called Chaotic Maps is introduced where the channel is classified in two states (a good state and a bad state), but the transition from one state to the other is a function of time spent in a state. The decision on the state transition is made based

on the solution of a chaotic system of equations (hence the name). In [49] a model for 802.11a links for industrial environments is presented and evaluated. That study shows that when a high degree of diversity is used (OFDM at 5GHz band) the error bursts are not that deep and the channel can be abstracted with an i.i.d. process (also called Bernoulli channel), where at each packet transmission there is an independent probability for the channel to be good. Considering also the fact that channel access for wireless sensor networks applications is less frequent than for typical 802.11 based applications, I believe that a Bernoulli model is the most appropriate for our study.

3.5.1 The SERAN Protocol

The protocol I propose for our application covers two layers of a classical protocol stack: routing and MAC.

Routing Algorithm

Routing over an unpredictable environment is notoriously hard. Our approach to leverage density and clusterization is to have a set of nodes within transmission range that could be candidate receivers; at least one of them will offer a good link anytime a transmission is needed.

In [45], the idea of deciding the next hop after an estimation of the links to neighboring nodes is presented. Although the estimation algorithm has very good convergence properties, the protocol shows stress when applied to fast varying links.

In [14], the idea of routing through a random sequence of hops instead of a predetermined one is introduced. In [47], the idea is further explored to reduce the overhead caused by the need of coordinating the nodes. Both approaches demonstrate that density ensures robustness even in fast varying links. In [47], an algorithm is

given for determining the optimal shape of the region from which candidate receivers should be selected.

The routing solution of SERAN is based on a semi-random scheme to reduce the overhead of purely random approaches. In SERAN, the sender has knowledge of the region to which the packet will be forwarded, but the actual choice of forwarding node is made at random. This random choice is not performed at the network layer, but it is a result of an acknowledgment contention scheme performed at the MAC layer by all the candidate receivers (see next subsection).

Consider the cluster connectivity in Figure 3.4. An arrow between two clusters means that the nodes of the two clusters are within transmission range. The first step of the SERAN routing algorithm consists of calculating the shortest path from every cluster to the Controller and generating the minimum spanning tree as in Figure 3.4.

Assume a particular node in Cluster 1 has a packet to forward to the Controller. The proposed routing algorithm works as follows on the example:

- The node that has the packet selects randomly a node in Cluster 2 and forwards the packets to it.
- The chosen node determines its next hop by choosing a node randomly in Cluster 4, and so on.

In other words, packets are forwarded to a randomly chosen node within the next-hop cluster in the minimum spanning tree to the Controller.

Hybrid MAC

The first priority for the design of our MAC is ensuring robustness against topology changes. Since nodes failure is a common phenomenon for WSN, I design a MAC that

is able to support the addition of new nodes for preserving the high level of density required to ensure robustness. This flexibility is usually obtained by using random based access schemes that may or may not support collision avoidance, depending on the radio interfaces that are used and the capability of the RF chip to support an effective clear channel assessment. In the WSN domain, an interesting example of this idea is presented in BMAC [12].

High density unfortunately introduces a large number of collisions, even if collision avoidance is supported. This drawback becomes crucial in our case because I have only one channel that can be used for communication. To reduce collisions, usually a deterministic MAC is used. A well-known deterministic approach is SMAC [11], where the network is organized in a clustered TDMA scheme.

Our MAC solution is based on a two-level semi-random communication scheme that provides robustness to topology changes and node failures typical of a random based MAC and robustness to collision typical of a deterministic MAC.

The higher level regulates channel access among clusters. A weighted TDMA scheme is used such that at any point in time, only one cluster is transmitting and only one cluster is receiving. During a TDMA cycle, each cluster is allowed to transmit for a number of TDMA-slots that is proportional to the amount of traffic it has to forward. The introduction of this high level TDMA structure has the goal of limiting interference between nodes transmitting from different clusters. The time granularity of this level is the TDMA-slot (see Figure 3.5).

The lower level regulates the communication between the nodes of the transmitting cluster and the nodes of the receiving cluster within a single TDMA-slot. It has to support the semi-random routing protocol presented in 3.5.1, and it has to offer flexibility for the introduction of new nodes. This flexibility is obtained by having the transmitting nodes access the channel in a p-persistent CSMA fashion [50]. If collision

avoidance (CA) is supported by the intended hardware platform, it can be used to improve performance. The random selection of the receiving node is obtained by multi-casting the packet over all the nodes of the receiving cluster, and by having the receiving nodes implement a random acknowledgment contention scheme to prevent duplication of the packets.

Calling CSMA-slot the time granularity of this level (see Figure 3.5), the protocol can be summarized as follows:

- Each of the nodes of the transmitting cluster that has a packet tries to multi-cast the packet to the nodes of the receiving cluster at the first CSMA-slot with probability p. If CA is supported, a clear channel assessment (CCA) for a random back off time is performed prior to transmission, and in case another transmission is detected, the node aborts the current trial to avoid collisions. If CA is not supported, the node simply transmits the packet.

- At the receiving cluster, if a node receives more than one packet, it detects a collision and discards all of them. If it has successfully received a single packet, it starts a back-off time T_{ack} before transmitting an acknowledgment. The back-off time T_{ack} is a random variable uniformly distributed between 0 and a maximum value called T_{ackmax}. If in the interval between 0 and T_{ack}, it hears an acknowledgment coming from another node of the same cluster, the node discards the packet and does not send the acknowledgment. In case of a collision between two or more acknowledgments, the involved nodes repeat the back-off procedure. At the end of the CSMA-slot, if the contention is not resolved, all the receiving nodes discard the packet.

  Note that this random back-off procedure is different from a CA procedure. This is because the nodes are already awake and listening to the channel for possible

packets and consequently such a scheme can be implemented even on platforms where performing instantaneous CCA is not supported or it is inefficient.

- At the transmitting node side, if no acknowledgment is received (or if only colliding acknowledgments are detected), the node assumes the packet transmission was not successful and it multi-casts the packet at the next CSMA-slot again with probability p. The procedure is repeated until transmission succeeds.

In this approach, nodes need to be aware only of the next-hop cluster connectivity and do not need a neighbor list of next hop nodes. I believe this is a great benefit because, while neighbor lists of nodes are usually time-varying (nodes may run out of power and other nodes may be added) and hence, their management requires significant overhead, cluster based connectivity is much more stable. In Section 3.5.3, I explain how to deal with permanent fades between clusters within transmission range.

In [47], it is shown how a similar acknowledgment contention scheme reduces significantly the packet duplication effect. However, I still cannot guarantee that duplicate packets are not generated. This may happen if a receiving node does not hear the acknowledgment sent by another node in the same cluster. Although these duplicate packets are detected at the Controller, they still create an extra amount of traffic in the network.

In most of the proposed MAC algorithms for WSN, nodes are turned off whenever their presence is not essential for the network to be operational. GAF [15], SPAN [13] and S-MAC [11] focus on controlling the effective network topology by selecting a connected set of nodes to be active and turning the rest of the nodes off. These approaches require nodes to maintain partial knowledge of the state of their individual neighbors, thus requiring additional communication.

Similar to this approach, our duty-cycling algorithm leverages the MAC properties and does not require extra communication among nodes. During an entire TDMA cycle, a node has to be awake only when it is in its listening TDMA-slot or when it has a packet to send and it is in its transmitting TDMA-slot. For the remainder of the TDMA cycle, the node radio can be turned off.

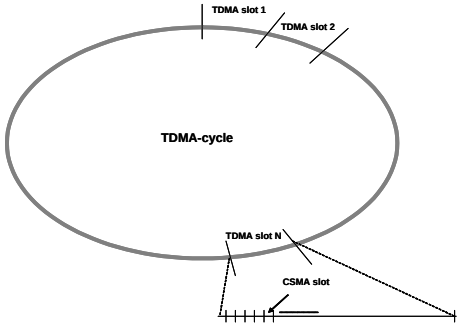

Figure 3.5. TDMA-Cycle representation.

Organization of the TDMA-cycle

Referring to Figure 3.4, assume for now that the average generated traffic at each cluster is the same.

According to our shortest cluster path routing solution, packets are transferred cluster-by-cluster along the shortest path until they reach the Controller. Consequently, clusters close to the Controller, have a higher traffic load since they need to forward packets generated within the cluster as well as packets coming from upstream clusters. In the example of Figure 3.4, the average traffic intensity that cluster 4 experiences is three times the traffic intensity experienced by cluster 1. Consequently, I can assign one transmitting TDMA-slot per TDMA-cycle to cluster 1, two transmitting TDMA-slots to cluster 2 and three transmitting TDMA-slots to cluster 4. Similarly, on the other path, the number of associated TDMA-slots per cluster can be assigned. Therefore, assuming I have P paths and calling B_i the number of clusters

in the $i-th$ path, I have a total of

$$T_f = \sum_{i=1}^{P} B_i(B_i+1)/2$$

TDMA-slots per TDMA-cycle. For the remaining of the paper, I call T_f the topology factor. As I will show later, T_f is an important parameter that abstracts the network layout and connectivity.

Notice that in case the traffic generated is not the same for each cluster, the relative number of TDMA-slots per TDMA-cycle for each cluster can be easily recalculated changing the weights in the TDMA scheme. For sake of simplicity, I outline our solution for a case with uniform traffic rate. The extension to a more generic traffic pattern is straightforward.

Once I decide the number of TDMA-slots per TDMA-cycle for each cluster, I need to decide the scheduling policy for transmitting and receiving. I consider an interleaved schedule (Figure 3.6). For each path, the first cluster to transmit is the closest to the Controller (cluster 4). Then cluster 2 and cluster 4 again. Then cluster 1, 2 and 4, and similarly on the other path. This scheduling is based on the idea that evacuating the clusters closer to the Controller first, I minimize the storage requirement throughout the network.

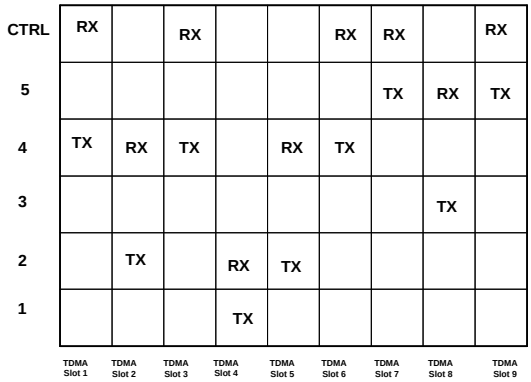

Figure 3.6. Scheduling: clusters close to the Controller are evacuated first.

3.5.2 Protocol Parameter Determination

In this section, I explain how access probability, slot duration, and storage are determined to satisfy application requirements (successful transmission probability and maximum delay), and optimize for power consumption. In general, for applications like vibration monitoring, an outage packet probability below 5% is a typical requirement. A packet could be in outage because it arrives over the latency requirements or because it was dropped by some node that reached its buffering limit.

First, I show how to set the access probability parameters, then I show how to set the storage requirements so that the probability of dropping packets is negligible. Finally, I show how to set the duration of a TDMA-slot to offer good latency performance and optimize for power consumption. First, I carry on this analysis for the case in which collision avoidance is not supported, and then I show how the model is modified to account for collision avoidance.

Access Probability

Call k the number of packets that the cluster has to evacuate at the beginning of a transmitting TDMA-slot. I consider the worst case scenario for collisions, i.e., when the k packets are distributed over k different nodes. I model the channel as a Bernoulli variable with parameter c. Notice that this parameter is close to 1 since it abstracts the cluster based connectivity, meaning the probability that at least one node in forwarding cluster is able to complete a succesfull communication.

When there are k packets to be forwarded the probability of having a successful transmission at the first CSMA-slot is $P[success|k] = ckp(1-p)^{k-1}$.

Assume the transmission was successful. The cluster now has $k-1$ packets to

forward. This time the probability of a successful transmission at the first CSMA-slot is $P[success|k-1] = c(k-1)p(1-p)^{k-2}$.

Again if the transmission was successful the cluster has $k-2$ packets to forward and so on. This allows us to represent the cluster behavior as a Discrete Time Markov Chain (DTMC) where the state is the number of packets that still need to be forwarded (see Figure 3.7).

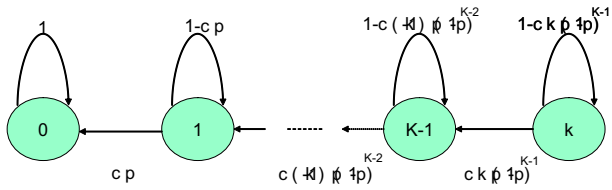

Figure 3.7. Discrete Time Markov Chain model.

This DTMC has an absorbing state in 0 which is the steady state solution of the chain. This means that the state 0 is eventually reached with probability one. I am interested in calculating the expected time (i.e., expected number of steps) to reach the absorbing state starting from a given state between 1 and k. This is equivalent to determining the average number of CSMA-slots required for forwarding a number of packets between 1 and k.

Since expectation is a linear operator and using the fact that the chain can advance only one step at a time, the expected time to absorption starting from a state k is equivalent to the sum of the expected time to transition from state k to state $k-1$ plus the expected time to transition from state $k-1$ to $k-2$ and so on until state 0 is reached. Given that the chain is in state j, the mass distribution of the required number of steps to transition to state $j-1$ follows a geometric distribution of parameter $1-cjp(1-p)^{j-1}$. Consequently, the expected time to transition from state j to state $j-1$ is:

$$\tau(j) = \frac{1}{cjp(1-p)^{j-1}}$$

Calling τ_k the expected number of steps to reach the absorption starting from state k, I have

$$\tau_k = \sum_{j=1}^{k} \tau(j) = \sum_{j=1}^{k} \frac{1}{cpj(1-p)^{j-1}} \tag{3.8}$$

Considering Equation (3.8), I notice that for each transition from state j to $j-1$ the access probability that minimizes the transition time is $p_j = 1/j$. With this choice, the expected number of transmission attempts for each slot is exactly one. This is the choice that maximizes channel utilization without incurring into excessive collisions. I now present two strategies for setting up the access probability given the number of packets that need to be transmitted at the beginning of the TDMA-slot. In the following subsections, I present the latency, storage, and energy performance of the two strategies, and in 3.5.2, I present a comparison of the two.

Fixed Choice

According to this choice, the access probability is the same for each node and it remains the same during the whole TDMA-slot duration.

It is possible to show that finding a closed form expression for the p_k that minimizes tau_k in Equation (3.8) is a non-trivial problem. However, the expression is a convex function in p. Indeed, (3.8) is a non negative weighted sum of the functions

$$\frac{1}{cpj(1-p)^{j-1}} \text{ for } j = 1 \ldots k \tag{3.9}$$

The functions (3.9) are a convex ones, since by taking the first derivative there is only

the following critical point in the interval $[0, 1]$

$$p_j = \frac{1}{j} \tag{3.10}$$

and the second derivative of (3.9) is strictly positive.

Although (3.8) is a convex function, its first derivative does not help to get the minimum in a closed form. However, the convexity allows us to use the bisection algorithm [51], which finds iteratively the numerical value minimizing (3.8) with any desired precision. Note that the algorithm is not computational demanding, and can be easily implemented on sensor nodes. If the initial guess used to feed the algorithm is good, the convergence to the optimal value minimizing (3.8) is very fast.

However, such an optimal selection of p_k would necessarily be higher than $1/k$. Since the most critical stage in our DTMC model, in terms of collision probability, is from state k to $k-1$, such an access probability would likely lead to a high number of collision at the beginning of the TDMA-slot. Consequently, I select the access probability $p_k = 1/k$ for the whole duration of the slot, which is suboptimal in terms of expected fowarding time, but it ensures that at the beginning of the TDMA-slot the expected number of transmission attempts for each CSMA-slot is one. The result is that the channel is highly utilized, while as the time progresses the channel will be less and less utilized.

The expected absorption time is:

$$\tau_k = \frac{k}{c} \sum_{j=1}^{k} \frac{1}{j(1-1/k)^{j-1}} \tag{3.11}$$

A close form solution for τ_k in this scenario is not easy to find, but I can find some useful upper and lower bounds.

Proposition 3.5.1 *For the fixed choice, for high k, the expected time to forward all the packets is bounded by:*

$\alpha_{flb} k \leq \tau_k \leq \alpha_{fub} k \ln k$

where α_{flb} and α_{fub} are constants.

Proof

Looking at Equation (3.11), I notice that a lower bound is given by the case in which all the expected transition times are the same as the expected transition time of the first transition (when the channel is optimimally utilized). This expected transition time is $\frac{1}{c(1-1/k)^{k-1}}$ and since

$$Lim_{k->\infty} (1 - 1/k)^k = e^{-1}$$

I can find a lower bound $\tau_{klb} = \frac{e}{c} k$.

The upper bound can be found considering that

$$\tau_k \leq \frac{k}{c} \left(\frac{k}{k-1} \right)^{k-1} \sum_{j=1}^{k} 1/j \leq \frac{e}{c}(k-1) \sum_{j=1}^{k} 1/j$$

The k-th harmonic $H_k = \sum_{j=1}^{k} 1/j$ grows as fast as $\ln k$ and it is upper bounded by $H_k < 1 + \ln k$. Consequently, for high k I have:

$$\tau_k \leq \frac{e}{c}(k-1)(1 + \ln k) \leq \gamma \frac{e}{c} k \ln k$$

For any constant $\gamma > 1$.

QED

Because of the interleaved schedule, each cluster evacuates all the locally generated packets before receiving the ones generated from the one-hop upstream cluster. First, I need to ensure that the expected time for the evacuation of the packets in a cluster is less then or equal to the duration of a TDMA-slot. If this does not happen,

packets keep accumulating and storage capacity is reached very soon with catastrophic consequences on performance.

I consider the upper bound for the forwarding time using $\gamma = 1$ to simplify our analysis. As I show in Figure 3.8 this is already a good enough upper bound, that is $\tau_k = \frac{e}{c} k \ln k$.

Let S be the duration of a TDMA-slot, Δ the duration of a TDMA-cycle, and λ the packet generation rate for each cluster.

Since during a TDMA-cycle each cluster generates $\lambda\Delta$ packets, I need to ensure:

$$S \geq \frac{e}{c}\lambda\Delta \ln(\lambda\Delta) \tag{3.12}$$

Recalling that $T_f = \left(\sum_{i=1}^{P} B_i(B_i+1)/2\right)$ and $\Delta = ST_f$, I can simplify the previous equation in:

$$S \leq \frac{e^{\frac{c}{eT_f\lambda}}}{\lambda T_f} \tag{3.13}$$

which, given a traffic generation λ sets a constraint on the maximum duration of a TDMA-slot.

As it will be clearer in Section 3.5.2, it is interesting to rewrite the previous equation as:

$$\lambda \ln(\lambda S T_f) \leq \frac{c}{eT_f} \tag{3.14}$$

Adaptive Choice

According to this choice, the access probability is increased every time a there is a state transition in such way that for each transition from state j to $j-1$, the access

probability goes from $1/j$ to $1/(j-1)$. As I already explained, this is the choice that minimizes the forwarding time and maximizes the throughput of the cluster.

The expected time to forward all the packets in this case is:

$$\tau_k = \frac{1}{c} \sum_{j=1}^{k} \frac{1}{(1-1/j)^{j-1}} \tag{3.15}$$

Proposition 3.5.2 *For the adaptive choice, for high k, the expected time to forward all the packets is bounded by:*

$\alpha_{alb} k \leq \tau_k \leq \alpha_{aub} k$

where α_{alb} and α_{aub} are constants.

Proof Looking at Equation (3.15), I notice that the slowest trasnition is the first one. Consequently, I can find an upper bound considering the case of all transitions taking the expected time of the first transition. Similarly to the case of the fixed choice, for high values of k I have:

$\tau_{kub} = \frac{e}{c} k$.

A lower bound can be found considering a successfull transition at every CSMA-slot This means $\tau_{kLB} = k$.

QED

Considering the upper bound bound, I can now derive some design constraints in the same way as I did in the fixed choice case:

$$S \geq \frac{e}{c} \lambda \Delta \tag{3.16}$$

Since $\Delta = ST_f$, I can obtain a limit for the maximum sustainable traffic:

$$\lambda \leq \frac{c}{eT_f} \tag{3.17}$$

Notice that in this case the constraint is only on the traffic generation rate and it depends on the topology and connectivity of the network, abstracted by the topology factor, and not on the TDMA-slot duration. Note also that, given a number of cluster, the configuration that minimizes T_f (and maximizes the maximum sustainable traffic) is a star topology, where each cluster is a single hop to the controller. Conversely, the worst configuration is a linear topology, where all the clusters are in a single multi-hop chain.

In case the maximum traffic condition is not satisfied even using the adaptive choice, a slot reuse mechanism can be introduced to obtain an operational network. This means to have more then one cluster transmitting and receiving during the same time-slot, provided that they are far enough apart. This solution can significantly increase the throughput of the network, but it is also much more power expensive. Consequently, it should be considered only if the stability requirement cannot be satisfied, otherwise a "lazy" network is preferable.

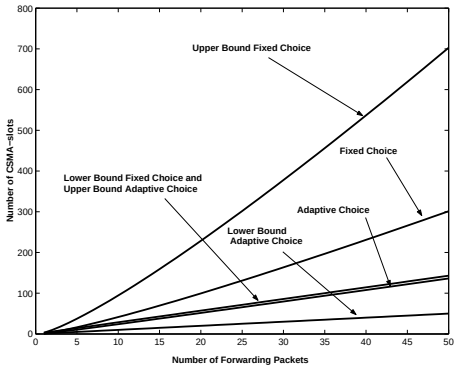

Figure 3.8. Expected forwarding time for fixed and adaptive parameter choice

Storage Requirements

I set the buffering requirement for each node based on the worst case scenario, i.e., when all the packets of a transmitting cluster are forwarded to the same node in the receiving cluster.

Assume I have N nodes in the receiving cluster and allow the nodes a storage capacity of $\lambda\Delta$ packets. If the requirement on the maximum sustainable traffic is satisfied, the probability of overflowing is bounded by the probability that one of the nodes reaches capacity during a receiving TDMA-slot.

This probability can be approximated by the probability that all the packets that are on average generated by a cluster $\lambda\Delta$ are forwarded to the same node multiplied by the possible choose of N nodes:

$$P_{overflow} \leq P_{capacity} \approx N(1/N)^{\lambda\Delta} \tag{3.18}$$

Although the value of $\lambda\Delta$ is determined by the application, in most of the cases, it is greater than 10. Consequently, the probability of overflow is negligible. However since it is different from zero, I must offer a scheme that will guarantee the network to continue operation even in this rare case. In the event of an overflow, in our scheme, the node will drop the oldest packets first.

Latency

The clusters that experience the highest delay are the furthest from the Controller. I want to have the delay of packets coming from those clusters less than or equal to a given D_{max}, the requirement set by the application (see Section 6.1.1).

Consider the packets generated in cluster 1. These packets have to wait, in the worst case, a TDMA-cycle before the first opportunity to be forwarded to cluster

2. Assuming for now that all the packets of a cluster are forwarded within a single TDMA-slot, then it takes 3 additional TDMA-slots to reach the Controller.

Generalizing to the case of P paths and B_i clusters per path, the worst case delay is:

$$D = \Delta + S \max_{1,..,P} B_i = S \left(\max_{1,..,P} B_i + T_f \right) \tag{3.19}$$

Consequently, the requirement on S is:

$$S \leq S_{max-d} = \frac{D_{max}}{\max_{1,..,P} B_i + T_f} \tag{3.20}$$

If during a TDMA-slot not all the packets are forwarded, a latency over the deadline is observed. I can model this phenomenon using the DTMC model introduced in 3.5.2. I want to evaluate the probability that the time to forward $\lambda\Delta$ packets exceeds the duration of a TDMA-slot.

Using the Central Limit Theorem, I can model the distribution of the time to forward $\lambda\Delta$ packets as a normal variable whose mean and variance is given by the sum of the expected times and variances to advance a step in the chain. Call T_{ev}, the time to evacuate $\lambda\Delta$ packets and call m_{ev} and var_{ev} its mean and variance. Consequently, the time T_{ev} to evacuate a cluster can be modeled as $T_{ev} \in N(m_{ev}, var_{ev})$, where in case there is no collision avoidance I have:

$$m_{ev} = m_{evnca} = \sum_{j=1}^{\lambda\Delta} \frac{1}{cpj(1-p)^{j-1}} \tag{3.21}$$

$$var_{ev} = var_{evnca} = \sum_{j=1}^{\lambda\Delta} \frac{cpj(1-p)^{j-1}}{(1-cpj(1-p)^{j-1})^2} \tag{3.22}$$

Consequently, the probability of not forwarding all the packets during a given TDMA-slot can be approximated by:

$$P[T_{ev} \geq S] \approx \frac{1}{2} erfc(\frac{S - m_{ev}}{\sqrt{var_{ev}}}) \tag{3.23}$$

Although it is not possible to find a closed form solution to 3.23, as I will show in Section 4.3.2, the requirement expressed in 3.20 usually ensures an outage probability well below 5%.

Energy Consumption

I am now interested in determining the total energy consumed by the network over a period of time. The energy cost is given by the contribution of the energy spent for transmissions, the energy spent to wake up and listen during the listening cluster-slots, and the energy spent for the clear channel assesment procedure in case collision avoidance is supported. I consider the energy spent for receiving a packet together with the energy consumption for listening.

The energy consumption for listening for a time δ is given by the sum of a fixed cost (the wake-up cost) plus a time-dependent cost (listening cost):

$$E_{ls} = R + W\delta \tag{3.24}$$

During a TDMA-cycle, nodes in cluster 1 never wake up for listening, nodes in cluster 2 wake up once for listening, nodes in cluster 4 wake up twice for listening, and so on.

Assume that there are N nodes per cluster, and that all nodes wake up in their listening TDMA-slot. During a given TDMA-cycle, the total number of wake ups is:

$$N_{WU} = N\left(\sum_{i=1}^{P} B_i(B_i - 1)/2\right) \tag{3.25}$$

To determine the energy spent for transmissions, I need to derive the average number of attempted packet transmissions during a TDMA-cycle. In case collision avoidance is not supported, I can use the DTMC introduced in 3.5.2.

Proposition 3.5.3 *For high vales of the number of packets accumulated in a cluster*

during a TDMA-cycle ($\lambda\Delta$), the expected number of attempted transmission is a linear function with the respect to the number of packets to transmit.

Proof

I model the number of attempted transmissions for each transition as the average number of nodes attempting to transmit during a slot multiplied by the average number of slots required for that transition. Assuming k packets to forward, the average number of attempted transmission during a TDMA-cycle can be modeled as:

$$N_{txNca} = (num_slots/cycle) \sum_{j=1}^{k} \frac{pj}{cpj(1-p)^(j-1)} \tag{3.26}$$

1. Case Fixed Choice

   Since $p = 1/k$, Equation(3.26) can be simplified as:

   $$N_{txNca} = T_f \frac{1}{c}(k-1)\left((1-1/k)^{-k} - 1\right)$$

   For high values of k, $\left((1-1/k)^{-k} - 1\right) \approx (e-1)$ and $k - 1 \approx k$.

   Since there is an average of $\lambda\Delta$ packets to be forwarded at every slot, I can write:

   $$N_{txNca} = \frac{T_f}{c}(e-1)\lambda\Delta \tag{3.27}$$

2. Case Adaptive Choice

   In this case the product pj at the numerator of Equation (3.26) is always equal to one, and the expected number of attempted transmission is equal to the expected number of steps required to forward all the packets. In Section 3.5.2, I showed that this number is a linear function on the number of initial packets whose slope is between 1 and e/c.

Consequently, I can write:

$$N_{txNca} = A_{nca}\lambda\Delta \tag{3.28}$$

QED.

The number of acknowledgment transmissions is equal to the number of successful packets:

$$N_{Ack} = T_f\lambda\Delta. \tag{3.29}$$

Calling E_{Tx} the energy consumption for the transmission of a packet and E_{Ack} the energy consumption for the transmission of an acknowledgment, the total energy consumption during a time $T \gg \Delta$ for the non collision avoidance case is:

$$\begin{aligned} E_{tot} &= \frac{T}{\Delta}[N_{txNca}E_{Tx} + N_{Ack}E_{Ack} + \\ &\quad N_{wu}R + N_{wu}WS] \\ &= TA_{nca}\lambda E_{tx} + T\lambda T_f E_{Ack} + \\ &\quad \frac{TN_{wu}}{ST_f}R + \frac{TN_{wu}}{T_f}W \end{aligned} \tag{3.30}$$

Since E_{Tx}, E_{Ack}, R, W are the parameters that characterize the physical layer, λ, B, N are given by the application, and T_f, N_{wu} depend only on the network topology, the only variable in Equation (3.30) is S.

Since, from equations (3.14) and (3.20), I have

$S \leq S_{max} = min\{S_{max-tr}, S_{max-d}\}$

and $E_{tot}(S)$ is a monotonically decreasing function of S, the optimal working point is $S = S_{max}$.

In this section, I present the modifcations to the performance analysis equations that I have just determined in case collision avoidance is supported.

Maximum Sustainable Traffic with CA

Again, the behavior of the transmitting cluster can be characterized by a DTMC whose state is given by the number of packets that remain to be forwarded. Differently from the non CA case, when collision avoidance is used, the probability of transitioning from a state j to a state $j-1$ is higher. This results from the fact that even if two or more nodes decide to transmit, the CA procedure is likely to avoid collisions and allow at least one packet to be transmitted succesfully.

Altough very small, the probability of a collision is still non zero and it is associated to a failure of the CA mechanism. For instance, if TinyOS [29] is used to program the hardware platform, such a failure may happen if in between the posting and the execution of a sending task of a node, another node starts its transmission. I call ϕ the probability of such a failure when two nodes are involved.

Consequently, the probability of transitioning from state j to $j-1$ in a given CSMA-slot can be modeled by the probability of having at least one node trying to access the channel multiplied by the probability that the other nodes do not interfere with the first that transmits. That is

$P[success|j] = c(1-(1-p)^j)(1-\phi)^{pj-1}$

Following the same procedure as the non CA case, I can model the average time to empty the cluster using:

$$\tau_k = \sum_{j=1}^{k} \frac{1}{c(1-(1-p)^j)(1-\phi)^{pj-1}} \tag{3.31}$$

In case the fixed choice is selected, the access probability is $1/k$. Using the same reasoning as in the non CA case, I can find an upper and lower bound for τ_k.

The lower bound becomes:

$$tau_{kLB} = \frac{e}{c(1-\phi)(e-1)}k$$

while the upper bound does not improve from the non CA case. Consequently, the constraint on maximum sustainable traffic and duration of a TDMA-slot remains the same as in the non CA case.

In case the adaptive choice is selected, the upper bound becomes $tau_{kUB} = \frac{e}{c(1-\phi)(e-1)}k$, while the lower bound remains the same. As a conseqeunce the constraint on the maximum sustainable traffic is slightly relaxed:

$$\lambda_{max} \leq \frac{c(1-\phi)(e-1)}{eT_f}$$

Delay with CA

The constraint on the maximum duration of the TDMA-slot does not change. What it changes is the mean and standard deviation of the Normal distribution that abstracts the distribution of thetime to empty a cluster. Specifically:

$$m_{ev} = m_{evca} = \sum_{j=1}^{\lambda\Delta} \frac{1}{c(1-(1-p)^j)(1-\phi)^{pj-1}}$$

and

$$var_{ev} = var_{evca} = \sum_{j=1}^{\lambda\Delta} \frac{c(1-(1-p)^j)(1-\phi)^{pj-1}}{(1-c(1-(1-p)^j)(1-\phi)^{pj-1})^2}$$

Energy Consumption with CA

The difference with respect to the non CA case is only in the number of attempted transmissions, and in the number of clear channel assesments. Ignoring the collision events, the number of attempted transmission can be easily modeled with the number of successful transmissions:

$$N_{txca} = T_f \lambda \Delta \tag{3.32}$$

Modeling the number of channel assesments, is similar to modeling the number of attempted transmissions when collision avoidance is not used and a similar reasoning may be used involving the manipulation of the relative DTMC. However a more simple model can be obtained, neglecting the collisions and considering the average number of transmission tries for each step. Consequently, I can write:

$$N_{cca} = \frac{T_f}{c} \sum_{j=1}^{\lambda\Delta} pj \tag{3.33}$$

Equation (3.33) becomes $N_{cca} = \frac{T_f}{2c}(\lambda\Delta + 1) \approx \frac{T_f}{2c}\lambda\Delta$ in case of fixed choice, and $N_{cca} = \frac{T_f}{c}\lambda\Delta$ in case of adaptive choice. In any case, I can model the expected number of clear channel assesments as a linear function:

$$N_{cca} = A_{ca} \lambda \Delta \tag{3.34}$$

Calling t the fixed duration of a CSMA-slot, I have:

$$
\begin{aligned}
E_{tot} &= \frac{T}{\Delta}\left[N_{txca}E_{Tx} + N_{Ack}E_{Ack} + \right. \\
&\quad \left. N_{wu}(R + WS) + N_{cca}(R + Wt)\right] \\
&= TT_f\lambda(E_{tx} + E_{ack}) + \frac{TN_{wu}}{ST_f}R + \\
&\quad \frac{T(N_{wu}}{T_f}W + TA_{ca}\lambda(R + Wt) \qquad (3.35)
\end{aligned}
$$

Also in this case I see that the energy consumption is a monotonically decreasing function of S, hence the optimal working point is $S = S_{max}$.

Comparison of Access Strategies

Comparing the traffic constraints in equations (3.14) and (3.17), it can be seen that the constraint relative to the fixed choice is more stringent. One way to interpret these results is that in a network there is a limit on the sustainable traffic which is given by the network topology and represented by the topology factor T_f. Furthermore, if the fixed choice is selected, the constraint becomes more stringent as the TDMA-slot increases.

Consequently, given a traffic to support, the selection of the fixed choice may limit the capability of extending the duration of the TDMA-slot (unless the constraint imposed by the latency requirement is the most stringent). As I showed in Section 3.5.2, this has a reverse impact on the power performance of the overall solution.

The adaptive choice is more efficient and it allows for a higher throughput. However, such a strategy is more difficult to implement in a distributed fashion because nodes may not be aware of the fact that other nodes completed a succesfull transmission, and there is no way to tell them without incurring into major overhead costs. The best way to implement this strategy is to have each node automatically update its access probability evaluating the expected time to complete a transition in the

chain. To do this, the node must be able to compute each term of the summation in Equation (3.8). Failure to compute those fractions, or lack of synchronization among the nodes may have a reverse impact on the efficiency of the solution and create either too many accesses, hence having more collisions, or too few access, hance wasting bandwidth. In the mathematical analysis I did not consider these events.

Since I decided to set the access probability in such a way that the expected number of attempted transmissions for a CSMA-slot is at most one, collision avoidance procedures do not improve performance dramatically. However, the greatest benefit is given by the extra robustness against inefficiencies in the implementation of the adaptive choice. That is a result of the fact that collision avoidance notoriously help stabilizing CSMA protocols when bandwidth utilization approaches the limit.

For all these reasons, I reccomend to use the adaptive choice only when the fixed choice is not good enough to serve the application requirements and the selected hardware platforms support an effective collision avoidance.

Optimizing the Protocol

In Section 3.5.1, I mentioned the problem of duplicate packets that can happen in our multi-cast scheme. This phenomenon can be simply modeled by introducing a variable ν that represents the probability of having a duplicate packet in each transmission. To consider this effect I just need to substitute λ with $\overline{\lambda} = \lambda(1 + \nu)$ in the previous equations. In [47], our acknowledgment contention scheme is proven to reduce ν to 0.1.

Further power saving can be obtained having only a subset of nodes per cluster waking up for their listening duty. The energy saving comes from three factors:

- The impact of the energy consumption due to listening decreases

- If the packets are forwarded to a smaller number of nodes, then the number of collisions in the following transmitting TDMA-slot is reduced. Assume only M out of N nodes wake up. In this case the number of attempted transmissions is no longer a constant, but a monotonically decreasing function of M.

- Since only few nodes are accumulating upstream packets, it is possible to implement efficient data-aggregation algorithms.

As already mentioned, nodes closer to the data collector have a higher workload. As a consequence, these nodes would be subject to early energy depletion with catastrophic consequences for the network lifetime. This problem is typical of single sink networks and not specifically related to our solution. The best way to deal with this issue is implementing some sort of packet aggregation algorithm.

Because of its modularity, SERAN can be extended and integrated with existing packet aggregation algorithms. There are two different strategies for packet aggregation: the first is to process the information of more than one packet and to create a single one; the second, more similar to data compression, tries to efficiently merge the payloads of more than one packet so that the overhead of the header is minimized. The approaches in [36] and [52] are examples of the first strategy, while [53] is an example of the second. Also in [53], it is shown how the two strategy can be combined to achieve maximum gain. Because of its modularity, SERAN is able to support all these algorithms.

Since having a packet aggregation procedure decreases the increment of traffic for clusters closer to the data collector, the number of TDMA-slots dedicated to those cluster decreases, making the design of the final SERAN solution even simpler. Furhtermore, a reduced number of TDMA-slots per TDMA-cycle will increase the maximum sustainable traffic for that topology.

However, having more nodes awake ensures robustness against fades but, if the

number of nodes per cluster is large enough, this extra optimization can be explored. Assume I need to wake up an average of M out of N nodes, an efficient and distributed implementation is obtained by having each node waking up at the beginning of its listening TDMA-slot with probability M/N.

In [15], [11], [46], and [13], alternatives solutions are proposed that can be employed to obtain this level of optimization. The flexibility of SERAN allows once more the integration of those techniques.

3.5.3 Operation of the Network

In this section, I introduce a token passing procedure that: allows the network to initialize and self configure to the optimal working point calculated in Section 3.5.2, ensures robustness against clock drift of the nodes, and allows for the addition of new nodes;

A token is a particular message that carries the information on the duration of a TDMA-slot and TDMA-cycle (S and Δ), the transmitting and receiving schedule of a TDMA-cycle, a synchronization message carrying the current execution state of the TDMA-cycle.

Note that once the information on cluster location is given to the Controller, the Controller has all the information to calculate the optimal set of parameters as in Section 3.5.2. Consequently, the controller is able to generate a token before the network starts operating. Notice also that once a node receives a token message, it is able to synchronize with the rest of the network and has all the information to work properly.

Our network initialization algorithm works as follows:

1. When the network starts, each node is awake and listening. The node remains in this state and cannot transmit until it receives a token.

2. The first transmission comes from the Controller. The Controller multi-casts a token to all the nodes of one of the connected cluster. In our example, assume the selected cluster is cluster 4.

3. Nodes of cluster 4 read the information on scheduling and duration of TDMA-slot and TDMA-cycle. Assume the scheduling is the one in Figure 3.6.

4. Nodes of cluster 4 start transmitting their packets to the Controller with the modalities indicated in the token.

5. At the end of the TDMA-slot, all the nodes of cluster 4 listen to the channel and start a random back-off counter. When the counter expires, if no other node sent a token, they broadcast it. Nodes in cluster 2 see the token and start behaving according to the scheduling algorithm. After they transmit their packets to cluster 4, one of them broadcast a token so that nodes in cluster 1 can hear it.

6. After the first branch of the routing tree is explored, the Controller sends a token to cluster 5, the new branch is explored, and so on.

7. The token passing procedure continues even in the following TDMA-cycles.

Call c_{same} the average probability of having a good channel between nodes of the same cluster, c_{neigh} the average probability of having a good channel between nodes of neighboring clusters, and n_{TXi} the number of transmitting TDMA-slots per TDMA-cycle of cluster i. According to our token passing procedure, the probability that a node in cluster i does not receive a token in a TDMA-cycle can be approximated as:

$$P_{NoToken} \approx [(1 - c_{same})(1 - c_{neigh})]^{n_{TXi}}$$

In Section 4.3.2, I show that this is enough to ensure a rapid configuration of new nodes and robustness against clock drifts.

The routing solution described in Section 3.5.1 is designed to cope with fast time-varying channels and not with permanent fades between clusters. This is the case in which a metal object is interposed (permanently or for a long time) between two clusters, hence cutting off their communication. This phenomenon is detected by the Controller that does not receive packets (or receives too little) coming from a particular cluster (or set of clusters). When this event occurs, the Controllers acts as follows:

1. It recomputes a minimum spanning tree, without considering the corrupted link and generate a new scheduling and protocol parameters.

2. For the following 5 TDMA-cycles it sends a token with a message to void the current scheduling.

3. It reinitializes the network sending a token with the new optimal parameters.

This re-initialization will happen more often at the beginning of the network life-cycle, but once the corrupted links are detected, it will be less and less frequent.

In the next chapter, I discuss a case study of a building monitoring application using SERAN as the underlying coomunication protocol, and I show how the parameters are selected and the final implementation obtained.

3.6 Example: IEEE 802.15.4

As we mentioned in the previous chapter, recently there has been a lot of attention in the industry community to develop low power standardized protocols. ZigBee is the most notable example in this direction, and the IEEE 802.15.4 is the MAC standard that supports the ZigBee network layer. I briefly describe some aspects of the IEEE 802.15.4 that are of interest to our case study. I refer to [16, 17] for a more detailed overview of the standard.

The IEEE 802.15.4 standard describes two different operation modes for low power WSN: the beacon mode and the no beacon mode or un-beaconed 802.15.4. The unbeaconed mode is currently used by ZigBee protocol stack (both 2004 and 2006 versions) and it is the one I consider.

An IEEE 802.15.4 unbeaconed network is charachterized by a parent, called coordinator, and a set of children. The coordinator is in general a ZigBee router and its radio is always on. The children can be either Zigbee routers or simple end devices that are allowed to turn off their radios for most of the time. I consider a network formed only by a coordinator and N end devices.

An end device can communicate only with a coordinator. There is no scheduled parent/child link so transmissions can be considered asynchronous using a CSMA/CA procedure with exponential back-off and retransmission in case of failed transmission attempt. Consequently, if an end device needs to push a data to the coordinator it simply wakes up and transmits, while if the end device needs to receive a query from a node, it wakes up and polls its coordinator. In this case, the coordinator buffers the packets for the sleeping child and waits for its wake up and polling phase.

The polling mechanism can be described as follows: a polling primitive is called periodically by the sleeping nodes (corresponding to a MLME-POLL primitive as

defined in the 802.15.4 specification) that issue a data request packet to the corresponding parent. In case of a frame to be received by the end device, the parent replies with an acknowledgement with frame pending subfield set to true. After the reception of this frame, the end device remains awake to wait for a data packet that comes from the parent. After receiving the packet, the end device signals the reception of the packets.

This wake up and poll procudure allows end devices to preserve battery life controlling the polling rate parameter. Latency also depends on the polling rate and in the next subsection I model this trade-off.

3.6.1 Mathematical Model and Parameters Determination

I present a mathematical model to describe the delay distribution performance of the unbeaconed IEEE 802.15.4 protocol. Specifically, I model the distribution of the E2E delay of a query, from the time it is originated, to the time in which a data is received by the controller. This delay is composed by the time that the controller must wait before the sensor wakes up and pulls the query, the time to reply to the query with the required data, plus the occasional extra delay due to collisions, and retransmissions.

When a node wakes up, it tries to access the channel and reach the controller to see if there are queries waiting for it. However, if the channel is occupied, it starts the random back off procedure outlined in the standard. Call $s-1$ the number of unsuccesfull attempts, and $MaxAtt$ the maximum number of attempted transmission before discarting the packet. The distribution of the E2E delay associated to the node j is:

$$P[D_j \leq \tau] = \sum_{i=1}^{MaxAtt} P[D_j \leq \tau | s = i] P[s = i] \tag{3.36}$$

which can be simplified as:

$$P[D_j \leq \tau] = P[D_j \leq \tau | s = 1] P[s = 1] + \sum_{i=2}^{MaxAtt} P[D_j \leq \tau | s = i] P[s = i] \tag{3.37}$$

The probability of being succesfull in a transmission attempt is given by the probability of no other node to wake up during a time frame which can be conservatively modeled by the duration of the contention window W for clear channel assesment plus the duration of a message exchange M. Modeling the cumulative wake up rate as a Poisson process of intensity $R = \sum_{1=1}^{N} r_i$ and the probability of having a good channel quality as a Bernoulli process of probability c, I have:

$$P[s = 1] = ce^{-(R-r_j)(W+M)} = p \tag{3.38}$$

and $P[s = i] = (1 - p)^{i-1} p$ for $i > 1$.

The distribution of delay given that the first attempt is succesfull is a uniform distribution, that is $P[D_j \leq \tau | s = 1] = r_j \tau$ for $0 \leq \tau \leq 1/r_j$.

I need to charachterize $P[D_j \leq \tau | s = i] = P[d_{j,i} \leq \tau]$ for $i > 1$. Notice that

$d_{j,i} = T(\alpha_1 + \alpha_2 + ... + \alpha_i)$

where T is the duration of a back off unit and

$\alpha_k \in U\left(0, 2^k - 1\right) \approx U\left(0, 2^k\right)$ for $k > 1$

and

$\alpha_1 \in U\left(0, 1/(Tr_j)\right)$

I can now model the delay $d_{j,i}$ using the Central Limit Theorem:

$$d_i \in N\left(\frac{1}{2r_j} + \left(\frac{T}{2}\left(2^{i+1} - 3\right)\right), \frac{1}{12r_j^2} + \frac{T^2}{12}\left(\frac{4^{i+1} - 13}{3}\right)\right) \tag{3.39}$$

Altough I charachterized all the different pieces of the summation in Equation 3.36 providing a closed for expression for the distribution, still this expression is extremely difficoult to handle. Consequently, it is useful to consider some possible bounds.

A lower bound to the delay distribution can be found stopping the expansion of Equation 3.36 to the first term. That is:

$L_j(\tau) = pr_j\tau$

A higher bound can be found considering the best case scenario when the first attampt is succesfull. That is

$H_j(\tau) = r_j\tau$

Consequently, $pr_j\tau \leq P[D_j \leq \tau] \leq r_j\tau$. This means that for high values of p the two bounds are very close and can approximate the cumulative distribution function with high precision.

Chapter 4

Static Mapping

Most of the current wireless sensor network deployments are charachterized by:

1. A cyclic monitoring and control routine. For example, periodic vibration or temperature monitoring with data that has to reach a base station within some deadline and a given reliability.

2. A sensor network infrastructure that is specifically deployed to serve a single application.

As a consequence, most of the design choices in terms of topology, communication protocols, selection of parameters and quality of services, can be done off-line (commission time) without the need of major run-time adjustments, except for sensor calibration that is done at deployment time. The end result is a network that can be highly optimized to serve a specific application with good energy performance but, despite a certain level of robustness against adverse channel conditions and nodes malfunction, it is hardly reusable across different applications, let alone be shared among different applications. However, since energy efficiency is a high priority in

this domain and applications are mostly periodic, there is a clear value in this design approach.

Because of the capability of mapping the different services onto the network resources directly at design time, I call this design problem *Static Mapping* to differentiate it from the *Dynamic Mapping* problem, that will be discuss in the next chapter, where the mapping decisions cannot be taken at design time and must be taken at run time by the controllers.

In this chapter, I describe the synthesis approach for the static mapping problem. First, I provide a formal description of the proposed synthesis process using the platform based design theory, then I introduce Rialto, a framework to capture the application specifics and derive constraints for the network design. Finally, I present two case studies, the first one on building monitoring and the second one on industrial automation, to show the methodology in action all the way down to testbed implementations.

4.1 PBD formulation

In this section, I present the formalization of our PBD methodology for static application, using the algebraic framework introduced in Chapter 2. Following the approach of the previous chapter, I use a set of platforms and abstraction layers and formalize the refinement steps as a sequence of mappings onto more refined semantic domains. I use the SNSP, SNIP, and SNAPP and to drive the first step of the design flow, I introduce a new platform, called the Virtual Connectivity Platform.

The synthesis is composed by a sequence of successive refinements that starts with a description of the application already mapped onto the SNSP and delivers at the other end a network of wireless sensor nodes running a communication protocol.

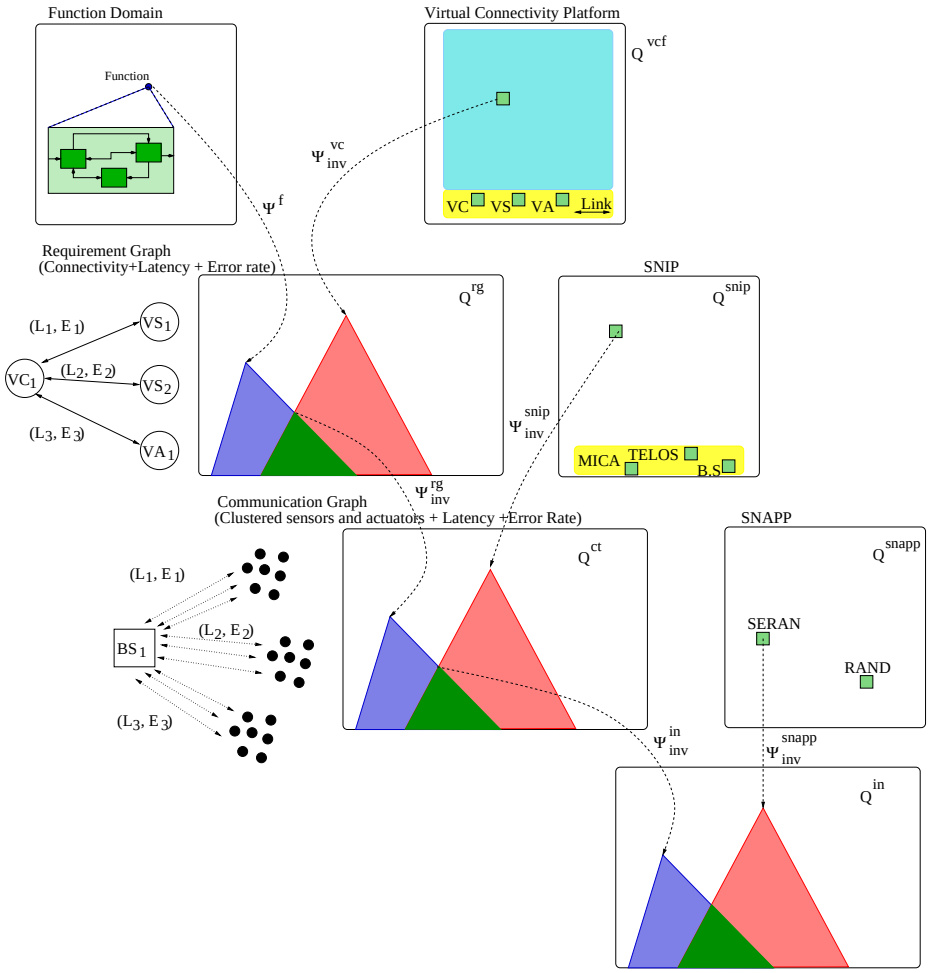

Figure 4.1. Layers of abstraction and design flow

With reference to Figure 4.1, the highest level of abstraction is a functional desciption of the control algorithm. In general, as I mentioned in the previous chapter, the application can be described using a combination of services belonging to the SNSP and conditional statements. However, for the case of cyclic control application, I developed a specific framework, called Rialto [54], to capture the specifications.

In Rialto, the control algorithm is specified as a sequence of queries and commands. Intuitively, a Query is a request for sensed data from a specified area (i.e. a robot or a room in a building). Similarly, a Command is a request for an actuation in a particular area. Resctrictions on how queries and commands can be composed in the

functional description, depend on the model of computation that the end user wants to employ. The next section is specifically focused on this framework.

The first platform that is used is the Virtual Connectivity Platform $\mathcal{Q}^{vcf}$. The library elements are of four types: the set of *Virtual Sensors* $\mathcal{S}_v$, Virtual Controllers $\mathcal{C}_v$, Virtual Actuator $\mathcal{A}_v$ and bidirectional links $\mathcal{L}$. A virtual sensor $\mathbf{s_v} \in \mathcal{S}_v$ is an abstraction of a sensing area, it is characterized by its position and will be later on refined in a cluster of sensor nodes. An example of a virtual sensor could be a room in a building in a building monitoring application. Similarly a virtual actuator $\mathbf{a_v} \in \mathcal{A}_v$ is the abstraction of an actuation capability, it is characterized by its position and will be eventually refined in one or more actuators. A virtual controller $\mathbf{c_v} \in \mathcal{C}_v$ is an abstraction of a computation capability and in general is refined in one or more base stations. Virtual components and links can be composed to form other agents. It is not possible to directly connect virtual components, but links must be used. Furthermore, a virtual sensor can be linked only to a virtual actuator, a virtual actuator can be linked only to a virtual controller and virtual controllers cannot be directly linked. A partial order can be defined for this domain such as $v_1 \preceq v_2$ if and only if for each virtual component in v_2 there is a correspondent virtual components in v_1 and for each link in v_2 there is a corresponding link in v_1.

The common semantic domain for the first refinement step is called *Requirement Graph* and denoted by $\mathcal{Q}^{rg}$. This domain is similar to $\mathcal{Q}^{vcf}$ but links are annotated with a pair (L, E_r) that represent the end-to-end latency and error rate requirements, and virtual sensors are annotated with a pair (S_r, F) that represent the sensing rate requirement and type of aggregate data (i.e. average value, max value, all values) for that area. While the rules of composition are the same as $\mathcal{Q}^{vcf}$, the order is such that $v_1 \preceq v_2$ if for each virtual component in v_2 there is a correspondent virtual components in v_1 with higher or equal S_r and same F, and for each link in v_2 there is a corresponding link in v_1 with smaller or equal latency and error rate. Intuitively,

given two comparable instances of this domain, the highest is the one with looser communication and sensing requirements. I call ψ^f the conservative approximation that maps the functional description onto $\mathcal{Q}^{rg}$.

Formally, given an agent $\mathbf{r} \in \mathcal{Q}^{rg}$, I can define a conservative approximation as follows. The lower bound Ψ_l^{vc} abstracts the quantities S_r,F,L and E_r. The upper bound Ψ_u^{vc} also abstracts all links. Agents $\mathbf{r} \in \mathcal{Q}^{rg}$ that are represented exactly in $\mathcal{Q}^{vcf}$ are agents with no links. I select an instance from the $\mathcal{Q}^{vcf}$ that has the minimum number of virtual components declared by the functionality. Ψ_{inv}^{vc} maps such instance onto $\mathcal{Q}^{rg}$. The cone in figure 4.1 represent the set of agents that have the same number and types of virtual components as in the selected instance but with links connecting them, and all possible combination of latency, error rate and sensing rate. For the sake of simplicity, in the sequel I do not specify the upper and lower bound for each conservative approximation whose construction should be intuitive.

The intersection of the two cones in $\mathcal{Q}^{rg}$ gives all the requirement graphs with connectivity, latency, error rate and sensing rate that are good enough to support the initial functionality. Among these possible refinements I choose the "highest", which is the one with the minimum number of virtual components and looser sensing, latency and error rate requirements. I call this instance $\mathbf{r_g}$ and this is the starting point for the rest of the synthesis flow.

As I explain in the next section, RIalto generates $\mathbf{r_g}$ starting from the functional description.Rialto starts from the sequential description of the control algorithm, it considers all the possible branches in the decision tree of the algorithm and analyzes all the communication and sensing requirements for each of these brances. Starting from these requirements, it generates the minimum requirements that each link and virtual sensor has to satisfy so that the communication and sensing infrastructure is

able to support the application in whatever decision branch it may end during the actual execution.

The next step is to refine $\mathbf{r_g}$ the requirement graph into a clustered topology. The goal is to substitute the virtual components with an adequate set of physical components.

To do this, I use the Sensor Network Implementation Platform $\mathcal{Q}^{snip}$. The elements of the library are a collection of *Physical Nodes* (i.e. Mica, Telos, Intel motes), *Base Stations* (i.e. Stargate), and links. Nodes and base stations are characterized by their hardware abstraction (i.e. component size, memory, power consumption, clock speed), radio interface, sensing capabilities, location, and price. Physical components can be connected only using links. A link represents the capability of communication between two physical components. Restrictions on the possibility of linking directly two components reflect the reachability due to their radio interface. For example MicaZ and Telos motes can be linked since they both are ZigBee compatible, while Mica2 and MicaZ, that are not radio compatible, cannot be directly linked, but a path between the two can exist only if there is also a third component (i.e. a base station or a node with a reconfigurable radio) that is able to support both radio interfaces. I can define a partial order similar to the one of $\mathcal{Q}^{rg}$ where the higher element is the one representing a minimum topology with looser requirements.

The common domain for this step is the *clustered topology* $\mathcal{Q}^{ct}$. An instance of this domain is a graph of interconnected physical components where components are associated to a cluster. Furthermore links are annotated with the usual requirement couple (L, E_r), components are annotated with a sensing requirement S_r and clusters with the aggregate function type F. I can define a partial order in the same way as in $\mathcal{Q}^{rg}$ where the higher element is the one representing a minimum topology with looser requirements.

The instance $\mathbf{r_g}$ is mapped to a set of ordered agents in $\mathcal{Q}^{ct}$. Each agent in this set has at least a base station for each virtual actuator, a sufficient number of clustered sensor nodes for each virtual sensor so that the sensing requirement is satisfied, a sufficient number of actuators for each virtual actuator, and a weighted link between the base station and each component with latency and error rate less than or equal to the correspondent quantities between the virtual actuator and virtual sensor in $\mathbf{r_g}$.

Selecting an agent in $\mathcal{Q}^{snip}$ and mapping it to $\mathcal{Q}^{ct}$, means selecting the hardware platform and a lower bound on the density of nodes for each cluster. The mapping gives a set of ordered agents that may or may not intersect the set of agents obtained by mapping $\mathbf{r_g}$ to $\mathcal{Q}^{ct}$. This intersection represents all the clustered topologies with a sufficient number of interoperable nodes per clusters and connotations that are good enough to satisfy the link and sensing requirements.

I now have to eliminate a set of solutions that are unfeasible for any of the following reasons: size constraints (i.e. it is impossible to place 100 Telos motes in a square foot), external constraints (i.e. for regulatory issues, it is not possible to place the motes in some specific locations), budget constraints (the overall solution has too many nodes and I cannot afford it). Among the remaining solutions, choosing the right one to propagate down in the synthesis process is not trivial. Given a number of nodes per cluster, I choose the solution with the loosest sensing and communication requirements. However it is difficult to understand at this level what is a good number of nodes per cluster. On the one hand, more nodes involve a higher cost of the solution. On the other hand, the higher the density of the network, the more energy efficient the final solution will be because nodes can be duty cycled for energy preservation. However this energy consumption cannot be estimated until the communication protocol is decided and this happens at the next step of the design flow. Consequently, I suggest to start with a solution that is relatively high in the space of the possible solutions (i.e. with a low number of nodes), and after the communication

protocol is mapped, evaluate if the energy consumption per node is satisfactory for a good lifetime of the network. If that is not the case, go back and select another possible solution with more nodes. As I will explain later, once the protocol is mapped on the nodes, the trade-off curve between energy consumption and density is available. Consequently, a good number of nodes can be selected hence facilitating this iteration process.

As already mentioned, the last step is concerned with associating a communication protocol to the physical components such that the communication requirements are satisfied and the energy consumption minimized. To drive this step I use the Sensor Network Ad-Hoc Protocol Platform (SNAPP) $\mathcal{Q}^{snapp}$. The library elements of the SNAPP are MAC and routing protocols. In the wireless sensor network domain space, the layering between MAC and routing is usually not a good solution since it significantly reduces the energy optimization capabilities associated with cross-layer design. Consequently, the SNAPP is populated by non composable instances of integrated MAC and routing solutions. Different protocols have been developed for different application classes. For example SERAN [43] was developed for periodic control applications with more than one cluster, while the randomized approach of [44] (called RAND in Figure 4.1) is optimized for single cluster topologies. These protocols are "parametrized protocols", meaning that their structure is specified, but their working point is determined by a set of parameters. For example, in SERAN the working point of the protocol is determined by a channel access probability p, a TDMA slot duration S, and a TDMA cycle duration Δ.

The common semantic domain in this step is represented by the instantiated network domain $\mathcal{Q}^{in}$. An instantiated network is an operational WSN, i.e. a network of physical nodes with a communication protocol. Mapping the selected clustered topology onto this common domain I obtain all the possible instantiated networks that satisfy the given E2E requirements on latency and error rate, while mapping a

SNAPP instance I obtain all the possible instantiated networks that use the selected protocol with all the feasible combinations of the free parameters (i.e. p, S, Δ for SERAN). The intersection between the two mappings gives all the possible instantiated networks that use the selected protocol and satisfy the given communication constraints. Among these solutions, I select the one that minimizes the energy consumtpion. At this point, I can evaluate if the synthesized solution can comply with the lifetime requirements of the network. If that is the case, I am done, otherwise I need to get back to the clustered topology domain and select an instance with more nodes.

This final refinement is obtained as the solution of a constrained optimization problem, where the constraints are the latency and error rate requirements while the cost function is the energy consumption that is estimated based on an abstraction of the physical properties of the candidate hardware platform. Note that in many cases, thanks to the mathematical model that charachterizes the protocol performance, once the protocol is selected, it is relatively simple to assess the reduction in energy consumption due to an increased number of nodes [43, 44]. Consequently, a single extra iteration of the last step is usually enough to reach a satisfactory solution.

4.2 Rialto

In this section, I describe Rialto: a framework to capture the specifications of a cyclic cotrol routine of a WSN and produce a set of constraints on sensing, latency, and error rate, that the communication and sensing infrastructure has to satisfy to correctly support the application.

Initially, this project was developed to allow the description of WSN application for industrial automation such as vibration monitoring of the robots in a manufac-

turing cell, but the utility of this framework is extended to all those domains in which the application developer is not the same person the design the communication infrastructure. For example, the application software for manufacturing plants is usually written by process or mechanical engineers that are expert in process control technology, but know little of the communication and sensing infrastructure that has to be deployed to support these algorithms. On the other side, the communication infrastructure is designed by communication engineers that know little about process control technology. Moreover, the adoption of wireless technology further complicates the design of these networks. Being able to satisfy high requirements on communication performance over an extremely unreliable communication channel is a difficult task. Consequently, the gap between the control algorithm designers and the network designers will inevitably increase and this phenomenon might delay the adoption of wireless sensor networks technology in manufacturing plants and in general in all those domains where the end users are not network designers.

Rialto is specifically aimed at bridging this gap. The goals are:

1. Allow the end user to describe the control application independently from the particular communication infrastructure or hardware platform.

2. Capture these specifications in a formal way and perform a state space exploration to analyze all the possible scenarios that the application may lead to.

3. As a result of this exploration, produce a set of constraints that the communication links and hardware infrastructure must satisfy to ensure correct functionality of the network.

Consider the example in Fig. 4.2. In this application, vibration and temperature sensors are deployed on each of the robots to report value of the vibration and temperature patterns to the controller. If the controller notices that a particular robot

shows high values of either the vibrations or temperature parameter, it determines that the robot needs maintenance. Consequently, it sends a message to all the actuators to switch the robots off so that a human operator can perform the required maintenance before the robot creates expensive damage to the production line. I use this case study to present Rialto.

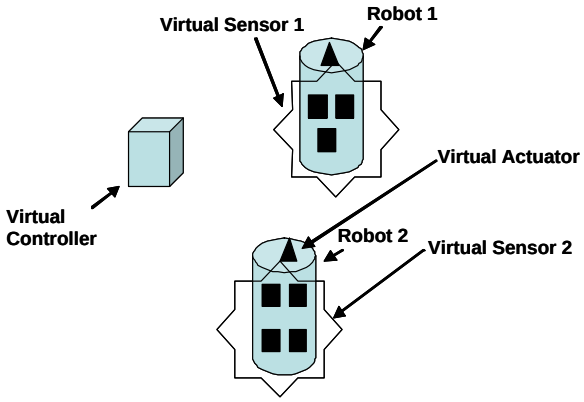

Figure 4.2. Application example

4.2.1 Overview

Following the approach presented in [22], applications should be described in terms of logical components that communicate via queries and commands. Queries are requests for data and, as a consequence, each query is followed by a corresponding response. Commands are used to set some parameters or trigger some actions and do not necessarily need a response.

A *query* is the formalization of the intuitive notion of data request such as "Sense vibrations from time T_i to time T_f with a sampling rate of X [samples/sec] and return to me the average within L seconds with a message error rate of 10^{-3}". Consequently, a query is composed of various fields whose parameter values define its content. One of these fields is the "attribute" that defines the quantity to be sensed (i.e. temperature, humidity, vibration, etc.), another one defines the required sampling rate, another one

the time scope of the query (in the above example T_i and T_f), and so on. A *command* is the formalization of the intuitive notion of triggering an actuation such as "Switch the robot off from time T_i to time T_f; the command has to reach destination within L seconds and with a message error rate of X [samples/sec]". Similarly to the query, the content of the command is specified using multiple fields. This approach offers a very intuitive way of describing the application relieving the control algorithm designer of the burden of dealing with the physical network implementation.

Because of the large variety of applications that could be implemented using a WSN, it is very difficult to propose a single MoC that allows the right level of expressiveness. Furthermore, the capabilities of the sensing and communication infrastructure are not related to the read and write semantics of the application. For example, the requirement that the link between two virtual components should allow for a maximum latency of L seconds or that sensing should be performed at the rate of X [samples/sec], is a consequence of the content of the query and does not depend on the semantics of the application. With this in mind, I think that the right approach is to allow designers to specify the selected read and write semantic, while the communication and sensing infrastructure should be derived independently.

In the proposed framework, designers describe the application in a Rialto Model in terms of Virtual Controllers, Virtual Sensors, and Virtual Actuators. (see Fig. 4.2).

A *Virtual Controller (VC)* contains the description of the control algorithm of the application. If the application has more than one independent control algorithm, multiple Virtual Controllers have to be specified. In our case study, I have a single VC with an algorithm that needs information on both temperature and vibrations to take its decisions. The VC is only an abstraction of the control capabilities required by the application. This abstraction does not restrict our design space to a centralized control solution. In fact, in the physical implementation, the control algorithm

described in a single VC could be implemented in a distributed fashion whenever it is convenient. Similarly, the functionalities of different Virtual Controllers could be implemented in the same physical component. Usually, designers already have a good idea of where the physical controller, or controllers, can be placed. Consequently, they can embed this location information into the VC and limit the design space space exploration. In our case, the virtual controller is given a convenient position close to the robots.

A *Virtual Sensor (VS)* represents a sensing area. This abstraction is useful because designers know which are the areas that need to be sensed, but they generally don't know how many sensors must be placed to cover that area and how they have to placed. For example, in our application, there are two virtual sensors (one for each robot). Similarly to the VC, there is not necessarily a one-to-one relationship between virtual sensors and physical sensors. The number and the type of physical sensors that will be used to implement a virtual sensor is an implementation choice. In our application, a virtual sensor will most likely be implemented with a set of multiple sensors.

A *Virtual Actuator (VA)* represents an actuation capability. Similarly to the VS, the user describes the position of the VA, but the number and type of physical actuators that will be selected to implement its functionality is an implementation choice. In our case, there are two Virtual Actuators, one for each robot.

After the virtual components are declared, the interaction among them is described using queries and commands. Rialto allows connections only between Virtual Controllers and Virtual Sensors and between Virtual Controllers and Virtual Actuators. Consequently, no connection is allowed between two Virtual Sensors, two Virtual Actuators, or a Virtual Sensor and a Virtual Actuator. This restriction makes sense because I am describing an application using logical components. Connections be-

tween two sensors (commonly refered to as multi-hopping) are an implementation option, and as such they don't belong to the application description level of abstraction. Similarly, a connection between two physical controllers is an implementation option, but at the application description level connections between two Virtual Controllers are not allowed. Hence, if a Virtual Controller needs a particular set of data, it has to send a query directly to a Virtual Sensor.

After the application is described, the description is translated into an internal representation called RialtoNet.

Since I want to generate a set of requirements to design a sensing and communication infrastructure that is able to satisfy every possible request of the controlling algorithms, I need to evaluate all the various combinations of requests that Virtual Controllers could generate. The RialtoNet is created precisely for an explicit exploration of all the possible combinations of queries and commands in a given application. Since the number in a control routine has is typically limited, the number of possible combinations is often very manageable.

By analyzing the software code of every VC, I detect all the possible combinations of conditional statements involving a request, and for each of them I create a new independent component, called VC Branch (VCB). Each Virtual Sensor is modified into a Virtual Sensor Skeleton and each Virtual Actuator into a Virtual Actuator Skeleton (VAS) that are obtained from the original code modifying the read and write semantic. A RialtoNet is generated by substituting each VC with its relative VCBranches, each VS with its relative VSS, and each VA with its relative VAS.

The execution of the RialtoNet is based on a model of computation that takes inspiration from Kahn Process Networks (KPN) [55, 56]. The model of computation is deterministic and ensures a deadlock free execution.

The RialtoNet does not follow the read and write semantic specified in the original

Rialto Model. It is only an internal representation that is generated to efficiently organize the analysis of the quantities that are of interest to set the requirements on the communication links and sensing capability of the physical network. Consequently, the RialtoNet is not used to check the functionality of the application specified in the Rialto Model. This is a trade-off that I pay to leave complete freedom to the application designer to speficy the control algorithm with the semantic and yet be able to derive information to build an appropriate network architecture.

During the execution of the RialtoNet, I generate a set of constraints on latency, bit error rate, and sensing requirements that are the starting point for the design of the physical network. Since the distinct VCBranches are executed as independent components and each of them represents a possible combination of queries and commands, the requirements on sensing and communication infrastructure guarantee that all the possible combinations can be supported.

Consequently, the end user is able to describe the application with no knowledge of the network architecture, while Rialto provides a bridge to the implementation platform. The constraints generated by Rialto refer to quantities that are of interest for the design of the network architecture. Starting from these requirements, the network topology can be generated and a communication protocol selected with the guarantee that, if these constraints are satisfied, the network architecture will be appropriate to support the correct functionality of the application.

4.2.2 Rialto Model

In this section, I explain how the application is described in the Rialto Model. The basic building blocks are actors and communication media. Actors exchange use communication media to exchange tokens.

Tokens

Tokens are the abstraction of queries and commands.

A query specifies the attribute to be sensed (i.e. vibration, temperature), the function to return (e.g., all the values, average value), the sampling rate, and the time scope. The time scope defines the interval of time for which the sensing should be performed. A command specifies the type of actuation (e.g., switch off the robot, increase temperature), the intensity of the actuation (i.e. temperature should be increased of 5 degrees Celsius), the need for an acknowledgment, and the time scope of the actuation. Both queries and commands also specify a tolerated latency and message error rate for the communication.

A token has nine fields, and its structure is:

$$Token = (q, c, n, a, v, Ti, Tf, L, Q),$$

where:

- $q \in \{0, 1\}$ specifies if it is a query or a command.
- $c \in \{0, 1\}$ specifies if it is a request or a response.
- n is the function to return for a query or the need for an acknowledgment for a command.
- a is the attribute of the query or the type of actuation.
- v is the required sampling rate (for a query) or the intensity of the actuation (for a command).
- T_i, T_f, are respectively the beginning and the end of the scope of the query (i.e. "Give me humidity data from time T_i to time T_f").

- L [sec] is the latency requirement.
- Q is the quality of service requirement (bit error rate).

Actors

There are three types of actors: Virtual Controller (VC), Virtual Sensor (VS), and Virtual Actuator (VA).

The internal structure of a VC is a cyclic control routine, that is a while loop that is periodically executed. Fig. 4.3 contains the pseudo-code of the VC routine for the monitoring application of Fig. 4.2. The code within the "while(true)" loop is called the *control cycle*. The number of queries and commands that can be generated during a control cycle must be limited. Consequently, no while loop with a query or command inside is allowed within a control cycle. In most of the WSN applications that we analyzed, the control cycle can be expressed with less than about 100 queries or commands. Furthermore, the user can specify the time scope of the control cycle (how much time between two consecutive executions). If such parameter is not specified, I assume that the time scope is given by the the lowest T_i and the highest T_f of the generated queries or commands.

```
/Virtual Controller

VC
Connections:
VS1,VS2
VA1,VA2

While(true){
        Q1=new query(1,0,avg,vib,100,init,t1,5s,e-3);
        //Query for vibration
        Q2=new query(1,0,avg,temp,10,t1,t2,10s,e-3);
        //Query for temperature
    Q3=new query(1,0,avg,vib,1000,t2,t2+10,1s,e-3);
    //Query for vibration with higher sampling rate
    Q4=new query(1,0,vib,temp,10,t2,t2+15,5s,e-3);
    //Query for vibration with lower samplin rate
        //to be performed only in non critical situation !!
    C1 = new command(0,0,x,turn_off,t2,t2,5s,e-3);
        //Turn off the robot

        send(Q1,VS"i");
        send(Q2,VS"i");
```

```
await(VS1){
        response1=receive(VS1);
        data1=response1->data;
        attribute1=response1->attribute;
        if (((attribute==vib)&&(data1>threshold_vib))||
         ((attribute==temp)&&(data1>threshold_num))){
           send(C1,VA1);
                send(C1,VA2);
                send(Q3,VS1);
        }
        else{
                send(Q4,VS1);
        }
}

await(VS2){
        response2=receive(VS2);
        data2=response2->data;
        attribute2=response2->attribute;
        if (((attribute2==vib)&&(data2>threshold_vib))||
         ((attribute2==temp)&&(data2>threshold_num))){
           send(C1,VA1);
                send(C1,VA2);
                send(Q3,VS2);
        }
        else {
                send(Q4,VS2);
        }
}
```

Figure 4.3. VC for the case study

```
//Virtual Sensor

VS
Connections: VC;

While (true){

        Evaluate Inputs{
                if (input ==null)
                        Task();
                else {
                        Read Input();
                        Update parameters();
                        Task();
                }
        }
```

Figure 4.4. VS and VA for the case study

VA and VS are sequential threads of computation. An example is shown in Fig. 4.4. They read the queries/commands at their inputs, perform their sensing/actuating task to satisfy those requests, and return data (if necessary) to the controller that sent the query/command. They are composed of two main functions: "Evaluate Inputs" and "Task". The "Evaluate Inputs" function specifies how the read semantics of the actor, while the "Task" function specifies how the actor fulfills the required sensing/actuation task.

As already mentioned, the user is free to specify any type of read and write semantics in the actors. For example, in our case study, the VC performs a blocking read at each of its inputs ("await" statement) before executing a corresponding atomic critical section. Conversely, the Virtual Sensors and Virtual Actuators of our example, perform a sensing task in a non-blocking read fashion. A pseudo-code for the Virtual Actuators and Virtual Sensors of our example is given in Fig. 4.4 (since in this case the VA and the VS have the same non blocking semantics, I report only the code for the VS).

I believe that this expressivness is very important to cover a large set of possible applications.

Communication Media

Actors communicate through bidirectional, lossless, unbounded FIFO channels. Each channel is characterized by two separated queues, one for each direction. Connections are allowed only between a VC and a VA, and between a VC and a VS.

4.2.3 RialtoNet

After the Rialto Model is specified, I translate it into an internal representation called RialtoNet to explore all the possible combinations of queries and commands that the communication and sensing infrastructure must satisfy. In particular, the generation of the RialtoNet goes through the following steps:

1. A set of actors called VCBranches is generated from each VC.
2. An actor called Virtual Sense Skeleton (VSS) is generated from each VS.
3. An actor called Virtual Actuator Skeleton (VSS) is generated from each VA.
4. An extra actor, called Sink, is generated.
5. These actors are connected together to form a RialtoNet.

Once generated, the RialtoNet is executed as follows:

1. Each VCB sends a sequence of queries and commands.
2. Each VSS and VAS elaborates the received queries/commands and sends to the Sink messages indicating the minimal requirements on sensing and communication to satisfy those queries/commands.
3. The Sink stores this messages and elaborates the requirements for the output format.

4.2.4 Generation of the RialtoNet

The goal of building a RialtoNet is to derive all possible combinations of queries and commands that the VC code of the original Rialto Model may produce during the system operations.

Generating VCBranches

I consider the case of analyzing the conditional branches in a single control cycle. For each conditional branch that involves the possibility of sending a request, I consider both scenarios: the one in which the branch is taken, and the one in which the branch is not taken. This analysis generates all the possible combinations of queries and commands within a single cycle. Each of these combinations generates a VCB. A VCB is composed by a sequence of "SEND" instructions that represent a possible combination. Consequently, the VCB is an actor that is only able to send a predetermined sequence of tokens ("source" actor). Since in the control cycle of the original VC code there is only a limited amount of queries and commands, also the number of "SEND" instructions in a VCB is limited.

Since in the example of Fig. 4.3 there are two "if" statements, four VCBranches are generated (see Fig. 4.5).

Notice that a VCB does not contain the informations on read and write semantics specified in the original VC code. This is in line with our approach of considering only the requirements on the sensing and communication infrastructure that the WSN must support.

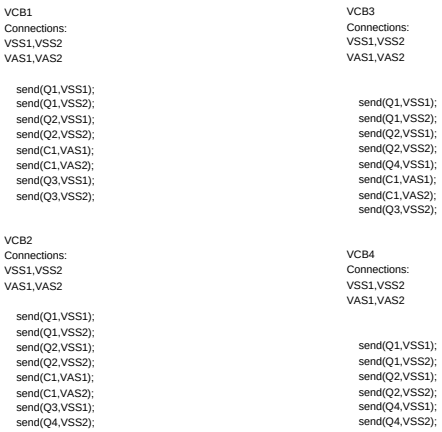

Figure 4.5. Virtual Controller Branches after Branch Separation

Generating VSS and VAS

The VSS and VAS are sequential threads of computation. Similarly to the VCB, the VSS and VAS do not inherit from their originating actor the information regarding read and write semantics. They are composed by a "Task" function that is fired whenever their firing rules are satisfied. The "Task" code is inherited from their generating VS or VA. The firing rules are explained in the next section.

VSS and VAS have an internal variable called *Progression Tag*. As I will see later in this section, this variable indicates the end of the time scope of the last query or command that has been served.

Sink

Every time a RialtoNet is generated, an extra actor called "Sink" is created. The Sink has only input channels and, as discussed in the next section, it is used to store the results of a RialtoNet execution.

Connections

Actors in a RialtoNet communicate also through unbounded, unidirectional, lossless, FIFO channels. Each VCB inherits the connections of its generating VC in the Rialto Model. The direction of these connections is from the VCB to the VSS or VAS. Each VSS and each VAS has an output connection to the Sink. The RialtoNet for the case study is shown in Fig. 4.6.

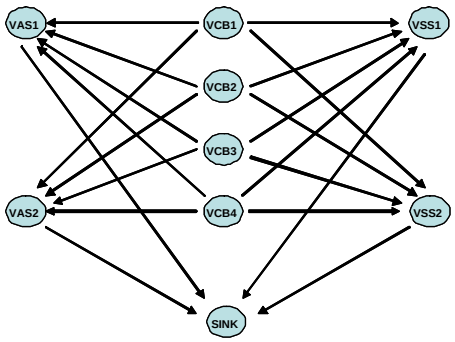

Figure 4.6. RialtoNet for the case study

4.2.5 Execution of the RialtoNet

Before describing the read and write semantics of the RialtoNet, I need to introduce the END Token, a particular token that is automatically produced in the following two cases:

1. From a VCB to all its output channels upon termination of its sequence of "SEND" instructions

2. From a VSS or VAS to the Sink whenever the its execution is terminated.

Its structure can be interpreted as:

$$END = (q, 0, 0, 0, 0, null, \infty, null, null)$$

Read and Write Semantics

The VCB follows a non-blocking write semantics. Since it is a source actor, no reading semantics needs to be specified. The Sink has a non-blocking read semantics. The VSS and VAS have blocking read and non-blocking write semantics. Since the blocking read rules for VSS and VAS are the same, I explain them only for the case of the VSS.

1. The VSS stalls its execution until all its input queues have at least one token.

2. Once that all the input queues are non empty, the VSS evaluates the first token of each of the input queues.

3. If a VSS has END tokens in all its input queues, it sends an END token to the Sink and stops executing.

4. Otherwise, the VSS selects the token with lowest T_f. If more than one token happens to have the same T_f and it is the lowest, all of these tokens are selected. Consequently, an END token is never consumed because it has ∞ in its T_f field.

5. The VSS fires its sensing task. The output of the sensing task depends on the "a" and "v" fields of all the input tokens whose T_i field is less than or equal to the T_f field of the selected token.

   For example, in our case study VSS has four input queues and the first tokens at those queues are:

   Input1:END

   Input2:END

   Input3:Q3=(1,0,avg,vibration,1000,t2,t2+10s,1s,e-3)

   Input4:Q4=(1,0,avg,temperature,10,t2,t2+15s,5s,e-3)

The VSS selects Q3 because it has lowest T_f and it advances its task until $t2 + 10s$. Requirements are generated for the interval $(t2, t2 + 10s)$ such as:

- Sensing: vibrations at a rate of 1000 samples/sec and temperature at a rate of 10 samples/sec.
- Communication: latency of 1s (the most restrictive among the two) and message error rate of 10^{-3}.

This is the set of requirements that the VS must satisfy in order to serve all the possible queries within that time scope.

The last action of a firing is the generation of a "Requirement Token". This token has the same nine fields of the other tokens, but instead of abstracting a query or a command, it embeds information of the generated requirements. In our example, the VSS generates a Requirement Token:

Out = (1,0,[avg,avg],[vib,temp],[1000,10],t2,t2+10s,1s,e-3).

This token encodes the following requirements: "From time $t2$ to time $t2 + 10$ seconds, the VS must be able to sense vibration and temperature at a rate of respectively 1000 sam/sec and 10 sam/sec., and return the average. Furthermore, it must be able to communicate with the Virtual Controller with a latency of 1 sec and a message error rate of 10^{-3}".

The VSS sends the Requirement Token to the Sink. Since the Sink receives only Requirement tokens (and END tokens which have a fixed structure), and it is the only one receiving them, there is no need to distinguish this token from the other types.

6. The selected token (in our case Q3) is consumed, meaning that it is removed from its input queue and destroyed.

Notice that this blocking read mechanism is different from the original KPN because the VS can evaluate input tokens that it does not consume.

The Progression Tag update happens at every firing of the sensing/actuating task. At the end of the firing, the progression tag of the VS/VA is set to the value of the Tf field of the selected token.

Queries sent in the same VCB connection must have non overlapping time scopes. This is to avoid the situation in which, after advancing to serve a query, a VS would have to backtrack to serve another query with different requirements. Once the code for a VCB is generated, this condition can be easily checked. If such a case is detected, the VCB code is modified and every couple of overlapping queries is replaced by three non overlapping queries. Queries emitted from the same VC branch must have non decreasing T_f field. This is to avoid the phenomenon of "sending a query to the past".

The execution terminates when each VSS and each VAS has sent an END token to the Sink.

4.2.6 Properties of the MoC

Determinism

The RialtoNet semantics is based on a deterministic MoC: there is only one possible behavior for the input and output sequences of the actors.

I define T the set of all the finite and infinite sequence of tokens, including the empty sequence ($\perp$).

Consider the prefix order ($\leq$) such that $s_1, s_2 \in T$, $s_1 \leq s_2$ if for all the $n \in N$ for which $s_1(n)$ is defined, $s_1(n) = s_2(n)$. For instance, if $s_1 = (a, b, c, d)$ and $s_2 = (a, b, c, d, e, f)$, then $s_1 \leq s_2$. A simple extension of the prefix order is the pointwise

prefix order ($\sqsubseteq$). Assume $(a_1, a_2), (b_1, b_2) \in (T \times T)$, the pointwise prefix order is defined as: $(a_1, a_2) \sqsubseteq (b_1, b_2)$ if $a_1 \leq b_1$ and $a_2 \leq b_2$.

The set T^N with the pointwise prefix order $(T^N, \sqsubseteq)$ is a complete partial order (CPO) [57]. A function F is monotonic with respect to $(T^N, \sqsubseteq)$ if for $a, a' \in T^N$ and $a \sqsubseteq \acute{a}$, it follows $F(a) \sqsubseteq F(\acute{a})$.

Similarly to the processes in KPN, the VAS and VSS are monotonic [55, 56]. This is a property inherited from the blocking read mechanism and by the fact that the choice of the token to be consumed is deterministic and based on the T_f field and not on the order of arrival of the candidate tokens.

Furthermore, since the input sequences are bounded by the total number of queries/commands declared in the VCB, the VAS and VSS are also continuous with respect to $(T^N, \sqsubseteq)$.

The VCB and the Sink are trivially continuous functions since they are source and sink function.

Since a RialtoNet is composed of continuous functions under a CPO, it converges to a least fixed point [55]. Consequently, the least fixed point is the only behavior and the model is deterministic [57].

Deadlock free

Another important property of the RialtoNet execution is that it does not deadlock. Deadlock may happen only if a VSS or VAS waits in vain for a token that will never arrive. The introduction of the END token is tailored to avoid this problem. The idea of introducing the END query to resolve unwanted deadlocks can be seen as a particular case of the "null" message introduced by Misra in [58] when dealing with asynchronous parallel simulations.

does not have any physical implementation, but it is a useful notation to avoid deadlock when capturing specifications.

Conservative advancement

The proposed blocking read mechanism forces the VSS and VAS to have a *conservative advancement* behavior. This means that before firing their sensing/actuating task, they need to wait for all their input queues to have a token, and when they advance they do it only up to the lowest T_f. The reason for this conservative behavior is that I want to be able capture all possible scenarios that the application may involve without having to backtrack the state of the actors.

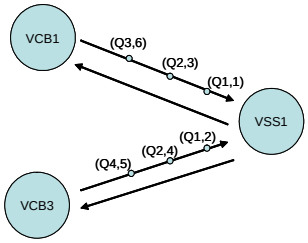

Figure 4.7. Example of conservative advancement

Consider what could happen in the interaction between VCB1,VCB3 and VS1 in our case study. Fig. 4.7 illustrate a scenario where these actors exchange two particular sequences of queries (the definition of the queries is given in Fig. 4.3). Assume that the VS1 does not perform the proposed blocking read. In this case VS1 would serve Q4 from VCB3 and advance its task to the Progression Tag value $t2 + 15s$ before evaluating Q3 from VCB1. As a result, since $Q3$ has higher sampling rate requirement than $Q4$, to correctly serve $Q3$, VS1 would have to go back to $t2$, void its latest execution and serve $Q3$.

This simple example shows that our blocking read mechanism is a clean way to

capture all the scenarios without the need of backtracking. The correctness of the specification capturing is a consequence of the determinism of the MoC.

It is important to understand that the END query and the blocking read mechanism of the RialtoNet are used to ensure the correctness of the execution of the internal format. They are not related to the application described in the Rialto Model and they are not related the final implementation. They only allow to correctly set the requirements on the links and on the number and type of sensors that have to be used. Consequently, RialtoNet allows to generate an architecture that is able to support the functionality, but is not concerned on how the functionality is implemented into the architecture.

4.2.7 Requirement Generation

After the program is terminated, the evolution of the sensing modalities, latency and bit error rate requirements over the time scopes of all the Virtual Sensors and Virtual Actuator are reproduced. These traces represent the requirements that the communication and sensing infrastructure have to satisfy to ensure correct functionality of the application. In this section I analyze the requirements generated by the case study.

```
Control Cycle Time Scope= t2+15sec;

Sensing:

VS1:
 temp:
  (t1,t2)-> 10 sam/sec;
 vib:
  (init,t1)-> 5 sam/sec;
  (t2,t2+10)-> 10 sam/sec;
  (t2+10,t2+15)-> 1 sam/sec;

VS2:
 temp:
  (t1,t2)-> 10 sam/sec;
 vib:
  (init,t1)-> 5 sam/sec;
  (t2,t2+10)-> 10 sam/sec;
  (t2+10,t2+15)-> 1 sam/sec;
```

```
Communication:

 VS1-VC:
  (init,t1)->(5sec,e-3);
  (t1,t2)->(10sec,e-3);
  (t2,t2+10)->(1sec,e-3);
  (t2+10,t2+15)->(5sec,e-3);


 VS2-VC:
  (init,t1)->(5sec,e-3);
  (t1,t2)->(10sec,e-3);
  (t2,t2+10)->(1sec,e-3);
  (t2+10,t2+15)->(5sec,e-3);

 VC-VA1:
  (t2,cycle)->(5sec.,e-3);

 VC-VA2:
  (t2,cycle)->(5sec.,e-3);
```

Figure 4.8. Requirements for the application

The execution of the model produces the traces of Fig. 4.8. Hence, to satisfy the

application requirements, I need to place enough sensors within a VS area to be able to sample vibrations at 1000 samples per second and temperature at 10 samples per second. Furthermore, the communication infrastructure needs to be able to report data to the controller with a latency requirement of 5 seconds with a packet error rate of 10^{-3}. These requirements are the basis to design the network architecture for this application.

4.3 Case Study : Building Monitoring

In this section, I overview a case study that illustrates how our methodology is applied in a building monitorin application. This class of applications is at the moment the greatest market driver for WSN because of its direct impact on improved service and energy savings as well as its clear economic advantages with respect to wired solutions [59, 1, 2].

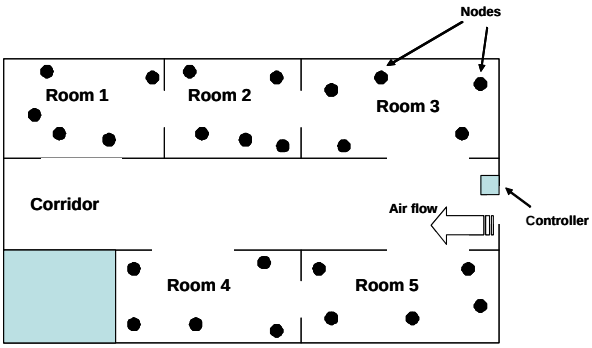

Figure 4.9. Scenario for building automation case study.

In Figure 4.9, I show the schematics of the floor of a building with five rooms and a corridor. I select a typical building monitoring application, where temperature sensors are deployed in all the rooms to perform periodical sensing and forward the data to a central control unit that decides the actuation on the air conditioning

system. Specifically, I want to sense each room once every 15 seconds, and I want the data to reach the Controller within 3 minutes with 95% probability.

Since this is a typical example of clustered environment, I decide to use SERAN, described in the previous chapter, as the underlying communication protocol and I synthesize its parameters to meet the application requirements and optimize for energy consumption.

4.3.1 Capturing specifications and topology selection

In Figure 4.10, I present the application desciption using the SNSP as described in the previous chapter. Again I invoked only the Query and Command service, and I used a conditional block to decide on the proper actuation on the air conditioning system

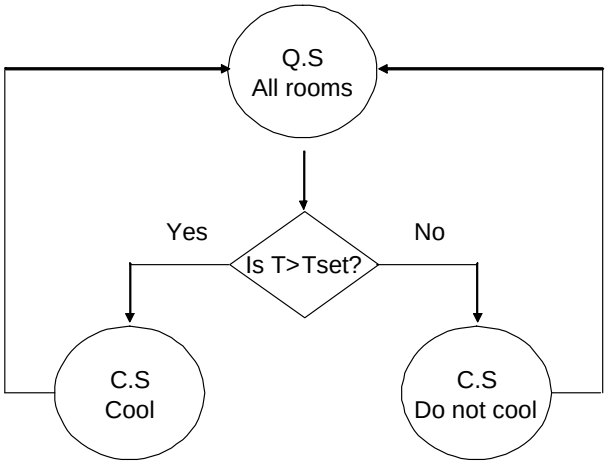

Figure 4.10. Flow chart for building automation case study.

In the Rialto Model, I have one virtual controller placed at the end of the corridor and five virtual sensors. In Figure 4.11, I present the pseudocode for the virtual controller (I reported the response only for one sensor as it is the same for the others). In Figure 4.12, I present the requirements for the communication infrastructure. I do

not consider the requirement on the link to the virtual actuator since I am assuming that the controller communicates with the air conditioning switch directly with a cable.

```
/Virtual Controller

VC
Connections:
VS1,VS2,VS3,VS4,VS5
VA

While(true){
        //Query for temperature
        Q1=new query(1,0,all,temp,1/15,t1,t2,180s,0.05);
        //Control air conditioning
         C1 = new command(0,0,x,turn_off,t2,t2,5s,e-3);
         C2 = new command(0,0,x,turn_on,t2,t2,5s,e-3);

        send(Q1,VS1);
        send(Q1,VS2);
        send(Q1,VS3);
        send(Q1,VS4);
        send(Q1,VS5);

                await(VS1){
                        response1=receive(VS1);
                        data1=response1->data;
                        if ((data1>threshold){
                           send(C2,VA);
                        }
                        else{
                                send(C1,VS1);
                        }
                }
}
```

Figure 4.11. Rialto Model for building automation case study.

```
Control Cycle Time Scope= 15sec;

Sensing:

VS1:
 temp:
   1/15 sam/sec;

VS2:
 temp:
   1/15 sam/sec;

VS3:
 temp:
   1/15 sam/sec;

VS4:
 temp:
   1/15 sam/sec;

VS5:
 temp:
   1/15 sam/sec;

Communication:

 VS1-VC:
  180sec,0.05;

 VS2-VC:
  180sec,0.05;

 VS3-VC:
  180sec,0.05;

 VS4-VC:
  180sec,0.05;

 VS5-VC:
  180sec,0.05;
```

Figure 4.12. Requirements generation for building automation case study.

Once I generated the reuirement graph, I need to select an adequate numner and type of hardware platforms. I decide to place 5 Mica2dots for each room and to use a Stargate base station at the controller. The choice of the number of sensors is, as I will show in the results, an educated guess that will lead us to an energy efficient solution.

4.3.2 Protocol parameter synthesis

As previously mentioned, I want to use SERAN as the underlying communication protocol for this application. I refer to the previous chapter for the description of

the protocol as well as the derivation of the mathematical model that describes the latency and energy performance.

From a connectivity perspective, the topology looks like the one depicted in Figure 4.13.

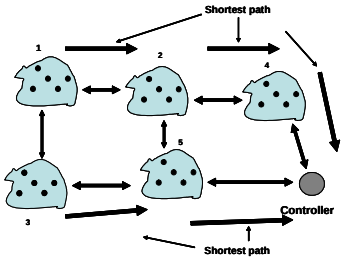

Figure 4.13. Connectivity graph for building automation case study

I considered a latency constraint on the end-to-end delay of $D_{max} = 180s$. I also implemented other power saving techniques.Specifically, nodes wake up for listening with probability 2/5. Furthermore, if a node has in its buffer packets generated by the same cluster, it calculates the average of the data and forwards a single packet. I abstracted the physical layer using a CSMA-slot duration $t = 0.1s$ which I assume to be enough for two nodes to exchange a packet and an acknowledgement. Later in the section I discuss the validity of such an assumption when discussing the testbed implementation.

I consider a packet generation rate per cluster $\lambda = 1pckt/15s$. I take the fixed choice for the channel access probability and I do not consider collision avoidance (notice that this means to turn off the collision avoidance mechanism present in the common distribution of TinyOS when implementing the solution on the testbed). In Table 4.1, I summarize the synthesized parameters.

D_{max}	λ	S_{max-tr}	S_{max-d}	p
$180s$	$1pckt/15s$	$54s$	$16s$	0.07

Table 4.1. Synthesized parameters for building automation case study

4.3.3 Mapping and Implementation

After creating the network infrastructure, the final step of the design flow consists in mapping the controlling algorithm onto the controller hardware platform, and mapping the communication protocol onto the wireless nodes.

The first step consists in mapping the controlling algorithm into the hardware platform of the controller. This represents a classical embedded systems mapping problem (i.e. not specific of the WSN domain) and it can be performed with classical mapping tools. For example, Metropolis [60, 61] is a design environment that was developed to support Platform Based Design. The advantage of using Metropolis in our design flow is that it supports any type of model of computation for the functional description. This property allows us to implementing our philosophy of leaving the user freedom to select the preferred model of computation for describing the control routines. Following the PBD, after the control routine is specified, an abstraction of the hardware platform of the PLC must be provided in order to drive the mapping process.

In many application practices, the controller comes already with a software development environment. Although these environments do not offer the same flexibility and formal methodology of Metropolis, they are commonly used by plant designers because they are user friendly and sold by companies with good customer support. However, I believe that a more formal approach to this problem should be pursued since a bad mapping often leads to suboptimal or faulty implementations.

The second step is to map the communication protocol on the physical nodes.

Since the communication protocols of the SNAPP are already described in a distributed fashion, the parametrized code for each node can be easily developed using the software interface of the nodes. Most often, this interface is given by TinyOS and the parametrized code can be written using NesC [29].

4.3.4 Results

I developed a three steps validation process for our work, which consists on the implementation of SERAN at three different level of abstraction. At the highest level I implemented SERAN in Omnet++ [31], a discrete event network simulator, at the mid level I implemented it in TOSSIM, the simulator for TinyOS based applications, and then I implemented it on a testbed.

The Omnet++ implementation allows for an easy understanding of the trade offs involved with the protocol as well as useful results on the performance, and this is the level in which most of the validation takes place. However, all the physical layer details have to be abstracted, conseqeuntly those results are valid only as long as these abstractions stand. To verify when these abstractions break down, I need to operate at a lower level. The TOSSIM environment does not give significantly more information from a simulation perspective, but it is essential to debug the NesC code for real life experiments on a testbed. The testbed I used is carachterized by 45 MICA2dot nodes placed on a 25 squared meters uniform grid, and connected with an ethernet cable to a PC that is used for both data logging and node reprogramming. Furthermore, the ethernet connection is used to power the nodes and prevent their battery to turn down unexpectedly during an experiment.

The goals of our validation are the following:

1. Determine the validity of our mathematical analysis. In particular, prove that

the analysis of Section 3.4.3 drives to a solution that satisfies latency constraints and minimizes power consumption.

2. Analyze power performance. Starting from the statistics on node duty cycle I want to infer data on the expected network lifetime.

3. Test robustness against clock drift.

During initialization, all the nodes were operational after the first TDMA-cycle. The introduction of a permanent fade between cluster 2 and cluster 5 forced the minimum spanning tree to the shortest path tree of Figure 4.16. The reinitialization of the network was successful after two TDMA-cycles.The most stringent constraint was due to the delay requirement.

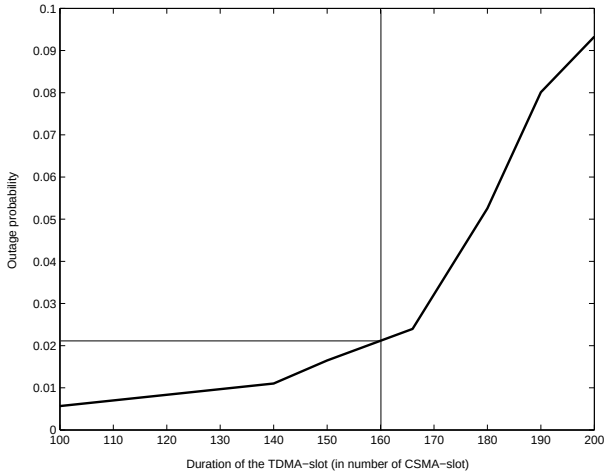

Figure 4.14. Outage probability vs. TDMA-slot duration for Scenario 2

As it is shown in Figure 4.14, the calculated optimal solution offered an outage probability around 2%. At this working point, I observed an average node duty-cycle around 1.4% which ensures a lifecycle of several months for the Mica2dot platforms. As it can be seen in Figure 4.15, better power perfomance can be obtained having longer TDMA-slot, but this would have the side effect of entering a region of ex-

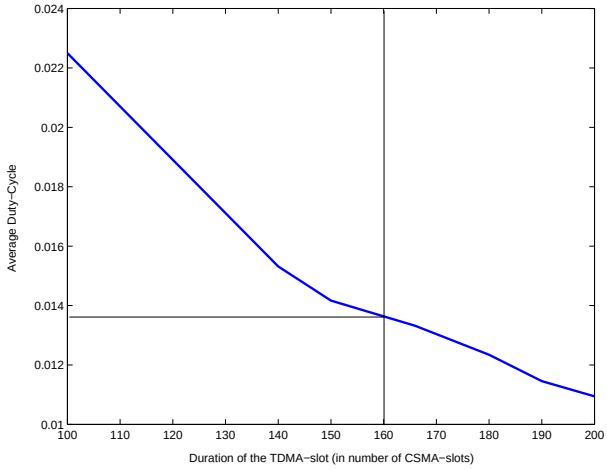

Figure 4.15. Average Duty-Cycle vs. TDMA-slot duration for Scenario 2

ponential growth of the outage probability. For this reason, I consider the optimal calculated solution as the one that offers the best trade-off.

Starting from the optimal calculated solution, I tested SERAN against clock drift. I simulated random clock drift rates of 10^{-2},10^{-3},10^{-4}. As expected, duty-cycle performance did not change. Outage probability raised to 11% in the case of a drift rate of 10^{-2} while it remained below 3% in the other cases. Considering that a maximum clock drift of 10^{-4} was observed for the Mica2 platform [62], SERAN shows high robustness against clock drifts.

I implemented the case study on our testbed and I noticed that the network showed the same behavior as the simulations. The reason for this perfect match is that a CSMA-slot duration of $100ms$ is clearly more than enough for a complete packet-acknowledgement exchange and no unexpected problems appeared.

It is interesting to evaluate how much I can shrink the duration of a CSMA-slot. Since all the other parameters can scale accordingly, the capability of reducing the CSMA-slot is essential to be able to utilize SERAN in a more real time scenario. Unfortunatly, I noticed that using a CSMA-slot duration of $50ms$ I observe a high level of unacknowledged packets that results in a network instability. In other words

there is not enough time to complete an exchange safely. Notice that according to the radio speed, $50ms$ should be enough to guarantee a good behavior (even accounting for a random back off for the acknowledgement transmission). Consequently, the limit in the available channel bandwith is not due only to the radio speed, but also to the capability of processing the information and retransmission with TinyOS. I believe that this is a major limitations for the implementation of real time applications over the existing platforms, at least for multi-hop topologies, and this is not a problem due to our methodology or SERAN, but it is intrinsic with state of the art platforms. Future research will have to focus on the development of more stable solutions for real time behavior at both the hardware and software level. The introduction of the Telos motes has already improved the hardware performance significantly.

4.4 Case Study: Industial Automation

The application of wireless sensor network (WSN) technology [59] to the design of field-area networks for industrial communication and control systems has the potential to provide major benefits in terms of flexible installation and maintenance of field devices, support for monitoring the operations of mobile robots, and reduction in costs and problems due to wire cabling [5, 6, 4].

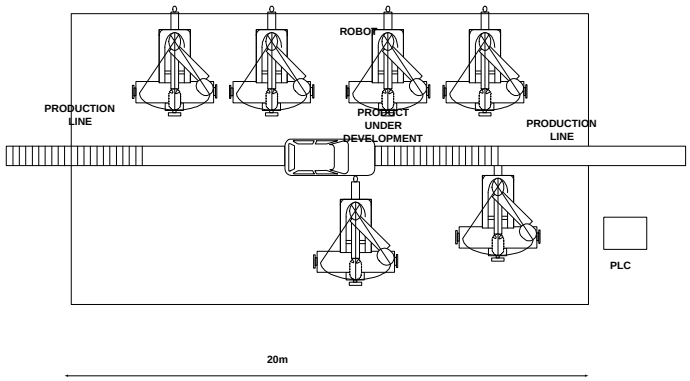

Figure 4.16. Manufacturing Cell

Figure 4.16 illustrates an example of *manufacturing cell*, i.e. a stage of an automation line in an industrial plant. The physical dimensions of this cell range between 10 and 20 meters on each side. In this area, a group of robots cooperate to manipulate and transform the same production piece under the supervision of a process loop controller (PLC), which is placed right outside the cell. The production piece is usually placed on top of a cart that is moved from cell to cell on a rail.

I consider a simplified positioning and identification application where five virtual sensors are deployed in different parts of the rail to perform periodic proximity sensing on the production piece so that the PLC can triangulate the cart position inside the cell. When the PLC estimates that the cart reached the right positioning for the piece to be worked on, it sends a message to a wireless sensor that is placed on the cart (or directly on the production piece). This sensor replies with a message with an identificator of the production piece as well as a code that identifies the types of operations that have to be performed on it. If these informations match the ones that the PLC expected, than the robots can start operating on the piece. Assuming that the final user requires a standardized protocol solution, I further restrict our design space to use the IEEE 802.15.4 protocol.

For this case study, I assume that proximity sensors have to send a message every $500ms$ and that they have access to power sources. This is typically the case in a manufacturing line where proximity sensors can be reached by power cables or, more recently, can be powered by wireless [27]. On the other side, the sensor on the production piece has milder timing constraint and is only battery powered. For this case study, I assume that it has a delay requiremnt of 2 seconds and I optimize the communication infrastructure to minimize its power consumption while satisfying the application deadline with the required probability. As I already mentioned, this is a simplified scenario. In a real scenario, the timing constraints are much stricter and the number of proximity sensors much higher than the ones proposed in the case study.

However, as it will be clear from the results, the proposed constraints can be met by the 802.15.4 protocol, while stricter constraints would require another protocol with higher communication bandwidth. In Figure 4.17, I present the flow chart for the application using the Query Service, Command Service and decision blocks.

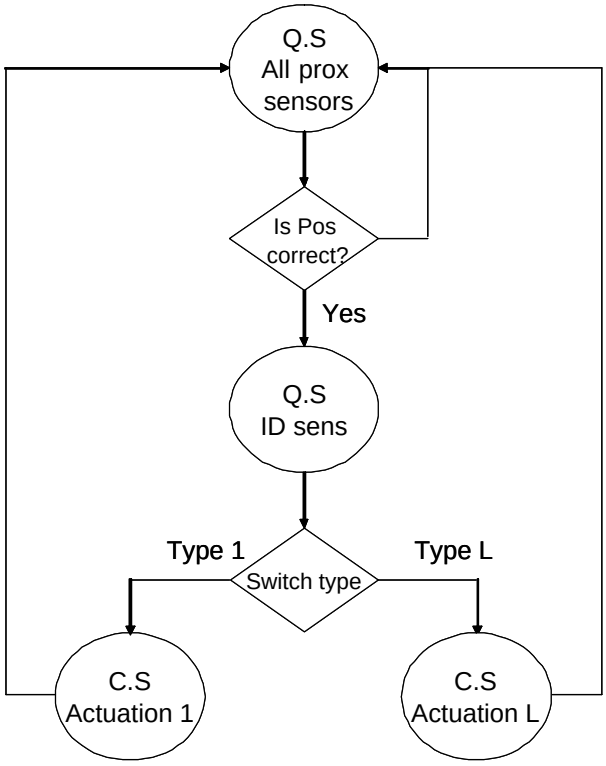

Figure 4.17. Flow chart for industrial automation case study.

4.4.1 Capturing Specification and Topology Generation

I formalize the description of the control algorithm using the Rialto framework. Specifically, I identify one virtual controller and six virtual sensors, with the appropriate sensing rate constraints. In Figure 4.18, I present a pseudocode for the application limiting the proximity queries to only one sensor. The communication between the different virtual components is described as a set of queries and commands that encapsulate the sensing rate and type, as well as the constraint on the E2E peformance

of the communication link. In the example, I set a deadline of 2 seconds with an outage probability of 0.2 for the query related to the piece ID. In Figure 4.19, I show the traces relative to the requirements on the different links. Starting from those requirements, I can synthesize the network.

```
//Virtual Controller

VC
Connections:
VS1,VS2,VS3,VS4,VS5,VS6

Cycle 500ms;

While(true){
        //Query for proximity
        Q1=new query(1,0,avg,prox,10,init,t1,50ms,e-2);
   //Query for piece id
   Q6=new query(1,0,num,ID,v,v,2s,0.1);

        if (evaluate==false){
       send(Q1,VS1);
         await(VS1){
              response1=receive(VS1);
         data1=response1->data;
                  attribute1=response1->attribute;
                  if ((attribute==prox)&&(data1>threshold_prox))
                    flag1 = true;
                  else
                        flag1= false;
                 }
        }
                                                    }
 if (flag1 && flag2 && flag3 && flag4 && flag5){
        evaluate = true;
send(Q6,VS6);
await(VS6){
        response6=receive(VS2);
   data6=response6->data;
        attribute2=response2->attribute;
        if ((attribute2==id)&&(data2=exp_id))
    start actuation;
        else
                send error message;
        }
        evaluate=false;
}
```

Figure 4.18. Rialto Model

```
Control Cycle Time Scope= 100msec;

Sensing:

VS1:
 prox:
  10 sam/sec;

VS2:
 prox:
  10 sam/sec;

VS3:
 prox:
  10 sam/sec;

VS4:
 prox:
  10 sam/sec;

VS5:
 prox:
  10 sam/sec;

VS6:
 ID:
  1 sam/120sec;

Communication:

VS1-VC:
 50msec,e-2;

VS2-VC:
 50msec,e-2;

VS3-VC:
 50msec,e-2;

VS4-VC:
 50msec,e-2;

VS5-VC:
 50msec,e-2;

VS6-VC:
 2sec,0.1;
```

Figure 4.19. Requirement Generation

Because of the relatively mild sensing requirements, I decide to use a single sensor for each virtual sensor.

4.4.2 Implementation and Results

According to the application requirements, I assume that the proximity sensors independently perform periodic sensing and pushing of data to the PLC. Consequently,

our design space is resctricted to the determination of the optimal pulling rate of the wireless node on the cart to minimize the wake up rate, while satisfying the E2E communication requirements. Using the lower bound derived in Section 3.6, I know that with a wake up period of two seconds, the expected E2E success rate is around 0.9 which is what I need to satisfy our requirements.

I implemented the proposed scenario on a testbed environment at U.C. Berkeley, using 6 Telos motes, a Stargate base station connected to a laptop, emulating the case study environment. I developed the communication protocol on TinyOS2, and set the polling rate of one of the nodes (that emulates the node on the cart) to two seconds, I changed the pushing rate of the other five (the proximity nodes) from $10ms$ to $1s$, and I accumulated the statistics on the outage events for a deadline of two seconds to verify the accuracy of our general model as well as the prediction for the working point. In Figure 4.20, I present the results.

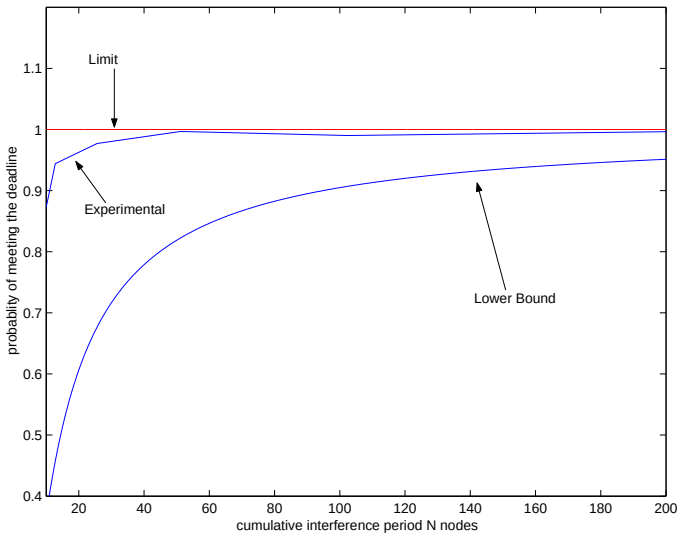

Figure 4.20. Testbed results

On the x-axis, I have the cumulative push period of the five proximity sensors (which is given by the push period of each sensor devided by the number of sensors), while on the y-axis, the observed success rate of the query. As it shows, the observed

success rate lays within our bounds and it shows a trend which is comparable to the lower bound. Consequently, using the lower bound to decide the working point of the system is a conservative but appropriate decision. Furthermore, I notice that at the desired working point, I estimated a success rate of 0.9 instead of an observed success rate of 0.97, which complies with the required 0.9.

Notice also that trying to stress the communication infrastructure to support stricter delay requirements substantially decreases the system reliability. That is because the selected communication protocol does not have enough bandwidth to support such requirements. I believe that the capability of quick estimating the feasibility of an application given a set of design and implementation constraints is an important added value of our methodology.

Chapter 5

Dynamic Mapping

Although most of the current wireless sensor network systems can be synthesized using the *Static Mapping* approach, the possibility of having wireless sensor network support different applications at the same time and interoperate with other commmunication protocols (i.e. WiFi) in a more heterogeneous environment, will inevitably drive the need for more flexible system architectures. In some domains like logistics and ambient intelligence, the push to develop flexible platforms that allow to connect the physical world with the human world are already evident [26]. Most likely, similarly to what happens in the integrated circuits world where there is space for both ASICS and microprocessors, there will be classes of applications that will continue to require higly specialized and optimized designs where the static mapping approach will be used, and other classes of applications where flexible platform capable to dynamically reconfigure the network and use the available resources for various applications.

Specifically, I define a *Dynamic Mapping* problem a WSN design problem where either of the following applies:

1. The application cannot be modeled as a cyclic control routine. This is the

typical case where a human can directly query the network with some handheld device.

2. There is more than one independent application sharing the same WSN resources. This may happen in a network with different controllers that are programmed with different applications that require data from the same sensors and these applications do not necessarily know about each other. This is also the case where there are independent networks with independent applications, but that use interfering protocols (i.e. Bluetooth and ZigBee). In this case, the shared resource is the channel bandwidth.

In these scenarios, everytime a controller initiates (or receives) a query, it has to observe the current capabilities of the network and take a run-time decision on which sensing and communication resource to use. Consequently, to perform an accurate mapping, it is required a lot of updated knowledge. This knowledge can be obtained by flooding the network with availability requests, but it may be not practical for networks of reasonable size because the multihop transmission chain may be so slow that the actual information becomes outdated before reaching the final destination, and most of all it may involve significant transmission overhead with consequent energy penalty. There have been some proposals for middlewares that solve this dynamic mapping [38]. Like in the case of microprocessors, for a dynamic mapping middleware to be succesfull, it has to find a sweet spot in the performance vs. flexibility trade off. Differently from [38], I believe that such a middleware should provide the necessary flexibility to offer E2E performance guarantees whenever it is possible, and gracefully decay the confidence in the guarantees whenever the information on the network is incomplete, eventually switching to best effort operations if only a first hop view of the network is available.

In this chapter, I present *FlexWSN*, a novel flexible platform that is exactly aimed

at solving these issues. It supports single controller networks as well as networks in which different controllers are running independent operations over a shared network infrastructure. It is composed of an application interface that is used to program the controllers of the network, a middleware for the controllers that sits between the application and the protocol stack (or stacks in case multiple protocols are supported) to optimally map the queries onto the possible quality of service, and a middleware for the sensor nodes that is necessary for the initialization and maintenance operations of the network. Thanks to the middleware, FlexWSN is compatible with any communication protocol that the different controllers or nodes support, and it adapts the quality of the delivery to the network dynamically changing conditions, using E2E performance estimations whenever possible. The application interface is an extension of Sensor Network Service Platform proposed in [22] and further refined in [54].

In the rest of the chapter, first I overview the proposed system architecture, then I provide a Platform Based Design formalization of the dynamic mapping strategy, and then I introduce the software infrastructure that implements such strategy. The last section is dedicated to the presentation of a case study on building monitoring and actuation with three conflicting communication protocols sharing the same network infrastructure.

5.1 System Overview

Similarly to the previous chapter, I distinguish three types of functionalities in a Wireless Sensor Network (WSN): sensing, control, and actuation. The control functions are the brain of the network. Their task is to ask for data to the sensing functions using Queries, collect the data, decide actuation, and communicate to the actuation functions using Commands. Usually, the control functionalitities are implemented on larger devices with networking capabilities that have access to power resources called

base stations (in the case of industrial automation PLCs), and sensing functionalitites are implemented on power constraint wireless enabled devices. The devices used to implement the actuation functionalitites depend strongly on the specific application and they may or may not be power constrained.

I identify two platforms and level of abstractions.At the highest level there is the usual Sensor Network Service Platform (SNSP) that is a collection of services that is combined to describe an application. The Application is the control algorithm and in a dynamic WSN there may be different applications, possibly deployed in different times and implemented in different controllers, active at the same time. These applications do not necessarily know about each other. For example, if there is a single meta application that was previously mapped in a distributed fashion on the different controllers, it is reasonable to assume that each controller has information on the type of operations that the other controllers are running. Conversely, if each application was programmed independently, then it is safer to assume that a controller does not directly know about other controllers, and it may only infer other presences observing some particular behaviors in the network. For the rest of the chapter, I consider this second case since it is the most challenging to manage.

At the lowest level, there is the Network Architecture Platform (NAP). This platform is the union of previously described SNIP and SNAPP(see Chapter 3). The reason for collapsing those two platform is that while in a static mapping problem the user has the possibility of selecting a topology and a communication protocol for a specific application, in a dynamic setting the options are restricted to using an already deployed sensing, actuation and communication infrastructure. Consequently, a NAP instance is the composition of the sensing, actuation and communication capabilities of the WSN that are deployed to serve the application. In case of multiple applications running at the same time, resources are shared among the different applications. Notice that during the lifetime of a WSN, a NAP instance is a time varying element,

because it may be upgraded with new nodes or protocols or nodes may die for energy depletion or simply be removed. A NAP instance is charachterized by:

1. Its sensing/actuation capabilities. This is the location and type of sensors and actuators that are present in the network and the id's of the nodes that support these resources. This is a run-time asset in case of addition of new nodes or more in general of network upgrade.

2. A discrete set of supported communication QoS classes and the id's of the nodes that support them. In general, a NAP instance can support more than one protocol depending on the RF and Physical layer charachteristics of the components. I define a QoS as the combination of a communication protocol and a set of parameters for that protocol.

3. A congestion status. This is the run-time load of the network when it is used by different applications. This status can be notified either explicitly by sending a specific message, or implicitly by refusing service.

Once the platforms are defned, I need to specify how the applications are mapped onto the NAP instance (see Figure 5.1).

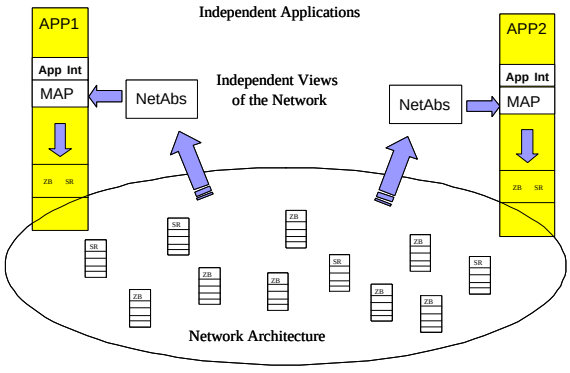

Figure 5.1. Overview of dynamic mapping architecture

I assume that independent applications are programmed at different controllers. Consequently, each controller has its own application to map that, in the worst case, is independent from the others and does not have prior knowledge ot their existence. The mapping is performed dinamically on a "'query-by-query"' basis, meaning that everytime the user specifies a new query, a middleware in the controller evaluates the content of the query and decides which sensor (or group of sensors) in the NAP instance to contact and with which communication class to use.

Because of the wireless and heterogeneous nature of the system, it is possible that at a given time a controller does not have a complete or updated knowledge of the NAP instance. Furthermore, different controllers may have different "'views"' of the NAP instance. These views are called Network Abstractions (NetAbs). Each controller has its own NetAbs and it is an abstraction of the subset of the capabilities of the NAP instance that the controller can observe. When a new query or a command is generated, the controller has to look at its current NetAbs and map the query on an appropriate communication QoS to satisfy the delay and error rate constraints of the query and minimize the energy consumption of the power constraint nodes of the network.

5.2 PBD perspective

In this section, I present the algebraic formulation of our synthesis approach for the dynamic mapping problem. With reference to Figure 5.2, I assume to have two independent applications programmed on two different controllers and supported by the same network.

The first difference from the static mapping approach is that the mapping happens on a query by query basis, hence the functional description is a single query. Except

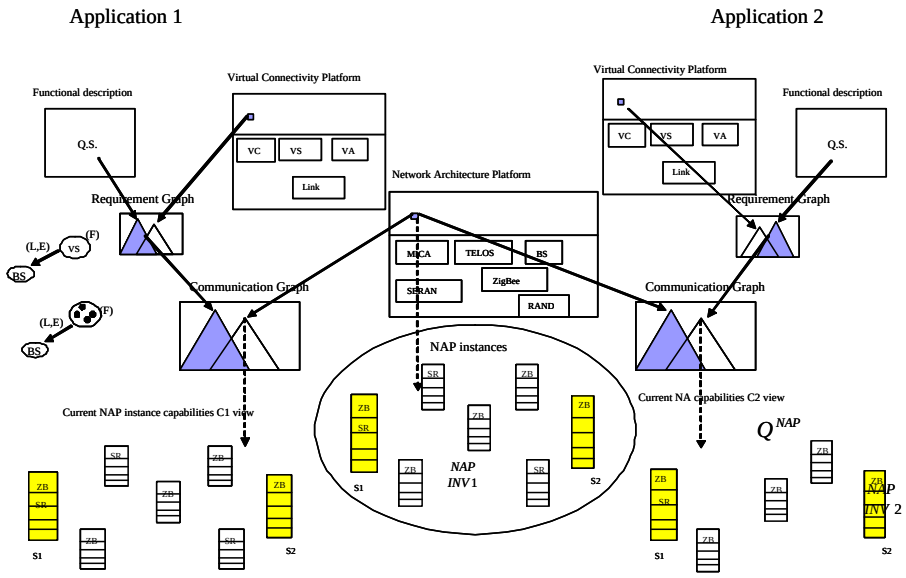

Figure 5.2. PBD interpretation for dynamic mapping problems

for this detail, the first step of the mapping process is the same as an instance from the virtual connectivity platform is selected and a requirement graph generated.

The next step is substantially different from the static mapping case. First, there is a new platform, the Network Architecture Platform (NAP). The agents of NAP are physical nodes, protocols and links. In an instance, each node must be connected with at least one communication protocols and different nodes can be connected with links if they support at least one common protocol and they are in communication range. In Figure 5.2, there is an example of a NAP instance with seven nodes, out of which two are controllers and five are sensors. Furthermore, one of the controller supports two protocols (SERAN and ZigBee), while the other only one (ZigBee). Out of the remaining five nodes, three support ZigBee and the other two SERAN. Links between compatible nodes are omitted for a better reading of the figure.

The common semantic domain of this step is the same instantiated network domain $\mathcal{Q}^{in}$ as in the static mapping case. An instance of this domain is a complete network of physical nodes with a communication protocol and tuned protocol paramters.

The usual partial order can be defined in this domain where a higher element is the one with minimum topology and looser end to end communication performance. Notice that the ordering of E2E latency performance is intended in the sense of the distributions. Specifically, call η and ν two possible solutions (i.e. combinations of protocol and parameters choice) and $F_\eta(\tau) = P[Delay_\eta \leq \tau]$, $F_\nu(\tau) = P[Delay_\nu \leq \tau]$ their E2E delay cumulative distribution functions, I say that

$$\eta \geq \nu \Leftrightarrow F_\eta(\tau) \leq F_\nu(\tau), \forall \tau$$

In general, solutions from different protocols are have different delay distribution profiles and may be not comparable, while different parameter combinations within the same protocol are usually comparable.

Mapping the rquirement graph $\mathbf{r_g}$ in $\mathcal{Q}^{in}$, I obtain a cone where the higher point is the network with a controller, the minimum number of sensors required to performed the virtual sensing function, and the loosest possible protocol solution supported by that node that satisfies the E2E requirements.

The mapping of the NAP instance onto the $\mathcal{Q}^{in}$ is performed through Ψ_{inv}^{NAP}. This function represents the previously described Network Abstraction and it opens a cone in $\mathcal{Q}^{in}$ where at the highest point there is the network topology recognized by that controller (that is not necessarily the complete topology of the NAP instance), with the loosest possible communication solutions. Note that Ψ_{inv}^{NAP} is time varying and different for different controllers. For example, in Figure 5.2, Controller 1 is able to see the whole network, while the other controller sees only the nodes that support ZigBee.

The intersection of the two cones generates a third cone that represent all the valid solutions for the mapping of the current query. Similarly to the static mapping case, I select an element which is high in this cone. In the next sections, I present the software infrastructure that implements this methodology.

5.3 Application Interface

Following the approach introduced in Rialto [54], the application is described in terms of queries, commands and conditional statements.

Queries are requests for data from a specific controller to a sensor or a group of sensors. Commands are requests for actuation to an actuator or a group of actuators. They are charachterized by the following fields:

1. Generator: the id of the controller
2. Type: Query or Command
3. Query ID: identificator of the query/command
4. Attribute: the type of data (i.e. vibration, position) in case of the query, or the type of actuation (i.e. switch off) in case of a command
5. Accuracy: how accurate the data has to be
6. Target: the intended receiver of a query/command. This could be either the id of a specific node, or a cluster of nodes, or even a higher level specification such as a geographic location (i.e. kitchen, living room, next cell in an automation line). If a high level specification is given, it is the duty of the middleware to select the appropriate physical node for that query.
7. Deadline: E2E latency requirement on the query/command
8. Reliability: error rate requirement on the query/command

A response to a query/command a similar format with the difference that the Generator is the sensor/actuator, the Target is the controller that originated the

query and there is an extra field, called Response, where in case of a query, the requested data is written, and in case of a command an acknowledgement is placed.

Similarly to Rialto, since I am interested only in the efficiency and guranatees of the communication infrastructure, I do not specify a model of computation to describe the interaction between queries, query responses and actuations, which constitute the control algorithm. I prefer to leave that choice to the application developer who can select an appropriate model of computation depending on the actual application domain, as long as queries and commands are passed down to the middleware with the above mentioned format. I believe that this separation between the description of the application and the dynamic synthesis of the communication infrastructure is fundamental to allow extensive reuse of legacy and proprietary control software.

5.4 Middleware: RTNOS

The Real Time Network Operating System (RTNOS) is the middleware that is implemented at each controller. It is positioned between the application and the protocol stacks, it receives queries and commands from the application, and query responses and acknowledgements from the network that are directly reported to the application.

When a query or command arrives, the middleware must perform three operations:

1. Context mapping: match the sensing/actuation requests of the query/command with the sensing/actuation capabilities of the network identifying the specific node (or set of nodes) to forward the query/command.

2. QoS mapping: select the appropriate communication protocol and combination of parameters to reach the node

3. Dispatch the query/command or holding it in a queue in case the network is already congested.

As I show in Figure 5.3, the middleware is logically divided in three blocks: Dynamic Mapper (DynMap), Network Abstraction (NetAbs), and Scheduler.

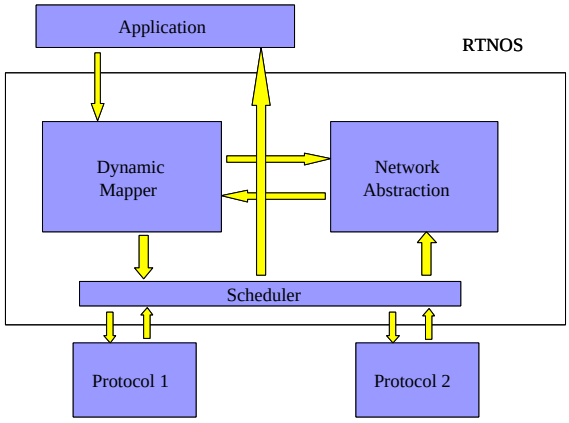

Figure 5.3. Middleware overview

5.4.1 NetAbs

To perform context and QoS mapping, it is fundamental to have an accurate abstraction of the Network Architecture. This abstraction has to provide a run-time picture of the current sensing, actuation, and communication capabilities of the network.

The NetAbs is implemented using a graph abstraction. The nodes of the graph represent the nodes in the network that the controller is aware of. Each node in the graph is charachterized by:

1. Node id: address

2. Attribute: sensing/actuation capability

3. Location: for example kitchen, living room, or cell number in an automation line

4. Supported Communication Protocols: for example ZigBee, Bluetooth

5. Free protocol parameters: parameters that can be modified to optimize the communication infrastructure, for instance the wake up and polling rate of nodes that support IEEE 802.15.4, togheter with the estimated distribution of E2E delay associated with the selection of those parameters.

Later in this section, I describe how the nodes sign up with the controllers to help them building the graph abstraction. As usual, the most delicate step in building and updating the graph abstraction is the estimation of the E2E delay distribution associated to specific configuration of protocol parameters. I use a procedure that is similar to the one outlined for the static mapping problem. Start from single hop behavior with respect to the different parameters (including probabilities of retransmission in case of collision/lost packets), this usually is available as a distribution. Once that is derived, the routing part has to be evaluated.

If the routing is a scheduled and deterministic, than I simply evaluate the schedule. Otherwise try to find an estimation of the required number of hops and use Gaussian additive model. This step is simple for some particular protocols where the number of hops (routing strategy) does not vary significantly depending on the connectivity. This is the case of geographic based routing [14, 44, 47]. It is hard to generalize at all the possible routing solutions, because in some of the most popular routing algorithms for WSN (i.e. MintRoute [45] and derived), the next hop is selected depending on an estimation of current local connectivity. However, these routing algorithms either flood the network or estimate offline the run-time connectivity and report in a parent child fashion the distance to each destination in terms of number

of hops. Consequently, the controller may just poll the first of these nodes to see a current estimation of how many hops are required to reach a destination.

The Dynamic Mapper can access the NetAbs with two primitives: Context Resolution and QoS Resolution. The first one is used by the DynMap to perform the context mapping for a query/command. It triggers a search in the graph structure to indentify all the nodes that match the attribute and the location of the query/command, and it returns the ID of all the nodes that are candidate receivers for that query/command. This information is then pushed back to the Dynamic Mapper. Similarly, the QoS Resolution primitive is used by the DynMap to perform QoS mapping. The primitive specifies a target (node or cluster) and it triggers a search in the graph structure that returns the different communication alternatives to reach the target, with relative performance. This information is pushed back to the DynMap as well.

The NetAbs is also accessible from the Scheduler using a Network Update primitive. This primitive is associated to the reception of a packet that is not a query response or a command, but it contains news to update the graph structure, such as a new node joining the network, a node changing its sensing/actuation/communication capabilities, or a node communicating its intent to disappear. This primitive triggers operations to manipulate and update the graph structure.

5.4.2 Dynamic Mapper

The DynMap is responsible for selecting the appropriate node to forward the query/command to, as well as the communication protocol and parameters to reach it (see Figure 5.4.

When a query first arrives from the application, the first operation is to temporary save the information in the query and prepeare a Context Resolution primitive to send to the NetAbs. This primitive contains the information on the attribute and location

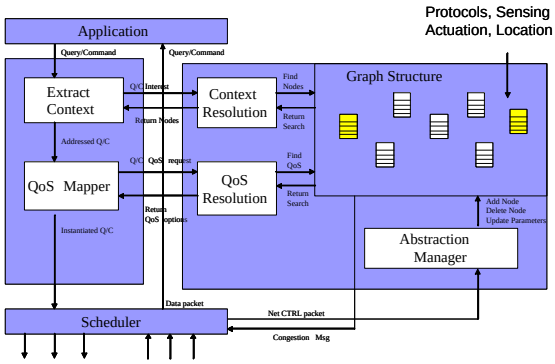

Figure 5.4. Dynamic maper overview

of the query, if the query already specifies a specific node, that information is also inserted. The NetAbs replies with a primitive called Candidate Target. The primitive may return:

1. Single ID: there is only one node who satisfies the attribute and location requirement, or if a specific node was already specified in the query/command and it was identified in the NetAbs.

2. Multiple IDs: this happens when there are multiple nodes that satisfy the requirements. In this case the DynMap has to select the one to forward the query/command to. The user may specify a preferred policy such as a deterministic round robin between the different nodes, or reaching all the nodes and then asking for an aggregate data such as average. The default operation is a random selection among the candidate nodes.

3. No ID: this happens when the NetAbs is unable to find a node that satisfies the requirements. This case will be discussed more in detail in the next section.

Once a target node is selected, the DynMap has to select an appropriate communication protocol. For this reason, it sends a QoS Resolution primitive to the NetAbs that responds with a QoS Candidate with the different options to reach the

target node, together with the E2E delay distribution associated to that option. In case there is more than one option, the options are ranked according to their delay distribution profile as explained in the previous section. The DynMap selects one of the solutions among the lowest ones that satisfy outage constraints, and specifically the one that belongs to the protocol that had the least usage in the near past. Since communication speed directly trades off with energy consumption, the selection goes to a solution that minimizes energy consumption and shares fairly the communication load among the communication resources.

Once a communication solution is selected, a Forward primitive is sent to the Scheduler with the same fiels of the query, plus the target node and selected communication solution.

5.4.3 Scheduler

The Scheduler has the following duty:

1. Forward the transmiting packets to the appropriate prococol stack
2. Check the received packets and forward them to the Application if they are a query response or an acknowledgement, or forward it to the NetAbs if they are network update packets

The Scheduler has also the possibility of temporarily stall the transmiting packets and store them in an output queue if a specific message is sent by the NetAbs that notifies a congestion status in the network together with the expected duration of the congestion period. The packets is eventually released by the Scheduler when at the expiration of the expected congestion period, or because a specific message was sent by the NetAbs that notifies that the congestion is over.

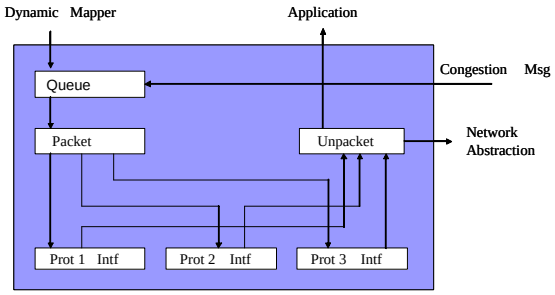

Figure 5.5. Scheduler overview

5.4.4 Registration and Maintenance

When a new node is added to the network, its first goal is to communicate its presence to as many controllers as possible. To do that, it floods the network with a Registration packet that contains information about the ID of the node as well as its sensing/actuation capabilities, location, and supported communication protocols. The way this message is propagated through the network, it depends on the actual communication protocol. When a controller receives this message, the Scheduler forwards it directly to the NetAbs that updates the graph structure.

If a node updates some of its sensing/actuation capabiltities or it is moved to another location, or it is to be disconnected and replaced, it floods the network with a node update packet that the controllers will forward to their own NetAbs.

5.5 Scalability and Print Reduction: the MetaNet

The most critical task for the RTNOS is maintaining and manage the graph structure of the NetAbs If the controllers are implemented on devices with limited memory capabilities, scalability becomes problematic. Furthermore, even if the controller has enough memory to allocate to the NetAbs, if the network has too many nodes, it may be impractical to propagate all the network update information to all the controllers.

To cope with these problems, I developed a solution that is able to explore the trade off between accuracy of the abstraction of the network architecture with memory and overhead efficiency. I reduce the size of the NetAbs to include only a local view of the network. How local the NetAbs view must be, that depends on the memory capabilities of the hosting hardware platform. In some cases it can be limited to the first hop neighborhood, or expended to two hops and so on.

I introduce a new block, called the MetaNet, that contains information regarding the closest controllers and how they can be used to reach remote sensing/actuation resources that are not directly available in the reduced NetAbs. Specifically, to reach a remote resource, the controller that initiates the query/command has to forward the query/command through a multihop chain between different controllers until it reaches the controller that has that specific resource in its NetAbs. The initiator controller does not necessarily know the complete sequence of the multihop chain, but it knows which is the first hop in this chain. When the first hop is reached, it checks the final destination and forwards it to the next hop in the chain, and so on. In Figure 5.6, I show the data structure of the MetaNet.

Ctrl ID	S2	S3	S4	
Next Ctrl Hop	S1	S2	S2	
Num Hops	1	2	3	
Target 1			SR1	
Target 2			SR2	
Target 3				
...				

Figure 5.6. MetaNet structure

The data structure of the MetaNet is built and augmented progressively anytime a new query/command is issued and no match is available in the current NetAbs or MetaNet representations. Consider the case in which a new query/command generated a no ID response in the Candidate Primitive from the NetAbs. This can mean two things, either the resource is not present at all in the network, or it is present

but the controller does not see it through the NetAbs. Consequently, the DynMap asks the MetaNet if the specific resource is present using a MetaContext Resolution primitive. The MetaNet checks its data structure and if the resource is found, it returns the address of the controller closest to the resource, the next controller in the hop chain, the expected number of hops, and the estimated delay distribution to reach the final destination. With these information, the DynMap generates the appropriate packet to forward down to the Scheduler.

However, if that resource is invoked for the first time, then the MetaNet does not have it in its data structure and a resource discovery procedure has to be invoked. There are many procedures proposed in literature that can be used usually involving different flavours of flooding. FlexWSN supports any procedure of logical flooding of controllers (multihop chains of sensors and actuators can be used to bridge the communication between two controllers if need be) until the target resource is identified and the data retrieved. When the query response gets back, the query initiator records the identity of the last controller that forwarded the information. It updates its MetaNet data structure associating the resource with the final destination controller and the controller that can used as a getaway for that resource. Consequently, if that resource is required again, a flood is no longer be required and this is due to the fact that while the topology and connectivity of WSN can change quickly, the topology and connectivity of the controller infrastructure is much more stable. However, should a controller be disconnected from the network, new routes have to be discovered to reach the resources in which the disconnected controller was involved.

This procedure is similar to cashing in computing architectures. Although there is no official benchmark, cashing seems a reasonable idea also for WSNs. Similarly to cashing systems, the memory print of the MetaNet can be bounded and every time a new resource has to enter the MetaNet, the least recently used can be removed.

5.6 Case Study

The above described middleware was implemented using the Omnet++ [31] simulation environment and tested with a cse study.

The scenario I selected is a typical building monitoring application. Consider the three stores building of Figure 5.7, a gas leakege monitoring and actuation network and temperature monitoring and actuation network are deployed in different rooms. The gas leakege network is implemented using Bluetooth nodes, while the temperature control network is implemented using IEEE 802.15.4. In each room there is a controller that supports both communication standards as well as a WiFi network for the communication between controllers in different rooms. I assume that sensors and actuators can communicate directly with the controller in their own room, and can randomly interfere with other rooms communication. With respect to the WiFi network, I assume the connectivity pattern described by the double sided arrows in Figure 5.7. Since the three communication protocols share the same band, I simulate a collision each time two packets, coming from close enough nodes, are generated within a time difference that is equal to a packet duration. In the NetAbs, to model the delay distribution of the different protocols, I used the model derived in Chapter 3 for IEEE 802.15.4, and models from licterature for both Bluetooth [63] and WiFi [64].

The application can be summarized as follows: each controller independently and periodically queries temperature and gas sensors, and if the data that is returned is over a certain threshold, then it sutomatically sends a command to the relative actuator. On top of that, on a random basis, each controller queries for gas and temperature also sensors in different rooms. Ideally, this random remote sampling models a sort of robustness against the failure of nodes in specific rooms.

In Table 5.1, I report the simulation parameters and the observed results. The constraints on queries for gas are of meeting a deadline of 1 minute with a success

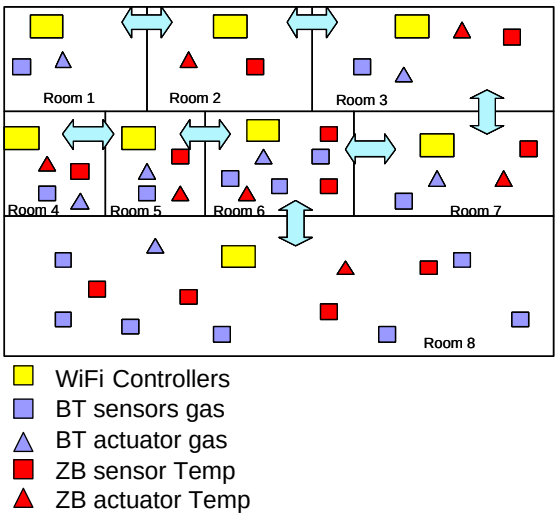

Figure 5.7. Scenario for dynamic mapping case study

Attribute	Query Deadline	Required Success	Observed Success	Command Deadline	Required Success	Observed Success
Gas	60s	0.9	0.91	20s	0.9	0.92
Temperature	120s	0.9	0.91	40s	0.9	0.91

Table 5.1. Summary of case study parameters for dynamic mapping.

probability of 90%, while for the command for gas, the costraint is 20 seconds with 90% probability. Constraints for queries and commands fo temperature are double the amount of time than gas, and with success rate 90%. As it can be seen, requirements are met almost with no slack. This means that the middleware was able to adapt the wake up rate of the diffefrent nodes to the minimum possible value that was still enough to satisfy the contraints, hence minimizing the energy consumption.

In Figure 5.8 and Figure 5.9, I show the results of the dynamics of the simulation. Specifically, on the x-axis there is the time (in 0.1 ms) and on the y-axis the percentage of queries and commands that were not able to meet their deadline for each room. As it can be seen, for both gas and temperature, there is an initial oscillation of the failure percentage, but after few minutes the performance converge to the required ones.

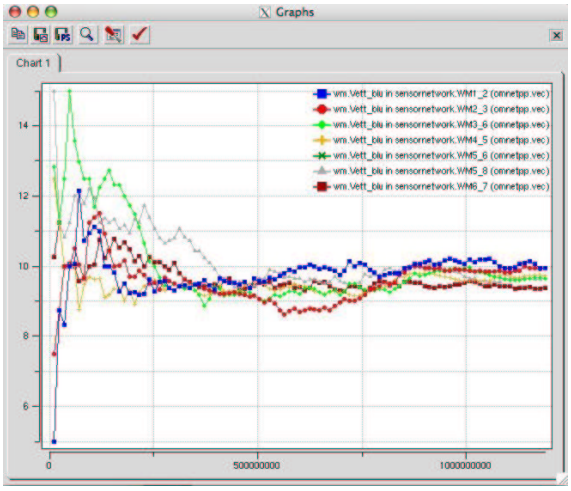

Figure 5.8. Time dynamics of E2E failure percentage for Zigbee queries (time in 0.1ms)

There have been different middleware and system architectures for WSN proposed in the last years. As it is usually the case, it is very difficult to compare these solutions, because there is no real benchmark for WSN applications. Furthermore, a system architecture is better judged only after some years, when the actual adoption or impact can be quantified. Consequently, I can only give some qualitative assesment of the proposed solution with respect to the others.

A recent proposal that had a good impact in the academic and industrial community is the SP platform presented in [38]. This platform is a middleware introduced at the link layer that is able to support any routing strategy on top of it, and any MAC protocol below it. This middleware is composed by a set of interfaces and a neighboring table that continuously monitors and ranks the quality of the links with the first hop neighbors. Depending on the link qualities, and final desctination of the incoming packet, a speficic next hop neighbor is selected to act as the next hop. This solution is tailored at the optimization of the single hop performance in a best effort fashion. Although the principles that drove the development of FlexWSN (end to end reliability) are very different, still if the NetAbs is reduced to a single hop view of the network, than our architecture converges to the SP one. On the other side, if there is

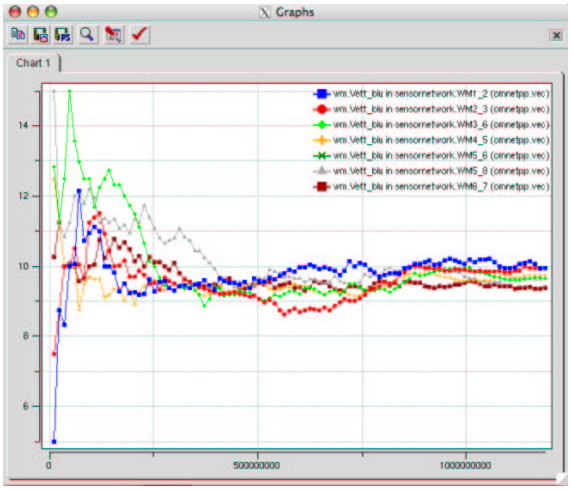

Figure 5.9. Time dynamics of E2E failure percentage for Bluetooth queries (time in 0.1ms)

a single controller and the NetAbs captures the view of the entire network, then the end to end performance of the architechture are maximized and the final solution is not very different from what it could be obtained using the static mapping approach. Consequently, the added value of this dynamic mapping proposal is that it gives an extra level of flexibility to exploit the trade offs between memory resources and mapping efficiency, and the claimi is that mst of the other architechtures are subcases of this methodology where a specific design point in this trade off is selected.

Chapter 6

Aggregation Algorithms

When presenting the case studies in Chapter 4 and Chapter 5, for the sake of presentation I always selected solutions where all the data was forwarded to the controller (or controllers) without performing any form of intermediate data aggregation. In this chapter, I try to correct this and introduce a set of aggregation algorithms that can be easily implemented on top of the described methodology to provide performance improvement.

Although great effort was dedicated to the development of algorithms for distributed computation for WSNs, the utilization of these algorithms in real life deployments is still very limited. Nevertheless, distributed computation algorithm are commonly considered a very important mean of reducing power consumption and consequently increase life time of a WSN.

The advantages of the introduction of aggregation algorithms is evident in clustered topologies application. In most of the promising future applications for WSNs, like building automation or manufacturing plants, the sensing areas are clearly identified and separated (i.e. rooms or robots [1, 59, 43]). Usually, sensors are deployed around these objects to perform mostly homogeneous sensing (i.e. temperature, vi-

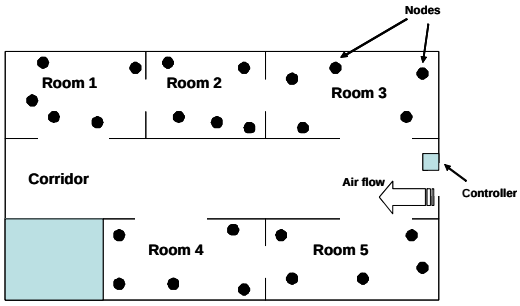

Figure 6.1. Building monitoring scenario

brations) and report their data to a single sink, that can be placed either somewhere inside the sensing area or at some remote location. These convergecast situations are obvious threats to network scalability, because the nodes close to the sink have to sustain the load of forwarding the packets generated by themselves, as well as the ones generated by more peripheral locations. As a result, nodes closer to the sink are candidate to early depletion, with catastrophic consequences on network life-time [43].

Data aggregation is a powerful mean to mitigate this problem. One or more nodes collect the packets from all the nodes in the cluster and then compact them in a single packet by computing some function (eg. mean) of the aggregate data and forward the result to the network sink. In general the parameters of an aggregation algorithms are optimized for either time or energy performance depending on the application requirements.

Consider a clustered application like the building monitoring aplication of Chapter 4 (see also Figure 6.1), where five sensors are deployed in each of the five rooms for sensing temperature and reporting data to the Controller.

A simple design is represented by a *centralized* approach where the Controller periodically receives a packet from every node in each cluster with the updated data on temperature of the corresponding room. Then, the Controller aggregates the values for each cluster to derive an estimation on the average temperature for the room. This centralized implementation, however, may be more or less efficient depending on the

cluster topology and the number of nodes per cluster. In fact, due to the multi-hop communication, those nodes that are closer to the Controller are required to support also the traffic due to packets coming from the distant nodes. Consequently, they dissipate more power, a critical resource in any WSN, and statistically end up having a shorter lifetime.

Since WSN nodes present also some computational capabilities, it is often preferable if the nodes on the same cluster locally compute the average vibration and select one of themselves to report the data to the Controller along the multi-hop chain. However, since node malfunctions and failures are not rare events in a WSN, it is critical to implement a fault-tolerant protocol which guarantees that multiple, if not all, nodes can take over the responsibility to compute and propagate the result to the Controller.

In the next two sections, I present two approaches to data aggregation for clustered wireless sensor networks. The first one is based on a novel adaptation of gossip based algorithms for WSN, the second, called EERINA [65] is a new algorithm that combines robustness and energy efficiency.

6.1 Gossip Based Algorithm

In this section, I show how the distributed computation can be implemented robustly and efficiently by using gossip-based algorithms on top of a communication infrastructure that takes advantage of the characteristics of clustered topologies with very litle overhead on design complexity. I conclude demonstrating how a *hybrid* design that properly combines the centralized and distributed approaches offers:

1. a drastic reduction of the average energy consumption;

2. a fair distribution of the energy consumption throughout all the nodes.

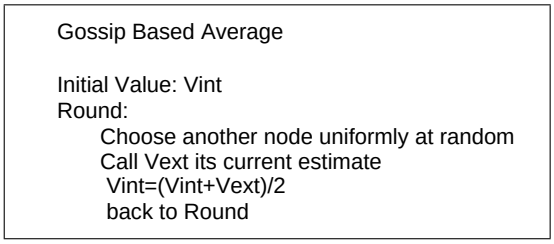

Figure 6.2. Gossip-based algorithm to compute the average among nodes.

Combined, these two results substantially improve the expected network lifetime.

Gossip-based algorithms have been proposed as a fault-tolerant approach to the distributed computation of aggregate functions. Consider a network of N nodes, each of them carrying some data information, the gossip problem is to make each node aware of the information stored in every other node [66]. This problem has been extensively studied for wired networks as it has important Internet applications [67]. The original goal of gossip-algorithms is to minimize the time to distribute the information to all nodes. In our case, the goal is to minimize the energy required under a given time constraint. For multi-hop communication in WSNs, however, the two targets are generally equivalent as they both correspond to minimize the number of messages.

Different gossip-based algorithms have been presented to calculate aggregate functions such as *average, max, min* among N values distributed over N different nodes. Although these algorithms may in general be quite complex, simplified versions are available for simple aggregate functions (such as calculating the sum or the average of the N values) [68].

Gossip-based algorithm are often described on the basis of a synchronous model of computation where message exchanges (events) occur along a sequence of "rounds" [68]: at each round every node picks randomly another node and exchanges information with it. In Figure 6.2, I report the gossip-based algorithm to calculate

the *average* that was proposed in [68]. If applied to a network of N nodes, this algorithms guarantees with very high probability that after $O(logN)$ rounds the value at each node is a good approximation of the final solution.

In [69], a variation of the gossip-based algorithm, called DRG, is introduced to take advantage of the broadcast nature of the WSN scenarios. Differently from the classical gossip-based algorithms, DRG does not have strong synchronization requirements among nodes, hence facilitating a WSN implementation. Unfortunatly, apllying DRG to our clustered environment where in each cluster all the nodes share the same communication channel and are within transmission range, would create a high number of collisions, hence jeopardizing the advantages of that approach.

6.1.1 Related Work

Directed diffusion [36] and TinyDB [37] are examples of query-based methods for high-level programming of WSNs. In both approaches, users specify the application as a set of tasks or queries (i.e. monitoring tasks) that the WSN must continuously perform. The request for data floods the network following a tree-based routing strategy until it reaches the desired node. Similarly, the query response is routed back to the originator of the query. On top of this, both approaches offer data-aggregation capabilities. Although I share the same vision of a WSN as a large distributed computing system where users describe the applications with simple and intuitive queries, I separate the design of the aggregate computation from the design of the communication infrastructure, thereby reducing the design complexity. Furthermore, leveraging an ad hoc communication protocol, I am able to offer clear guarantees on delay and loss rate for the application, instead of best-effort solutions as in [70, 36, 37].

An approach to develop algorithms for WSN applications based on a clear separation of specification from implementation is introduced in [35]. I push further such

orthogonalization of the design space by separating aggregate computation from the communication infrastructure that supports it.

When modeling power performance for WSNs, I focus on the contribution due to the radio-frequency (RF) activity because it dominates the contribution due to nodes internal computation [71]. There are two sources of RF power consumption: the power dissipated for transmitting packets and the power dissipated for idle listening the shared communication channel. Idle listening is proven to be the dominant factor for most WSNs [72]. This is particularly the case for real-time applications where the packet arrival must meet a hard deadline and, therefore, nodes must be awake for longer times to be able to relay a packet whenever it becomes available. The duty-cycle of a node is the metric that captures its overall RF activity. Considering the long lifetime requirements of the monitoring networks for manufacturing chains (usually few months) and the power performance of the most common WSN hardware platforms [8], a target duty-cycle performance around 1% for all the nodes in the network is appropriate.

Topology-independent communication protocols for WSNs like Task [70] cannot reach such target due to their generality and the inability to take advantage of the high-node density and clustered topology.One of the contributions of this work is to show how combining gossip-based algorithms with a protocol like SERAN (see Chapter 3 leads to further reductions of the duty cycle by a factor between 2 and 4.

6.1.2 Asynchronous Implementation of Gossip-Based Algorithms

Figure 6.3 reports an asynchronous implementation of the gossip-based algorithm of Figure 6.2 to calculates the average across N nodes in a WSN. I call *value exchange* the interaction of two nodes that results in an update of their average estimated value.

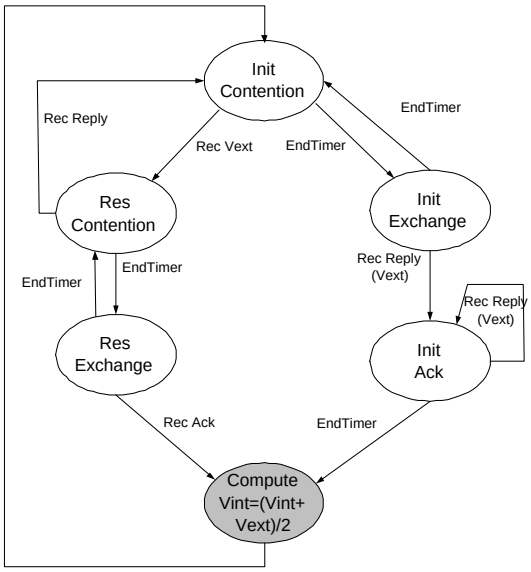

Figure 6.3. Asynchronous implementation of the gossip-based algorithm of Fig. 6.2.

Let Initiator and Responder be the two nodes participating in a value exchange. Let Vint denote the "internal" current estimation of a node and Vext the "external" estimated value that a node receives from another node. Then, the behavior of the asynchronous implementation of Figure 6.3 can be described from the viewpoint of a node as follows:

1. Init_Contention. This is the starting state. Every node in the WSN competes to become the Initiator by starting a randomized grenade timer and idle listening to the channel (random back-off contention procedure [50]). A node remains idle until either the timer expires (EndTimer) or it hears a non-colliding message from another node (RecVext). If the timer expires, the node becomes the Initiator and goes to the Init_Exchange state. Otherwise, it means that some other node won the contention and it goes to the Res_Contention state.

2. Init_Exchange. As the Initiator, the node broadcasts a message with its own current estimate Vint and then waits for a reply from the Responder with the latter's current estimate Rec Vext. When the reply is received, the Initiator moves to the Init_Ack state. If after a while no response has arrived, it means

that the original message was lost. This can be due to either bad channel conditions or to the less likely event that more than multiple node complete the random back-off almost simultaneously (i.e. within the time of fly of a message) and start broadcast thereby creating a collision. Either way, the node goes back to the starting state and competes again to become the Initiator.

3. Res_Contention. In this state, the node has just received a message from the Initiator and now has to compete with the remaining nodes to become the Responder. Similarly to the Initiator contention case, it performs a random back-off and listens to the channel. If a non colliding message is received before the timer expires, the node goes back to the starting state. Otherwise, when the timer expires, it becomes the Responder and goes to the Res_Exchange state.

4. Res_Exchange. The Responder sends a reply message to the Initiator with its current estimate and waits to receive an acknowledgement from the Initiator (Rec Ack). If it receives the acknowledgement then it moves to the Compute state. If no acknowledgment arrives in time, then the node assumes its reply message was lost. Similarly to the Init_Exchange case, the loss can be due to either bad channel conditions or a failure in the Responder contention that resulted in a collision. In either cases the node goes back to the Res_Contention state.

5. Init_Ack. The Initiator has received the reply message from the Responder and it sends an acknowledgement to the Responder. If after a while no other message is received, it assumes the acknowledgement went through properly and it goes to the Compute state. If instead it receives another reply message from the Responder, it means that the original acknowledgment was not received and it repeats the procedure.

6. Compute. In this state both the Responder and the Initiator have the two esti-

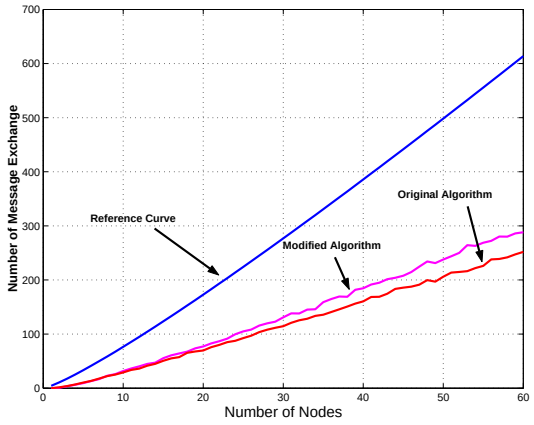

Figure 6.4. Algorithm speed of convergence.

mates and they simply update their Vint with the average of the two estimates. As this concludes the value exchange, the node is ready to restart the process.

When a particular application requires the computation of aggregate functions different from *average*, an asynchronous implementation of gossip-based algorithms can often be obtained by simply modifying the instructions in the Compute state. This is true for all the aggregate functions that can be approximated using linear combinations [68]. For example:

1. To calculate the Sum over N values, it is sufficient to add a multiplication at the end of the average calculation.

2. To calculate the $Max(Min)$ over a set of N values, I just have to change $V_{int} = (V_{int} + V_{ext})/2$ with $V_{int} = max\{V_{int}, V_{ext}\}$ ($V_{int} = min\{V_{int}, V_{ext}\}$).

Similarly to [69], it is possible to define a library of fault-tolerant distributed algorithms for WSN industrial applications.

6.1.3 Performance Analysis of the Algorithms

The asynchronous implementation of the gossip-based algorithm illustrated in Figure 6.3 converges to the final solution because it holds the "mass conservation" property [68]: to calculate the mean over the currently stored values at any point during the execution of the algorithm gives the correct average value. Nevertheless it is necessary to verify the speed of convergence of the algorithm, i.e. the number of value exchanges that are needed to approximate the final result.

For the synchronous gossip algorithm it can be shown that if there are $N\left(\lg N + \lg \frac{1}{\epsilon} + \lg \frac{1}{\delta}\right)$ value exchanges then all nodes approximate the final result within ϵ with probability $(1 - \delta)$ [73]. Our asynchronous implementation, however, does not ensure a perfect balance in the execution of the algorithm. In fact, due to the random back-off, there is the possibility that one of the nodes performs more value exchanges than the others, thereby slowing down the convergence of all the nodes to the final result. However, experimental results show that the impact of this event remains limited.

In Figure 6.4, I present simulation results that compare the average speed of convergence of the synchronous and asynchronous algorithms for different values of N and $\epsilon = 1\%$, together with the reference curve given by $N\left(\lg N + \lg \frac{1}{\epsilon}\right)$. Although the asynchronous algorithm converges slower than the synchronous one, it still remains within the reference curve. Consequently, given N and the target accuracy ϵ, I can use the reference curve to model conservatively the number of required value exchanges. Furthermore, I can introduce a node based stopping criterion for the iterative algorithm of Figure 6.3, namely when a node has performed $\lg N + \lg \frac{1}{\epsilon}$ values exchanges.

Finally, if I want to quantify the speed of convergence in terms not only of number of value exchanges but also in terms of elapsed time, then I need to consider the

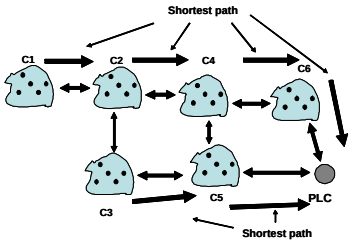

Figure 6.5. Connectivity for clustered topology

worst-case time that is necessary to complete a single value exchange. This time must be larger than the sum of the periods necessary to resolve the contentions for the Initiator and the Responder. I call τ this time and in Section 6.1.5 I show how it can be modeled.

6.1.4 Automatic Distribution of Computation

In Chapter 3 I introduced SERAN, a communication protocol for clustered topology and in Chapter 4 I presented a case study on building automation that uses SERAN as the underlying communication protocol. That solution is based on centralized approach, meaning that all the nodes send their data directly to the Controller. The diagram of Figure 6.5 illustrates the six clusters of sensor nodes and the inter-cluster connectivity structure. According to SERAN, data communication is routed via a multi-hop cluster chain along the shortest path from each cluster to the Controller. For example, a query on the nodes of cluster C_2 is serviced by having all the nodes of C_2 transmitting their value to the nodes in cluster C_4, and, in turn, the nodes of C_4 transmitting their value to the nodes in cluster C_6, and so on until the Controller is reached.

The medium access control is based on a combination of Time Division Multiple Access (TDMA) and Carrier Sense Multiple Access (CSMA) protocols. Time is divided into TDMA cycles, each cycle in TDMA slots and each slot in CSMA sub-slots

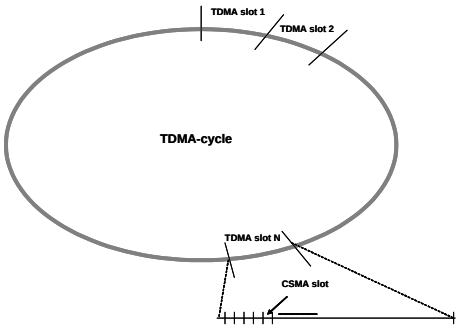

Figure 6.6. TDMA-Cycle representation.

(Figure 6.6). During each TDMA slot only one cluster can transmit and only one can receive. Further, the nodes of the transmitting cluster try to send their packets using a p-persistent CSMA protocol [50]: at every CSMA sub-slot each node attempts to send a packet with probability p, and it repeats this procedure until it receives an acknowledgement packet that confirms the success of the communication. Packets are broadcasted over the nodes of the receiving cluster. At the same time, nodes of the receiving cluster idle listen for incoming packets, and whenever a single non-colliding packet is detected, they start a random back-off contention procedure that is similar to the one discussed in Section 6.1.2: the first node to reply with an acknowledgement is the one who processes the packet while the other nodes discard it.

In Chapter 3, it is shown how the duration of the TDMA slots and CSMA sub-slots can be tuned in such a way that:

1. all the packets of the transmitting cluster are forwarded by the end of the TDMA slot;
2. requirements on end-to-end delay and packet loss are satisfied;
3. per node average power consumption is minimized.

Starting from this valid centralized solution, I analyze every single query that the

application requires and I compare the energy cost needed to satisfy that query with a distributed implementation against the centralized one.

The advantage of this iterative refinement procedure is that once the valid centralized solution is available, the modifications of the communication infrastructure to support a distributed implementation are relatively minor. Modifications are necessary only within the TDMA slot where, using a distributed algorithm, each node iterates through multiple executions of the asynchronous gossip-based algorithm until a sufficient approximation value for the aggregate function is computed. Once this approximation is reached, a single node (i.e. the node winning the first Initiator contention) takes the role of transmitting the result of the aggregate computation to the next cluster where eventually a single node takes the responsibility of receiving it. In the subsequent TDMA slots, only the packet with this aggregated result needs to be forwarded through the multi-hop cluster chain until it reaches the Controller. This is the main difference with respect to the centralized solution, where multiple packets are forwarded from each given cluster and the aggregate functions are computed by the Controller after processing them. Notice however, that both the structure of the TDMA/CSMA communication protocol and the cluster-to-cluster routing algorithm that are used in the centralized solution can be seamlessly used in the distributed solution. This is a key point because it enables the separation of the mechanism for computing the aggregate function from the design of the inter-cluster communication infrastructure, thereby minimizing the extra design complexity due to the introduction of the distributed aggregation capability.

With reference to Figure 6.5, assume N nodes for each cluster and consider the case of a query asking the average value of vibration intensity to the nodes monitoring the robot in cluster C_2. Using the centralized solution, $3 \cdot N$ message exchanges are necessary to reach the Controller and with SERAN this requires $Ntx_{cen} = 3N(A+1)$ packet transmissions(where A is a constant that depends on N). During each hop the

nodes of the receiving cluster must idle listen for incoming packets throughout the entire duration S of the TDMA slot. Consequently, the cumulative CSMA sub-slots listening time is equal to $L_{cen} = 2 \cdot N \cdot S$.

Assuming that the asynchronous gossip-based algorithm is able to converge within a single TDMA slot , the distributed implementation requires $N(\lg N + \lg \frac{1}{\epsilon})$ value exchanges between nodes of cluster 2, plus three more message exchanges to forward the result to the PLC. Hence, the resulting number of transmissions is given by the following equation:

$$Ntx_{dis} = 2\Big(N\big(\lg N + \lg \frac{1}{\epsilon}\big) + 3\Big) \tag{6.1}$$

Furthermore, since only one packet is forwarded to the next cluster, the nodes of the receiving cluster don't need to stay awake for idle listening throughout the entire duration of the TDMA slot. If $1/T$ is the fraction of the TDMA slot when receiving nodes must be awake, then the *cumulative listening time* is equal to $L_{dis} = 2 * NS/T$ CSMA sub-slots.

Consequently, comparing Ntx_{cen} and L_{cen} against Ntx_{dis} and L_{dis}, it is possible to determine which one is the winning solution when the Controller queries the nodes of cluster C_2. Since this comparison can be performed statically for each cluster in the WSN, it is possible to implement a hybrid strategy, where the application of the asynchronous gossip-based algorithm is decided on a query-by-query basis. The benefits of this approach can be quantified on the reduction of cluster activity.

6.1.5 Results

I complete here the discussion of the case study of the six rooms building. Specifically, I assume to deploy a cluster of five Mica nodes for a total of thirty nodes. The six clusters are connected as described in Figure 6.5. The Controller queries periodically each cluster in a round robin fashion for the average temperature. The

	C_1	C_2	C_3	C_4	C_5	C_6
Centralized	2410τ	1615τ	820τ	820τ	25τ	25τ
Distributed	306τ	227τ	148τ	148τ	69τ	69τ

Table 6.1. Overall RF activity costs for querying each cluster.

maximum end-to-end communication delay occurs between the first cluster and the Controller. I assume that the application requires that such delay does not exceed 20 seconds.

For a centralized solution , this design constraint translates into setting the TDMA slot duration to $1540ms$ and the CSMA sub-slot duration to $10ms$. Since I rely on the same contention procedures, I assume that $10ms$ work also for this solution as long as the asynchronous gossip-based algorithm is guaranteed to converge in a single TDMA slot. Assuming the required precision on the average value is $\epsilon = 5\%$, Equation 6.1 guarantees that the algorithm applied to a cluster of five nodes converges within $340ms$.

Since in a WSN the energy consumption due to the RF activity dominates the consumption due to the nodes internal computation [71], I can estimate the overall power dissipation in the WSN by considering the number of messages exchanged within the WSN and the idle listening periods that it requires. Therefore, to compare our distributed solution with the centralized solution , I can focus on cumulative RF activity and follow the approach outlined in Section 6.1.4. Assuming that a value exchange happens during the duration of a CSMA sub-slot, I can simply add the number of value exchanges to the number of listening CSMA sub-slots and compare the two solutions. Furthermore, I assume that for the distributed solution the nodes of a receiving cluster need to be awake for only 1/10 of the TDMA slot. This is equivalent to $154ms$, a time that is sufficient to ensure a single message exchange.

The estimated costs in terms of total RF activity that are necessary to complete

a query from the PLC to each of the six clusters are reported in Table 6.1 for both the centralized and the distributed solution. These results confirm that to service a query through a distributed computation of the aggregate function is not always convenient. In fact, due to the proximity of clusters C_5 and C_6 to the PLC, it is better if their nodes send the sampled values to the PLC that can use them to compute the aggregate function. However, for all other clusters, which are located further away from the PLC, the cost of running the asynchronous gossip algorithm is amortized by the saving of avoiding expensive multi-hop chains and the decrease of the idle listening duty. In summary, these results call for a hybrid strategy where the query is implemented in either a centralized or distributed fashion based on the particular cluster location.

I tested a hybrid implementation using Omnet++ and I compared its performance with the centralized solution proposed in Chapter 4. The experimental results are presented in Table 6.2. The first six columns report the average duty cycle per node for each of the six clusters of Figure 6.5, while the last column reports the average duty cycle per node across all clusters. From Table 6.2 we notice that:

- the hybrid solution offers a better overall average duty cycle performance by more than a factor of 4;
- the hybrid solution reduces the differences in power consumption across the nodes of different cluster.

These properties of the hybrid solution are due to the major traffic reduction in the proximity of the Controller. This leads to a reduction in the idle listening time for those nodes that are closer to the PLC as illustrated by the one order of magnitude decrease of duty cycle for cluster C_6. Ultimately the importance of this improvement is that it greatly reduces the likelihood of an early depletion of the nodes closer to the Controller, an event that would have catastrophic consequences for the network

	C_1	C_2	C_3	C_4	C_5	C_6	Avg.
Centralized	0.2	3.5	0.2	6.8	3.5	10.0	4.0
Hybrid	0.7	1.0	0.7	1.3	0.5	1.2	0.9

Table 6.2. Per-node average duty cycles (percentages).

connectivity and the overall application. In centralized solution, the only way to cope with the increase of the network activity in clusters close to the PLC is to increase the number of nodes in those clusters accordingly. The hybrid solution allows to overcome this issue without further deployment of resources.

6.2 EERINA

In this section, I present EERINA, an aggregation algorithm for wireless sensor networks applications characterized by clustered topologies.

There are two main approaches to data aggregation for clustered environments. The first one are the already presented gossip based algorithms. In general, gossip-based algorithms are extremely robusts (all the nodes reach the aggregate result), but they usually require a convergence time and energy consumption that is $O(NlogN)$ for a cluster of N nodes. This inefficiency is due to the use of point to point communication, without taking avantage of the inherent broadcast nature of the wireless channel. In [69], a variation of the gossip-based algorithm, called DRG, is introduced to take advantage of broadcasting. Unfortunatly, applying DRG to our clustered environment where in each cluster all the nodes share the same communication channel and are within transmission range, would create a high number of collisions and jeopardize the advantages of that approach. A further limitation is that nodes do not keep track of partial results, but only of the final result. Consequently, if the application

required an aggregation function that is not linear the gossip-based approach cannot be used.

The second one will be referred to as the *Cluster Head* approach. It consists of having a node in the cluster that is elected Cluster Head (C.H.) at the beginning of every aggregation round, and the remaining nodes sending their packets to the C.H. according to the underlying MAC protocol. The C.H. will be responsible for forwarding the data to the network sink. Usually, the C.H. is randomly selected at the beginning of each aggregation to ensure a fair distribution of the workload. LEACH [74] is a widely known example of this type of algorithm. In general these algorithms offer convergence times and energy consumption that are linear with respect to the number of nodes. However, good performance is usually paid by a lack of robustness against events such as the death of the C.H during the aggregation round, thatstops the aggregation procedure and loses all the data accumulated up to that point.

In section we introduce an approach that combines the efficiency of the cluster head algorithms with the robustness of the gossip algorithms. In EERINA, every node plays the same role initially, and only at the end of the aggregation phase the *Cluster Leader (CL)* is selected. Furthermore, EERINA takes advantage of the broadcast medium to minimize the number of transmitted messages and create a high level of data redundancy.

The combination of the late selection of the cluster leader and the bandwidth efficiency, ensures a very high degree of robustness with respect to node failures, malfunctions, or temporal disconnections, with very limited overhead and good timing and energy performance.

EERINA is completely orthogonal to the underlying communication mechanism to forward the data between *CL*'s and towards the sink. Consequently, it is very

scalable, and allows for local network changes (eg. node additions and deletions) without modifying the overall network structure.

I also developed a mathematical model to determine analytically the energy and time performance of the algorithm given a set of values for the key parameters. This allows us to tune the algorithm to an optimal working point for any combination of application requirement, hardware platform and topology without needing extensive simulations.

6.2.1 Algorithm

I consider a scenario in which sensors are deployed in clusters to observe clusterized physical phenomena. I assume that nodes within the same cluster are within direct communication range and that they all share the same channel.

I further assume that the aggregation task is organized in *rounds*, where each round starts with all the nodes in the cluster having a data packet (from a sensor) and finishes with at least one node collecting all the packets. At that point, the node can compute any function of the received packets, depending on the application requirements.

The goal of the algorithm is to complete the aggregation, while minimizing the energy consumption or the convergence time, depending on the application requirements.

An aggregation round is characterized by an *Initialization* phase followed by an alternation of *Exchange* and *Contention* phases.

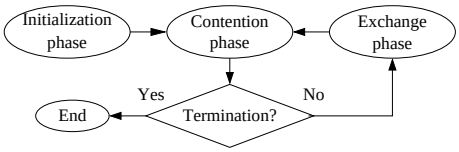

Figure 6.7. algorithm flowchart

Initialization phase

The goal of this phase is to allow all the nodes to repeatedly broadcast their data and to receive data from the other nodes. It starts at the beginning of the aggregation round (call this time $t = 0$) and it finishes at time $t = T_{con}$.

Assume there are N nodes in the cluster. Each node transmits its data following an exponentially distributed intertrasmission time of average $1/\mu_{TX}$. The duration of each transmission τ_{TX} depends on the application (packet size) and the phisical comunication bit rate.

The receiving process is also randomized. Each node wakes up for listening for a given deterministic time τ_{RX} whose value is selected as a function of τ_{TX}. The inter wake up times are exponentially distributed with mean $1/\mu_{RX}$.

When a node is neither transmitting nor receiving, it can safely turn off its radio to preserve energy. In the next subsection I show how to tune T_{con}, μ_{TX}, and μ_{RX} to efficiently cope with the design objectives, given τ_{TX} and hence τ_{RX}

At the end of this phase, each node has received and stored internally a random number of packets, which is different between nodes. This asymmetry then guides the *Cluster Leader* selection in the "Contention" phase.

Contention phase

The goal of this phase is to elect a *Cluster Leader* (*CL*) among the nodes in the cluster. The *CL* will be the one driving the exchange phase.

All the N nodes partecipate to the election. Call s_i the number of data packet from different sensors stored by node i. Each node starts a back-off timer whose duration is proportional to $N - s_i$. When the timer expires, the winner of this back-off contention assumes to be the *CL* and broadcasts a "Contention Packet" (CP). The CP is a special message with a string of N bits that are set to one if the data from the corresponding node node has been received by the node, zero otherwise. Consequently all the other nodes that receive the CP kmow if the *CL* already has their data.

It may happen that a node does not hear the CP while its back-off timer is still running. When its timer expires, the node assumes it is the *CL* and sends a CP. The nodes that already received a CP discard this second message. This ambiguity between various candidate *CLs* is progressively resolved in the folllowing contention phases.

Exchange phase

The goal of this phase is to allow the nodes whose value was not heard yet by the *CLs* to transmit more frequently and accelerate the convergence of the aggreagation procedure. At the beginning of the exchange phase there is one (or more) *CL*, while the other nodes know if their data was already stored by *CL*.

The nodes that have not been heard already by the cluster leader start transmitting their value with exponentially distributed intertransmission times with a mean $1/\mu_{TXE}$. The intertransmission time parameter is set such that the channel occu-

pancy is the same as in the initialization phase. That means that if the *CL* already has data from k out of N nodes, then $\mu_{TXE} = \frac{N}{N-k}\mu_{TX}$.

During this phase, the *CL* (or the *CL*s in case the last contention phase ended with more than one winner) is awake all the time to listen for the packets arriving from the nodes. The other nodes do not listen any more, since now reception of their packet is guaranteed. Hence they can turn off their radio whenever they do not have to transmit. This phase lasts until the next multiple of T_{con}.

Alternation

The aggregation procedure continues with a periodic alternation of the contention and exchange phases. Specifically, a contention phase starts at every multiple of T_{con}.

During these following contention phases, the back-off contention is repeated with all the nodes partecipating. Obviously, the *CLs* having a higher numbers of stored packets, also have a increasingly higher probabilities of been reconfirmed. In case the first contention phase ended with more than one node pretending to be the winner, these following phases will progressively reduce the probability of having more than one *CL*.

Another advantage of having several contention phases is that nodes can receive an update of the fact that their data was successfully received by the *CL* in one of the previous exchange phases. If this is the case, they stop broadcasting their data. Otherwise they keep broadcasting during the following exchange phase with a rate that is inversely proportional to the number of data that the *CL* is missing.

This algorithm is also beneficial for robustness against nodes death or malfunction. Assume for example that a node dies or suddently becomes disconnected during the aggregation round in which it is the *CL*, in the following contention phase another node is elected and the aggregation procedure goes on. This mechanism obviously

increases the time required to complete the aggregation, however it ensures a very high degree of fault tolerance.

The algorithm ends wheneither the *CL* accumulated the data from all the nodes or some data packets are still missing, but the corresponding nodes do not transmit their data at the required rate. The latter could happen if the nodes are either malfunctioning or unrechable. In either case, there is no point in continuing the procedure and the aggregate data is computed with the received samples. In any case, the termination is decided by the *CL* and communicated to the other nodes during the last contention phase.

Synchronization

Synchronization is needed at the beginning of the aggregation round, when the nodes are supposed to start at the same time. I do not specifically address this issue in this work, however for periodic monitoring applications this almost simultaneous wake up can be scheduled at the end of the previous round [43], or with a specific query [37]. Since EERINA is agnostic with respect to the system level architecture, it can be supported in any type of query based system.

6.2.2 Mathematical Model

We describe a procedure to derive a mathematical model to predict the timing and energy performance starting from the parameters of the algorithm. While the algorithm in principle can work with any mac, this analisys and the experimental results are derived assuming a CSMA MAC (consistent with MICA and TinyOS).

Specifically, from the application I know the number of nodes N and from the application and the hardware platform I know the time to complete a packet trans-

mission τ_{tx}. The parameters that I need to set are the transmission rate during the initialization phase μ_{tx} (for the exchange phases I already explained how to derive the transmission rates), the wake up rate for listening μ_{rx}, the periodicity of the contention phase T_{con}, and the duration of a contention phase τ_{con} (the difference is the duration of the exchange phase). First, I outline here the procedure to set μ_{rx} and τ_{con}, then I present a mathematical model to determine the relation between the remaining variables μ_{tx} and T_{con} from the time and energy performance metrics.

Fixed quantities: μ_{rx} and τ_{con}

The wake up rate is set to have at least one receiver awake at any given time with high probability so that almost no data transmission is completely unheard and wasted.

Since the inter wake up times of each node are exponentially distributed with a parameter μ_{rx}, the average duty cycle D is: $D = \frac{T_{on}}{T_{on}+1/\mu_{rx}}$. Consequently, the probability P_w that at least one node out of N is awake at a given time t, with $0 < t < T_{con}$, is $P_w = \big(1 - (1 - D)^N\big)$.

The duration of the contention phase τ_{con} is set so that the back-off contention procedure can be safely completed. For the contention round to be successful, the first bit of the first CP must arrive to the intended receivers before any other node starts sending his own CP. The worst case duration will be $N * (t_{fly} + \Delta_{sync})$ where t_{fly} is the fly time of a bit and Δ_{sync} is the worst case time sincronization uncertainty beetwen the N nodes of the cluster.

Expected time to converge w.r.t μ_{tx} and T_{con}

The first metric that I consider is the expected time to complete the aggregation $\bar{T_{tot}} = E[T_{tot}]$. Call M the random variable denoting the number of contention phases

needed to complete the algorithm. I have $E[T_{tot}] = E[M]T_{con}$. Consequently, I need to characterize the behavior of M with respect to μ_{tx} and T_{con}.

For this purpose, I model the cluster behavior with a Discrete Time Markov Chain (DTMC) where the state represents the number of packets that still have to be collected by the cluster leader at the end of a contention phase and a transition j states away represents the event of having j new packets arriving at the cluster leader during the follwing exchange phase.

Consequently, at the beginning of the algorithm the chain is in state N, and the aggregation is completed when the state 0 is reached. Notice that the chain is unidirectional, meaning that it can evolve only to lower states. It is also absorbing, since it stops at state zero This model is a simplification of the real life behavior because it does not account for the event of the death or malfunction of nodes during the aggregation, which should be sporadic enough to be neglected during parameter optimization. However, in the next subsection, I consider them in our simulations when I test the robustness of the algorithm.

To characterize the DTMC I need to consider the transition probability matrix Q. Since

$q_{i,j}$ = P(next state j — previous state i)=0 for $j > i$

Q is lower triangular. Call $C(i, i-j) = \frac{i!}{(i-j)!j!}$ the chose of i-j elements out of i, and c_i the probability that a single node (whose data has not been collected yet) of *not* is nott able to transmit to the cluster leader during the exchange phase, given that the chain is in state i. For $i \geq j$ I have:

$q_{i,j} = C(i, i-j) * (1-c_i)^{i-j}(c_i)^j$

Notice that c_i, can be decomposed into the conjunction of the probabilities of an

unsuccesfull transmission for each attempt during the exchange phase. Call the event an unsuccesful transmission $miss$ and the number of attempted transmissions t.

$c_i = \sum_{t=0}^{t=\infty} P(miss|t)P(t)$

Since I am assuming for this analisys a non persistent CSMA MAC, a single unsuccesful attempted transmission happens if during the time frame of $\tau_{tx} + t_{fly} = V$ seconds before the attempt, at least one of the remaining i-1 nodes uses the channel. Call *fail* the event of a single unsuccesfull transmission. Since t is a Poisson process whose intensity is $R_i * (T_{con} - \tau_{con})$, where R_i is the packet transmission rate per node at the state i, I have:

$P(miss|t) = P(fail)^t = (1 - e^{-R_i V(i-1)})^t$

and

$c_i = \sum_{t=0}^{t=\infty} (1 - e^{-R_i V(i-1)})^t * e^{-R_i * (T_{con} - \tau_{con})} \frac{(R_i * (T_{con} - \tau_{con}))^t}{t!}$.

Finally, I need to express the packet transmission rate R_i as a function of the optimizing variable μ_{tx}. Recall from Section 6.2.1 that the packet rate per node increases proportionally with respect to the number of remaining nodes, so that the cumulative traffic offered by the cluster remains the same. Hence I have $R_i = \frac{N}{i}\mu_{tx}$.

After characterizing the transition probability matrix P, I can now procede to determine the expected number of steps for the DTMC to reach the absorbing state 0 starting from state N. I can rewrite P as follows:

$$P = \begin{bmatrix} 1 & 0 \\ R & T \end{bmatrix}$$

where 1 is a single element matrix (the probability of remaining in the absorbing state), 0 is an all-zeros row vector of dimension N, R is a column vector of dimension

N, and T is an NxN lower triangular matrix. Call I_N the NxN identity matrix, and define $A = (I_N - T)^{-1}$[1].

Using [75], I can compute the expected number of steps to absorption from state N summing over the last row of A The total expected time to convergence will be:

$E[T_{tot}] = E[M]T_{con} = T_{con} \sum_{j=i}^{N} a_{n,j}$

Expected energy consumption w.r.t μ_{tx} and T_{con}

To calculate the expected energy consumption during an aggregation En, I consider $En(\mu_{tx}, T_{con}) = \sum^{(} M)_{r=1} C_r$, where C_r is the cost at round r and M is the number of rounds. Call N_{wutx} the number of wake ups to transmit, E_l the energy cost for low power listening to implement the CSMA protocol, N_{wurx} the number of wake ups to receive,C_w the power consumption for listening for arriving packets, τ_{rx} the listening time, N_{tx} the number of transmissions so far, E_{tx} the energy cost of each transmission, N_{rx} the numer of receptions, E_{rx} the energy cost for each reception, and E_k the average energy cost for each contention. I can rewrite the energy consumption in the first round (the initialization phase) as:

$En(\mu_{tx}, T_{con}) = N_{wutx}E_l + N_{wurx}\tau_{rx}C_w + N_{tx}E_{tx} + N_{rx}E_{rx} + E_k + \sum_{r=2}^{M} C_r$

Notice that the average cost for each contention is $E_k = E_{tx} + (N-1)E_{rx} + NC_w\tau_{con}/2$

The number of wake ups to transmit is a function of the cumulative cluster offered load G and the duration of the exchange phase. Consequently, I have $N_{wutx} = G(T_{con} - \tau_{con}) = N\mu_{tx}(T_{con} - \tau_{con})$

The number of wake ups to receive can be derived from μ_{rx} with $N_{wurx} = N\mu_{rx}(T_{con} - \tau_{con})$. The number of actual transmissions (and also of receptions since

[1]Notice that also $I_N - T$ is lower triangular and as such quick to invert

I am assuming a very high probability that at least one node is receiving during each transmission) depends on the characteristic throughput S of the chosen MAC scheme. From [50]. I know that:

$S = \frac{Ge^{t_{fly}G}}{e^{t_{fly}G}+G(\tau_{tx}+2t_{fly})}$

Hence, $N_{tx} = N_{rx} = S(T_{con} - \tau_{con})$.

Considering that in the successive exchange phases, the cumulative offered load and throughput remain the same and that the listening cost is a factor only for the CL, I can write the total energy consumption as:

$$En = M(T_{con} - \tau_{con})(GE_l + S(E_{tx} + E_{rx})) + ME_k + (N\mu_{rx}\tau_{rx} + M(T_{con} - \tau_{con}))C_w$$

Since I already determined G and S as a function of τ_{tx}, I can easily find the expression for the expected energy consumption by replacing M using the equation previously derived for the expected convergence time.

6.2.3 Simulations and Validation

To validate the mathematical model and test the robustness of the algorithm against node failures or malfunctions, I implemented a simulation framework using OMNeT++, an object-oriented discrete event network simulator [31].

For the energy costs and packet duration, I considered the typical parameters of the MICA platform [8]. I further implemented a virtual clock drift at each node to test the robustness of EERINA against synchronization problem. The goals of these simulations are to analyze the performance of th EERINA, to validate the mathematical model, and to test the algorithm for

In Figure 6.8, I show the resulting average total energy consumption with respect to the contention periodicity T_{con} and the cumulative normalized offered traffic load

$G\tau_{tx}$[2] for a cluster of 20 nodes. Notice that the energy consumption is a convex function and that the minimum is obtained for $G\tau_{tx} \approx 0.6$, independent of the value of T_{con}. This is important because it allow us to optimize the two variables separately, and, in particular, to choose T_{con} to optimize energy consumption or convergence time, depending on the application context.

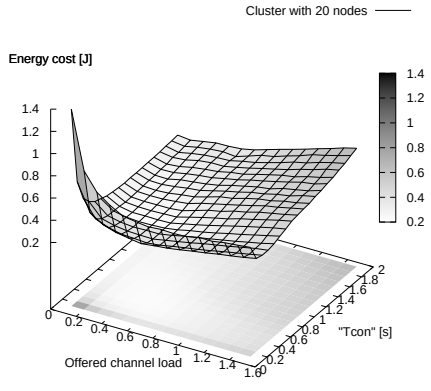

Figure 6.8. Energy minimizations in a cluster of 20 nodes

In Figure 6.9, I show the average energy consumption and convergence time for different number of nodes in the cluster when the parameters are optimized for each of the two metrics.

Notice that the simulated values (solid line) are very close to the values predicted by the mathematical model (dashed). This confirms the validity of the model to predict performance without the need of extensive simulations.

Notice also that convergence time and energy consumption are both linear function of the number of nodes in the cluster. Consequently, EERINA is able to perform like any classical cluster head algorithm [74], while tolerating the death of the cluster leader.

To test the robustness against node death or malfunction, I consider a set of

[2]As I explained in the previous section, $G\tau_{tx}$ is directly prportional to μ_{tx}

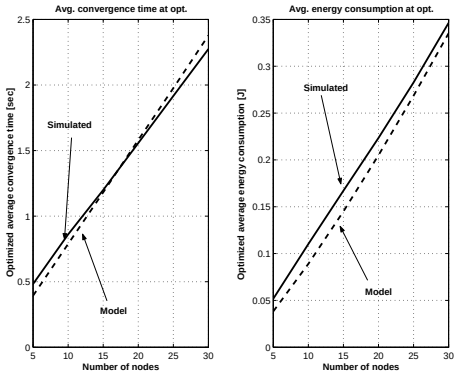

Figure 6.9. Energy consumption and convergence time with optimized parameters. Simulated values in solid, model prediction in dashed

Table 6.3. Mean time to failure performance

MTTF [s]	Average time varition	AVG+2σ Variation (WCET)
5000	0.23 %	0.82 %
1000	0.46 %	2.10 %
500	0.52 %	2.34 %
100	4.11 %	24.3 %
50	6.33 %	26.59 %
10	34.14 %	104.91 %
5	86.87 %	242.33 %

scenarios where each node has an exponentially distributed time to failure, whose mean (MTTF) is reported in the first column of Table 6.3.

Notice that the percentage increase in the average and worst case execution time of an agggregation is significant only for values of MTTF that are comparable with the duration of a round For more realistic MTTF value, such as one minute, EERINA is extremely robust. Although, a MTTF of one minute may seem ivery unlikely for a MICA node, this scenario may happens when nodes are getting close to energy depletion. I believe that is a very important added value of EERINA, since the capability of offering very good network performance even when the nodes are close to their final stage, allows for early fault detection and efficient just-in-time maintenance, without significant degradations of the network functionalities.

Chapter 7

Future Directions

In this dissertation, I presented a novel methodology for system level design of wireless sensor networks. This methodology takes inpiration from the Platofrm Based Design methodology that was already developed for classical embedded systems design and readapts its concept to the design issues typical of the WSN domain.

Specifically, I identified three layers of abstractions: the aplication layer, the communication protocol stack, and the hardware nodes, and defined a platform at each of these layers. It was shown how these platforms are a useful abstraction to define mapping problems that allow to develop an effective system level design methodology for different classes of applications.

I addressed different categories of application. *Static Mapping* applications are those problems where the application is representable by a cyclic control routine and the WSN can be deployed ad hoc to support that application. This is the case of vibration monitoring in manufacturing lines, or temperature monitoring in buildings. The other category I addressed was the one represented by the *Dynamic Mapping* problems. This category includes all the problems that cannot be classified as static mapping , and it is usually populated by those domains in which the traffic and the

types of queries cannot be predicted a priori. This is typical of the scenarios in which the WSN has a continuous exposure to human interactions, as well as those scenarios in which different independent applications have to be mapped on a preexisting network infrastructure. Case studies were presented for both categories that showed how the relative design flow is able to deliver solutions with the required level of reliability as well as support for heterogeneous systems and consistent energy savings. This combination distinguishes the proposed approach from other methodologies that were able to offer either reliability or interoperability with different protocols, but not both feratures at the same time.

Together with the main methodological contribution, there are also two important side contributions:

1. The development of two innovative low power communication protocols that are able to exploit node densities to build reliable systems out of unreliable components. While the first one, called RAND, was developed for uniform topologies, the second one, called SERAN, was devewloped for clustered topologies.

2. The development of robust and reliable data aggregation algorithms for clustered topologies. The first one is an evolution of previous gossip based algorithms, while the second one, called EERINA, is very novel and combines the robustness of decentralized aggregation algorithms with the energy performance of centralized ones.

7.1 Impact

It is in general very difficult to assess the impact of a design methodology, and even more so for WSN because there is no real benchmark in this domain and the adoption of this technology is still very limited. The real answer to this question will

be visible only years from now, when it will be possible to observe if this methodology is used in industry and academia or if it will an important starting point for another succesfull methodology. However, the work presented here already received positive feedbacks:

1. Several papers, whose content represented part of this dissertation, were published in transactions and other international conference procedings. In particular, the paper that presented SERAN [43] was awarded of the Best Application Paper Award at the Mobile Ad-hoc Sensor Systems conference in November 2005 (MASS 05).

2. In the industry, a PBD based methodology (similar to the one outlined in this dissertation), was used to the design and develop the first safety critical certified (class 4 certification) wireless communication system for industrial automation by Comau Inc. Using my system level approach, that project was able to ensure the required level of reliability while coping with a harsh environment and maximizing the reutilization of off-the-shelf components.

3. Several parts of proposals for the 6th and 7th European Framework Program were tailored around the development of aspects presented in this dissertation. In this domain, the European network of eccellence for embedded systems called Hycon has the development of such a methodology as one of its goals.

This combination of academic and industrial interest is to be considered a promising driver for the exploitation of the results presented here.

7.2 Avenues of Future Research

Despite a lot of commited capital, still after almost a decade the penetration of WSN technology in mass markets is very limited. One of the reasons is that, although major research efforts in both academia and corporate research were visible in the last years, still this technology needs further maturation before entering markets with strong real time and reliability requirements. I already discussed how this work is an important step in this direction.

Another important aspect is the capability of creating value for the end user from the interaction of different communication networks. The last decade was characterized by a great effort in the wireless domain to develop communication protocols, standards and hardware platforms to support different classes of applications. On the one hand, there are wireless internet standards like WiFi and WiMax. On the other hand, there are low power standards for indoor communication such as Bluetooth, for wireless sensor networks such as 802.15.4 and ZigBee, and also a growing market for passive radio communication using RFID chips.

Although great progress has been made, still these efforts are mostly bottom up and in general targeted to solve some specific problems. This is typical of an emerging technology, but in the long run it may prevent some key assets of this technology from emerging. For instance, the convergence and integration of different wireless standards has the potential to offer the capability to create a connection between physical phenomena (wireless sensor networks), human operators (wireless internet), products and supply chain management networks (RFID), with a major impact on our lifestyle as well as in the business processes. The middleware presented in Chapter 5 is a step in this direction. This effort should be continued, possibly integrating important development at the physical layer such UWB technology and cognitive radios.

The most promising markets for these new opportunities are:

1. Health care. Offering wireless networking capabilities to the new generation of body sensors represents a huge opportunity to refine the monitoring of senior or ill conditioned citizens as well as assisting them and guiding them during recreational and rehabilitation activities. Other possible applications come from the creation of RFID networks to track the status of medicine in the supply chain and send this information to the collection points such as hospitals or rehabilitation centers. The integration of these two applications is an example of coordination of human interaction, physical sensing and product tracking that can create a new paradigm for health monitoring which is more efficient, cheaper and less invasive.

2. Ambient Intelligence. The ability to integrate user oriented applications such as wireless internet and multimedia, with wireless sensor networks for controlling home features (e.g. light, temperature, gas leakage etc.) will provide a new generation of service oriented home communication systems which will take advantage of the new flexible RF systems such as cognitive radios as well as UWB radios.

3. Industrial automation. In a manufacturing plant, there are tens of thousands of sensors, each of which is cabled to data collection points and the data is then conveyed using Ethernet cables. The maintenance cost of these cables, together with the time required to place them, has a reverse impact on the flexibility and the economics of these plants. Wireless technology will have a huge impact in this domain not only in terms of cost reduction, but also as an enabler of higher plant flexibility as well as new capability of process control and monitoring. Furthermore the integration of RFID networks will provide the necessary cooperation between the supply chain management and

the plant operations with great advantages in the operations and overall business efficiency.

Bibliography

[1] D. Snoonian. Smart buildings. *IEEE Spectrum*, pages 18–23, 2003.

[2] J. Rabaey, E. Arens, C. Federspiel, A. Gadgil, D. Messerschmitt, W. Nazaroff, K. Pister, S. Oren, and P.Varaiya. Smart energy distribution and consumption information technology as an enabling force. *White Paper*, http://citris.berkeley.edu/SmartEnergy/SmartEnergy.html.

[3] G.Huang. Casting the wire. *Technology Review*, pages 50–56, 2003.

[4] R. Zurawski. Introduction to special issue on industrial communication systems. *Proc. of the IEEE*, 93(6):1067–1072, June 2005.

[5] A. Willig, K. Matheus, and A. Wolisz. Wireless technology in industrial networks. *Proc. of the IEEE*, 93(6):1130–1151, June 2005.

[6] A. Willig. Wireless lan technology for the factory floor. In R. Zurawski, editor, *The Embedded Systems Handbook*, Industrial Information Technology Series. CRC Press, Florida, August 2005.

[7] J. Rabaey et al. Picoradios for wirless sensor networks: The next challenge in ultra-low-power design. *Proceedings of ISSCC*, 2002.

[8] D. Culler J. Hill. Mica: A wireless platform for deeply embeded networks. *IEEE Micro.*, 122(6):12–24, 2002.

[9] O. Kasten J. Beutel and M. Ringwald. Btnodes - a distributed platform for sensor nodes. *Proceedings of SenSys 2003*, pages 292–293, 2003.

[10] D. Culler J. Polastre, R. Szewczyk. Telos: Enabling ultra-low power wireless research. *Proceedings of IPSN/SPOTS 2005*, 2005.

[11] John Heidemann Wei Ye and Deborah Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506, 2004.

[12] J. Hill J. Polastre and D. Culler. Versatile low power media access for wireless sensor networks. *Proceedings of SenSys 2003*.

[13] H. Balakrishnan B. Chen, K. Jamieson and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Proceedings of MobiCom 2001*.

[14] M.Zorzi and R.R.Rao. Energy and latency performance of geographic random forwarding for ad hoc and sensor networks. *Proceedings of WCNC 2003*, 2003.

[15] J. Heidemann Y. Xu and D. Estrin. Geography-informed energy conservation for ad hoc routing. *Proceedings of MobiCom 2001*, pages 70–84, 2001.

[16] IEEE 802.15 WPAN Task Group 4 (TG4). http://www.ieee802.org/15/pub/tg4.html.

[17] The Zigbee Alliance. http://www.zigbee.org.

[18] A. Sangiovanni-Vincentelli. Reasoning about the trends and challenges of system level design. *Proceedings of the IEEE*, 95(3), 2007.

[19] A. Ferrari A. Sangiovanni-Vincentelli. System design - traditional concepts and new paradigms. *Proceedings of ICCD 99*, pages 2–12, October 1999.

[20] F. De Bernardinis A. L. Sangiovanni-Vincentelli, L. Carloni and M. Sgroi. Benefits and challenges for platform-based design. *Proceedings of DAC 04*, 2004.

[21] A. Sangiovanni-Vincentelli R. Passerone A. Pinto, A. Bonivento and M.Sgroi. System level design paradigms: Platform-based design and communication synthesis. *ACM Transactions on Design Automation of Electronic Systems*, 2006.

[22] A. Sangiovanni-Vincentelli M. Sgroi, A. Wolisz and J. M. Rabaey. A service-based universal application interface for ad-hoc wireless sensor networks. *U.C.Berkeley, Whitepaper*, 2004.

[23] Dust Inc. http://www.dust-inc.com/.

[24] Crossbow Technologies Inc. http://www.xbow.com/.

[25] Ember Corporation. http://www.ember.com/.

[26] F. Boekhorst. Ambient intelligence: The next paradigm for consumer electronics. *Proceedings IEEE ISSCC 2002*, 2002.

[27] R. Steigman and J. Endresen. Introduction to wisa and wps. *Whitepaper, ABB Inc.*, August 2004.

[28] Intel Inc. Preventive maintenance on an oil tanker in the north sea: The bp experiment. *http://www.intel.com/research/experience/*.

[29] R. von Behren M. Welsh E. Brewer D. Gay, P. Levis and D. Culler. The nesc language: A holistic approach to networked embedded systems. *Proceedings of Programming Language Design and Implementation (PLDI)*, 2003.

[30] M. Weksh P. Levis, N. Lee and D. Culler. Tossim: Accurate and scalable simulation of entire tinyos application. *Sensys*.

[31] A. Varga. Omnet++ discrete event simulation system. In *Proc. of ESM*, Prague, Czech Republic, June 2001. IEEE.

[32] E.A. Lee X. Liu Y. Zhao P. Baldwin, S. Kohli. Visualsense: Visual modeling for wireless and sensor network systems. *UCB ERL Memorandum UCB/ERL M04/8*, 2004.

[33] The Ptolemy Project. http://ptolemy.eecs.berkeley.edu.

[34] Y. Yu, B. Hong, and V.K. Prasanna. Communication models for algorithm design in wireless sensor networks. In *Proc. of APDCM*, Denver, CO, April 2005. IEEE.

[35] A. Bakshi and V.K. Prasanna. Algorithm design and synthesis for wireless sensor networks. In *Proc. of ICPP*, pages 423–430, Montreal, Quebec, Canada, August 2004. IEEE.

[36] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proc. of the Sixth Ann. Intl. Conf. on Mobile Computing and Networks (MobiCom 2000)*, pages 56–67, Boston, Massachussets, August 2000.

[37] S. Madden. *The Design and Evaluation of a Query processing Architecture for Sensor Networks*. PhD thesis, University of California, Berkeley, 2003.

[38] J. Polastre et al. A unified link abstraction for wireless sensor networks. *Sensys*.

[39] IEEE 1452.2. Standard for a smart transducer interface for sensors and actuators - transducer to microprocessor communication protocols and transducer electronic data sheet (teds) formats. *IEEE*.

[40] James A. Rowson and Alberto L. Sangiovanni-Vincentelli. Interface-based design. In *Proceedings of the* 34^{th} *Design Automation Conference, DAC 1997*, pages 178–183, June 1997.

[41] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vincentelli. Addressing the system-on-a-chip interconnect woes through communication-based design. In *Design Automation Conference, DAC '01*, June 2001.

[42] R. Passerone. *Semantic Foundations for Heterogeneous Systems*. PhD thesis, University of California, Berkeley, 2004.

[43] A. Bonivento, C. Fischione, A. Sangiovanni-Vincentelli, F. Graziosi, and F. Santucci. Seran: A semi random protocol solution for clustered wireless sensor networks. In *Proc. of MASS*, Washington D.C., November 2005.

[44] A. Bonivento, C. Fischione, and A. Sangiovanni-Vincentelli. Randomized protocol stack for ubiquitous networks in indoor environment. In *Proceedings of CCNC*, Las Vegas, NV, January 2006.

[45] T. Tong A. Woo and D. Culler. Taming the underlying challenges of reliable multhop routing in sensor networks. In *Proceedings of Sensys*.

[46] A. Bonivento J. Rabaey K. Ramchandran A. Sangiovanni-Vincentelli J. Van Greuen, D. Petrović. Adaptive sleep discipline for energy conservation and robustness in dense sensor networks. In *Proceedings of ICC*.

[47] D. Petrović E. Lin J. Van Greuen J. Rabaey R.C. Shah, A. Bonivento. Joint optimization of a protocol stack for sensor networks. In *Proceedings of Milcom*.

[48] A. Willig A. Kopke and H. Karl. Chaotic maps as parsimonious bit error models of wireless channel. In *Proceedings of Infocom*.

[49] R. Scopigno A. Bonivento R. Calcagno F. Rusina' D.Brevi, D. mazzocchi. A methodology for the analysis of 802.11a links in industrial environments. In *Workshop on Factory Communication Systems (WFCS)*.

[50] T.S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, Upper Saddle River, NJ, 2001.

[51] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[52] R. Govindan C. Intanagonwiwat, D. Estrin and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems, Vienna, Austria*.

[53] K. Ramchandran D. Petrović, R.C. Shah and J. Rabaey. Data funneling: Routing with aggregation and compression for wireless sensor networks. In *Proceedings of SNPA 03*.

[54] A. Bonivento, L.P. Carloni, and A. Sangiovanni-Vincentelli. Rialto: a bridge between description and implementation of control algorithms for wireless sensor networks. In *Proc. of EMSOFT*, Jersey City, NJ, USA, September 2005.

[55] G. Kahn. The semantics of a simple language for parallel programming. In *Proceedings of the IFIP Congress 74, North-Holland Pub*.

[56] D. B. MacQueen G. Kahnand. Coroutines and networks of parallel processes. In *Proceedings of Information Processing 77*.

[57] E. A. Lee and A. Sangiovanni-Vincentelli. A framework for comparing models of computation. *IEEE Transactions on CAD*, 17, 1998.

[58] J. Misra. Distributed discrete-event simulation. *ACM Computing Surveys*, 18(1):39–65.

[59] D. Culler, D. Estrin, and M. Srivastava. Overview of sensor networks. *IEEE Computer*, 37(8):41–49, August 2004.

[60] Felice Balarin et al. Concurrent execution semantics and sequential simulation algorithms for the metropolis meta-model. In *Proceedings of the Tenth International Symposium on Hardware/Software Codesign*, Estes Park, CO, May 2002.

[61] Felice Balarin, Luciano Lavagno, Claudio Passerone, Alberto L. Sangiovanni-Vincentelli, Marco Sgroi, and Yosinori Watanabe. Modeling and designing heterogeneous systems. In *Concurrency and Hardware Design, Advances in Petri Nets*, pages 228–273, London, UK, 2002. Springer-Verlag.

[62] L. Girod J. Elson and Deborah Estrin. Fine-grained network time synchronization using reference broadcast. In *Proceedings of the 5-th Symposium on Operating Systems Design and Implementation*.

[63] G. L. Pierobon D. Miorandi, A. Zanella. Performance evaluation of bluetooth polling schemes: An analytical approach. *MONET*, 9(1):63–72.

[64] G. Bianchi. Performance analysis of ieee 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Com- munications*, 18(3):535–547, 2000.

[65] L. Lavagno L. Vanzago A. Sangiovanni-Vincentelli L. Necchi, A. Bonivento. Eerina: an energy efficient and reliable in-network aggregation for clustered wireless sensor networks. In *Proceedings of WCNC*.

[66] D. Liu and M. Prabhakaran. On randomized broadcasting and gossiping in radio networks. In *Proc. of COCOON*, pages 340–349, Singapore, August 2002.

[67] J. Luo, P.T. Eugster, and J.P. Hubaux. Route driven gossip: Probabilistic reliable multicast in ad hoc networks. In *Proc. of INFOCOM*, pages 1–12, San Franciso, CA, April 2003. IEEE.

[68] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. of FOCS*, pages 482–491, Cambridge, MA, October 2003. IEEE.

[69] G. Pandurangan J. Chen and D. Xu. Robust aggregates computation in wireless sensor networks: Distributed randomized algorithms and analysis. In *Proc. of IPSN*, April 2005.

[70] P. Buonadonna, D. Gay, J. Hellerstain, W. Hong, and S. Madden. Task: Sensor network in a box. *Intel Research Lab Report*, January 2005.

[71] J. Kahn, R. Katz, and K. Pister. Next century challenges: Mobile networking for smart dust. In *Proc. of MobiCom*, pages 271–278, Seattle, WA, August 1999.

[72] B. Hohlt, L. Doherty, and E. Brewer. Flexible power scheduling for sensor networks. In *Proc. of IPSN*, Berkeley, CA, April 2004. IEEE.

[73] D. Kempe and J. Kleinberg. Protocols and impossibility results for gossip-based communication mechanism. In *Proc. of FOCS*, pages 471–480, Vancouver, Canada, November 2002. IEEE.

[74] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Trans. on Wireless Coomunications*, 1(4):660–670, October 2002.

[75] H.M. Taylor and S. Karlin. *An introduction to Stochastic Modeling.* Third Edition, Academic Press, 1998.