Università degli Studi di Pisa

Dipartimento di Informatica
Dottorato di Ricerca in Informatica

Ph.D. Thesis

Dealing with Non-Uniformity in Wireless Sensor Networks

Francesco Nidito

Supervisor
Prof. Susanna Pelagatti

Worthy of my undying regard

Abstract

In this thesis, we step inside an unexplored region of Wireless Sensor Networks (WSNs) research. Nowadays, almost all WSNs research relies upon a hidden uniformity assumption. This assumption involves deployment, distribution and radio transmissions. Unfortunately, the real world is not uniform. In the thesis, we break the uniformity assumption and study the non-uniformity influence in WSNs. In particular, we show that addressing common WSN problems taking non-uniformity into account can provide results that are sensibly different from the ones achieved in a uniform world. In our work, we focus on the influence of non-uniformity on a particular aspect of WSNs: data management. First of all, we point out that even widely accepted solutions based on the uniformity assumption are not able to survive inside an non-uniform world. Then, we propose our approach to data management and detail a solution able to deal successfully with non-uniformity. This allows us to catch out the fundamental aspects of non-uniformity influence in WSNs and to cope with non-uniformity. Results, discussed in the thesis, show that models and solutions we propose are competitive in a uniform scenario and continue to work properly in a non-uniform world.

Acknowledgments

Probably, acknowledgments are the most difficult part of a Ph.D. Thesis to be written: I always fear to forget someone. Indeed, I do not fear to forget someone *generic*, I fear to forget someone *vindicative*... by the way, I will try to acknowledge everyone.
First of all, I want to thanks all my girlfriend Chiara, for her infinite patience, support and love during the all Ph.D. years and (especially) during the writing of this Thesis.
A very special thanks goes to my supervisor Susanna Pelagatti, she is more than my supervisor, she is a great friend. She supported me during my Thesis and listen to my strange ideas without throwing me throughout the window of her office... also when throwing me throughout the window could seem the only logical reply to my statements.
Then, I want to thanks my family and my girlfriend's family for everything they did to support and encourage me during the years of study.
I want to thanks my commission: Stefano Chessa, Fabrizio Luccio and Paolo Santi. Moreover, I want to thanks the reviewers of this thesis: Elias P. Duarte Jr. and Jose Rolim.
I would like to thanks my historic friends since the first days of University: Carlo Bertolli, Pietro Bini, Patrizio Dazzi, Simone Montaresi, Stefano Pacifici, Daniele Picciaia, Luca Pizziniaco, Luca Saiu and Andrea Venturi.
I want to thanks my officemates at the department, Gianni Franceschini and Claudio Scordino, for the great time together (and silence when needed). Moreover, I want to thanks all the Ph.D. students that lived with me this extraordinary experience.
Another special thanks goes to Prof. Giuseppe Attardi who gave me the opportunity to be his teaching assistant for the class of "Programmazione Avanzata" (Advanced programming) where I learned more things than I taught.
I cannot forget all the friends of my adventure/research experience in Boston: Stefano Basagni, Michele Battelli, Luca Caracoglia, Alessio Carosi, Paolo Casari, Elisa Dell'Oglio, Rituparna Ghosh and Michele Nati.
Finally, I want to thanks all the fiends and colleagues at the Ask.com R&D office in Pisa).
A last note to everyone I acknowledged here (and for the ones that I forgot): the dedication of this thesis "Worthy of my undying regard"* is for you.

* That is the same that Joseph Conrad used in "The Shadow Line".

Contents

List of Figures

Introduction

Vi veri veniversum vivus vici
- Dr. Johan Faust

The objective of this introduction is to provide a guideline for the readers of this thesis. Our work in the network research field was originated by the author's interest, and love, for networks in general. This love is depicted in Section I.1, where we point out the networks represent the hidden structure of the world we live in. Then, in Section I.2, we unveil our development plot for this thesis: we describe what we are intended to show with the thesis, pointing out what we like and what we do not like in the current wireless sensors networks research and how we intend to react to this. Finally in Section I.3, we provide a brief outline of the thesis' structure.

I.1 Linked: how networks changed our perception of the world

Networks are the hidden structure of our world. They play a fundamental role in natural, social and technological sciences. Current research works in all these fields have spotted out the common network structure of the world.

Moreover, recent studies try to find out common properties of all kinds of networks to archive a better understanding of our world and life. These works, such as (Dorogovtsev and Mendes, 2003), (Watts, 1999) and (Watts, 2003), point out that networks can be found in any shape and size in the world around us. These works also study the problem of finding out the common structures and properties of both natural and artificial networks. These studies point out that all the networks (natural and artificial) have common structures and patterns inside them. Paradoxically, these outstanding results tend to show that the world, as we know it, does not contain network structures but, as opposite, the world appears as it is because it is generated by these (not so much) hidden network structures.

In this section, we want to follow a (short) path that will review the networks starting from natural networks, running throughout human networks and finally, coming to computer networks. Moreover, we will point out the natural *wireless shift*: in nature, networks tends to move wireless to be more efficient and to grow up at a higher rate.

I.1.1 Natural networks

Recent studies in various aspects of biology have shown how networks are a constant structure in biological systems.

The preys/predators structure in a ecosystem is the simplest network-shaped system that we may find in biology (Lässig et al., 2001): various species are linked together in an eaten-eater graph structure. This graph structure links with few hops species that at first sight can be considered very distant, for instance, a flower and a tiger.

Another very interesting case of network-like structure that we can find in nature is represented by the south-eastern Asia fireflies visual communication network (Buck, 1988). These little beings synchronize their lights looking at their neighbors so that large groups are able to flash simultaneously. Recent hypothesis about the meaning of this phenomenon are related to the fact that this synchronized flashing may help firefly males to attract distant females and/or to create a natural defense because a predator is confused by the number of possible preys, but this reasons are out of the scopes of this thesis and, a more important fact, out of the scope of this introduction.

I.1.2 Human networks

Sociologists sentence that the success of mankind is related to our feeling for communications (Lieberman, 1998). In sociology's terms, the *success* is identified with the capacity of humans to organize in large societies as nations, which is not common in other species.

Since ancient times, we, as humans, have the need to communicate our experience and stories. To easily prove this, we can think to cave paintings made by primitive men, that were used to communicate *throughout time*, generation by generation, the experience and believes of the population they belonged to. Today, as in the ancient times, we continuously leave signs of our presence writing books, building monuments and modifying the face of the world in which we live. As an extreme case of this kind of communications, in 1977, we were not fully satisfied by communication *throughout time* and we decided to communicate also *throughout space*: we sent the "Voyager Golden Record" to communicate sounds and images of the Earth to intelligent extraterrestrial life forms that the Voyager spacecraft may encounter in the space.

Apart from this, the principal network structure of our society is still language: our need of communication brought to complex languages with thousands of terms and structural forms to join them and being able to establish complex social relationships.

The need for communication of the human race was, and it is still, the most important engine that brought mankind to invent forms of networks that were not

present in nature, for instance, the postal system and telephones. As a consequence, our world became smaller and smaller and our society highly connected (Milgram, 1967).

John Donne, a Jacobean metaphysical poet, said that "No man is an Iland, intire of it selfe; every man is a peece of the Continent, a part of the maine" (Meditation $XVII$). In our opinion, John Donne was probably right.

I.1.3 Computer networks

Computer networks try to answer to the natural need for communications. As well as the postal service and other human-related communication networks, computer networks are a powerful communication network characterized by an incredible high speed and a high number of users[†].

Wired computer networks were born in the middle of the 20^{th} century to respond to military need for a *reliable* and *fault-tolerant* communication infrastructure. Computer networks rapidly evolved enabling people to exchange pieces of information at rates never seen before. However, in the past twenty years there has been a growing need for ubiquitous communications (anytime, everywhere) (Kleinrock, 1996). This need cannot be satisfied by wired media and brought new interest to wireless communications. Satellites, cellular and radio communications, that were used only to communicate voice or broadcast TV, are now used to enable people to exchange data from everywhere.

As a consequence of this *unwiring revolution*, current hosts, lighter and more powerful, can move around and connect to the web or network resources from almost everywhere.

I.1.4 Everyone goes wireless

Every kind of communication seems to move "naturally" to wireless.

The south-eastern Asia fireflies use a kind of *optical* network to coordinate and to synchronize: each firefly looks at other fireflies flashing and starts to synchronize with them.

Humans at great distance, once communicating with letters, now prefers to phone each other. The spoken word prevails in interpersonal communications: it is more expressive due to intonations[‡].

Today communication networks are experiencing the same unwiring revolution: our traditional telecommunication systems are dropping the heavy cables moving to wireless connections due to their flexibility because we want to be able to communicate each other from everywhere.

[†] As of June 10, 2007, 1.133 billion people use the Internet.

[‡] In the same way IP phones, for instance Skype, are eclipsing e-mail

I.2 The plot behind this thesis

The first thing we do, let's kill all the lawyers
- William Shakespeare (Henry VI)

In this section, we present the plot behind our thesis. We present where we are, what we want to study and, most important, why we want to study it. In this thesis, we study the non-uniformity issues in wireless sensor networks: that is how the non-uniformity influences the behavior of this kind of networks and how we can prevent (or use) this influence. Non-uniformity can influence all the aspects of wireless sensor networks, for this reason, in this thesis, we focus on the problem of the influence of non-uniformity in data management.

I.2.1 Wireless sensor networks

We focus on one particular kind of wireless network: the *wireless sensor networks* (Akyildiz et al., 2002a). Wireless sensor networks (WSNs for short) are made up by a large number of little devices that sense the environment and exchange data with wireless media. They represent a natural evolution of monitoring and sensing systems: wireless sensor networks' possible applications range from military to medic, from domotics to industry and precision agriculture.

Wireless sensor networks can be used also to provide an easily deployable communication infrastructure in areas where there was some kind of natural disaster. In this case, sensors form a backbone for the routing of messages between groups of rescuers.

I.2.2 Non-uniformity

In this thesis, we study a particular aspect of the WSNs, namely the influence of *non-uniformity* in such kind of networks.

Non-uniformity is a topic that is far from the mainstream of WSNs research. In current WSNs' research, there is a silent axiom that rule out the models and the protocols development: the uniformity of these kind of networks. We refer to this assumption with the name *uniformity dogma*.

We intend to show up that the *uniformity dogma* has no foundations and we intend to point out that non-uniformity exists in nature, that non-uniformity can influence WSNs and that non-uniformity can prevent the good behavior of the protocols that were designed with only uniformity in mind.

Uniformity is assumed in various aspects of WSNs. It is assumed in the deployment when nodes are randomly scattered in the area intended to be monitored, also when deployment occurrs by dropping nodes from above. It is assumed in communication

patterns and technologies, also when the nature itself of radio communications brings to unstable links from both bandwidth and connectivity point of view. Finally, it is assumed into the environment, also when the deployment can happen into woods or even cities where obstacles can prevent close nodes to communicate properly.

I.2.3 Data management in wireless sensor networks

Due to the fact that non-uniformity can influence the largest part of current WSNs solutions, we choose to focus, as a *case study*, on the influence of non-uniformity in data management.

We will start from the influence of non-uniformity in a widely accepted data management solution, then, we will find out the *non-uniformity soft spots* in the design of that solution and then we will propose our *non-uniformity-proof* solution: pointing out the design choices that we must take to produce *non-uniformity-proof* solutions.

I.3 Organization of the thesis

This thesis is organized as follows.

Chapter 1 provides an introduction to WSNs. We point out that WSNs are a recent technology designed for unattended, remote monitoring and control, which have been successfully employed in several applications. WSNs are designed to perform environmental data sampling and processing, and to guarantee access of the processed data to remote users. Moreover, we point out that in traditional WSN these tasks consist in transmitting sensed data to a powerful node (the sink) which performs data analysis and storage.

Chapter 2 points out how most of the current WSNs research focus on uniform networks and more specifically on networks in which the sensors are all identical or on networks that are distributed following the uniform distribution. Then, we show that in current WSNs research uniformity is a dogma, a believed true assumption that no one wants to offend. At this point, we analyze real life and show that uniformity does not actually exist in nature. Starting from this consideration, we can only begin to study non-uniformity, its fundamental characteristics, how non-uniformity influences WSNs results and how we must learn to think in a non-uniform way to prevent its influence.

Chapter 3 provides the description of the state of the art of the data management in WSNs. All the reasonable uses of WSNs deal with the idea of data acquisition and data retrieval. These two concepts are strictly related because data retrieval is the response to a user query. The user should be able to

actively program the network, via control programs, to retrieve data that is considered useful. In traditional WSN models, these tasks consist in transmitting sensed data to a powerful node (the sink) which performs data analysis and storage. However, these models resulted unsuitable to keep the pace with technological advances which granted to WSNs significant (although still limited) processing and storage capabilities. For this reason, recent paradigms for WSNs introduced database approaches to define the tasks of data sampling and processing, and the concept of data-centric storage for efficient data access.

Chapter 4 presents our contribution to non-uniformity in the data management research in WSNs. We focus on the the data-centric storage model and our main contribution in this area is Q-NiGHT. Q-NiGHT originated by the analysis of the experiments we performed on the Geographic Hash Tables (GHT) approach in non-uniform networks. Results of such experiments point out the inability of plain GHT to provide good results in non-uniformly distributed WSNs. Moreover, we provide an application scenario in which Q-NiGHT is used as a building block for a more complex system that enables the networks to find out special nodes that are able to provide services to the other nodes.

Chapter 5 proposes the Stripes suite of protocols, that are used to find out in a efficient way the network distribution. The knowledge of the network distribution is a central point in many protocols that are intended to be non-uniformity aware. The Stripes suite of protocols uses an on-demand strategy to rebuild the network density. When a node needs to know the density of the network, asks for density samplings from predefined points of the network and then reconstructs the density with a simple algorithm. To optimize the resources usage of the network, Stripes uses a caching strategy to store the sampled data around the network and not only in the sampling points and into the nodes that reqired such information.

Conclusions draws the conclusions of the work that was presented in this thesis.

I.4 The pillars of this thesis

This thesis is based on various papers by the author of this thesis and some of his colleagues (and friends).

In (Chessa et al., 2007), we previously revised, described and analyzed the data management techniques that are used in WSNs.

In (Albano et al., 2006a), (Albano et al., 2006b) and (Albano et al., 2007), we presented Q-NiGHT. We analyzed the problems related to non-uniformity in data management and how non-uniformity can prevent current systems to work properly.

Then, we proposed our solution to this problem providing a *modus operandi* for the non-uniformity management in WSNs.

In (Nidito et al., 2007), we presented an application scenario that takes great benefit of the Q-NiGHT system to provide a reliable and load balanced system to locate services inside heterogeneous WSNs.

In (Nidito and Pizziniaco, 2006), we studied the problem of non-uniformity from a new and higher point of view. We studied the problem of finding out the number of sensors that is needed to acquire connectivity in a WSN deployed using a non-uniform distribution.

Chapter 1

Wireless Sensor Networks

Then Jesus asked him, "What is your name?"
"My name is Legion," he replied, "for we are many."
- Mark 5:9

Abstract

Wireless sensor networks (WSNs) are a recent technology designed for unattended, remote monitoring and control, which have been successfully employed in several applications. WSNs perform environmental data sampling and processing, and guarantee access of the processed data to remote users. In traditional WSN models these tasks consist in transmitting sensed data to a powerful node (the sink) which performs data analysis and storage.

A Wireless Sensor Network (WSN) is a computer network formed by a large number of little and inexpensive wireless devices (the *sensors* or *sensor nodes*) that cooperate to monitor the environment using transducers (Zhao and Guibas, 2004) (Al-Karaki, 2004) (Akkaya and Younis, 2005) (Akyildiz et al., 2002b) (Akyildiz et al., 2002c). Recent technology advances have enabled the design and the development of tiny processors and radio systems that can be easily embedded in little multi-purpose and easily programmable sensors. A sensor is a micro-system which also comprise one or more sensing units (transducers), a radio transceiver and an embedded battery. Sensors are spread in an environment (the *sensor field*) without any predetermined infrastructure and cooperate to execute common monitoring tasks which usually consist in sensing environmental data from the surrounding environment. Due to low cost, sensors have poor reliability and are subject to failures and battery exhaustion. Sensors are typically deployed in harsh environments where the nodes' substitution can be impracticable. Due to these reasons protocols and sensors' applications must be highly fault-tolerant and the network must be able to self adjust to fast configuration changes.

WSNs can be employed in a large variety of tasks. They can be used in medicine, agriculture, military, inventory monitoring, intrusion detection and many other fields. In the medical field, they can be used to remotely monitor patients' conditions in a non intrusive way: this enables the patients to move in the medical facility while their life parameters are still monitored (Malan et al., 2004, Gao et al., 2005, Amato et al., 2005b). In agriculture, sensors can be used to enable the so called *precision farming*, in which the fields' conditions are constantly monitored to tune the water or fertilizing quantities to maximize the production. Pollution monitoring can be enhanced by such little sensors. The presence of a large number of non intrusive sensors enables a fine grain monitoring of the environment. Such a fine grain monitoring can be essential in the location of pollution sources. The same sensor systems can be used to monitor the environment to prevent flooding, fire or other natural disasters (Steere et al., 2000). A more recent application of the wireless sensor networks is the monitoring of animals in a non intrusive way, due the reduced size of the nodes (Szewczyk et al., 2004, Cerpa et al., 2001, Wang et al., 2003). Other applications range from home automation to education (Srivastava et al., 2001) to inventory monitoring and machinery status monitoring.

The WSN advantage is in the capability of reporting data every time from everywhere in the sensor field. One of the main issues in sensor networks is how to organize data management and retrieval in a reliable and efficient way. In early sensor networks, data collection was performed by human operators who had to physically reach specific positions in the deployment area to acquire data from the environment. This manual operation has three main drawbacks:

1. The data collecting operation was very expensive because human operators had to manually collect data.

2. The operation was error prone due to poor automation.

3. The operation could be risky because of environmental hostile conditions (radiations, poison etc.).

The typical sensors' deployment in an environment consists of hundreds of sensors. The deployment can be both random or predetermined by the user of the network. After the deployment, the sensors self-organize to form a multi-hop network to enable communications between nodes that lie out of the communication range of each other. For instance, in structures monitoring, sensors can be deployed on both static structures, as bridges and building, or dynamic structures, as airplanes or cars. Sensors continuously monitor these equipments ensuring their reliability. In particular, they sense the current status of the system and help to forecast the future evolution of that equipment (Lin et al., 2003).

The user can query a WSN using one or more special purpose nodes called *sinks*. The network can be queried using different paradigms. In traditional networks, sen-

sors collect data and send them to the sink without processing. More recent approaches, however, use the whole network as a database enabling the user to perform complex queries and in-network computation.

Chapter organization. This chapter is organized as follows. Section 1.1 presents the description of WSNs architecture, standard applications and technologies. Section 1.2 presents the abstract model of the WSNs used for our research work. Section 1.3 reviews briefly the research issues in WSNs research. Finally, section 1.4 draws the conclusions of the work described in this chapter.

1.1 Applications, technology and architecture

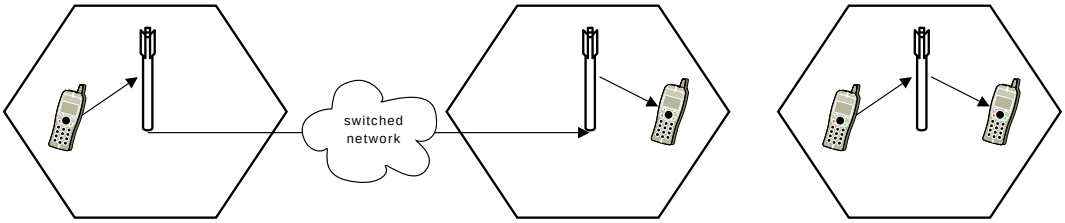
In this section, we review the main characteristics that identify WSNs: architecture, applications and technology. We move throughout these three characteristics to find out the distinctive traits of this kind of networks.

1.1.1 Infrastructured vs. ad hoc wireless networks

Wireless networks can be divided in two main categories: the *infrastructured* and the *ad hoc* networks.

Infrastructured networks. Infrastructured networks have one or more central coordination points. A good example of such networks are cellular networks (Rahnema, 1993). To communicate to another peer, each cellular device needs to negotiate with the infrastructure hardware of the cell it belongs to. Once this communication has took place the infrastructure takes care of setting up a channel between the peers (and to manage accounting, related services etc.). It seems a paradox, but all the communications in an infrastructured network must use the infrastructure. In cellular networks the communication between two peers must pass throughout the network infrastructure also if the two devices could be able to communicate to each other because they are in the transmission range of each other. As depicted in Figure 1.1.a, two peers belonging to two different cells need to communicate to the respective cell manager to set up a communication. The same happens in the case depicted in Figure 1.1.b, in which the two peers would be able to communicate directly but need to use the infrastructure.

Ad hoc networks. *Ad hoc* networks are totally self-organizing and usually do not rely upon any centralized authority. A distinguished feature of ad hoc networks is being able to organize themselves to perform any activity (e.g. routing, balanced data storage, etc.) without a centralization point. All the peers of an ad hoc network must perform two different roles: (*i*) the role of an active communication peer (as sender or receiver) and (*ii*) the role of passive communication peer, performing



(a) Communication between different cells

(b) Communication inside the same cell

Figure 1.1: Peers' communications in infrastructured networks. (a) depicts the communication between two peers located into two different cells using the infrastructure to communicate and (b) two peers inside the same cell that need to uses the infrastructure too.

routing for the other peers' communications. In this sense, ad hoc networks are *collaborative* networks in which all the nodes need all the others to be able to set up communications.

A particular kind of ad hoc networks is given by WSNs. WSNs (Akyildiz et al., 2002a) are formed by hundreds, or even thousands, of simple nodes that are deployed to monitor areas of interest. Each node is able to acquire data from the environment, store the sensed data, possibly after little processing, and coordinate to route the acquired data to one or more special nodes, namely the *sink* nodes. Usually the sink nodes store and use data in different ways depending on the user's needs. Due to their strict constraints in CPU performance, memory on board and battery lifetime, WSNs represent an "extreme" case of ad hoc networks because all the problems related to ad hoc networks are amplified by such strict constraints.

1.1.2 Applications of wireless sensor networks

Due to the nature of self-organization, the WSNs have found a lot of applications in all the fields that need to collect data in environments in which a predefined infrastructure cannot be built up for different reasons: due to time and/or space constraints, or more simply, the user does not want to build such infrastructure.

Military. Military applications are the simplest to imagine. WSNs can be set up to monitor the battlefield or to create intrusion detection systems. In a similar way wireless sensor networks can find an application in the monitoring of ammunition or to create an easily deployable communication infrastructure in which each soldier is equipped with a sensor. This sensor can be used to communicate with other soldiers and to monitor soldiers' health status.

Medical. WSNs can be used also for medical purposes. We can think to employ them to monitor patients when they move inside the hospital or to locate medical stuff, for instance a moving x-ray machine, inside the same structure.

Environmental. Another application is represented by environmental monitoring and control. In this scenario, sensors can be attached to objects, for instance industrial robots, to control their status and position and to send commands to them. In this way, we can also use sensors to locate some furniture or devices quickly, for instance monitors or projectors, in an office.

Domotics. In domotics, WSNs can be used as the underlying architecture to provide all the features required by domotics applications. For instance, sensors can be used inside rooms to monitor the presence of people to automatically turn on and off lights.

Assisted learning. In a possible assisted learning application, people are free to move inside a museum and when they arrive close to an object of the museum collection, an automatized system can start to describe the object. Moreover, the sensors can synchronize themselves with other sensors-like devices that visitors bring with them to provide a multilingual description of the object.

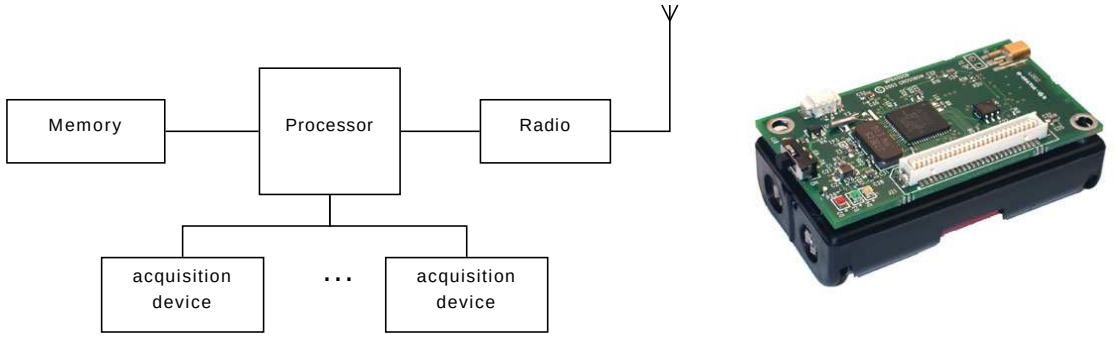
Disasters. Finally WSNs can be used in disaster areas with two main applications: the first is to create an easily deployable communication system and the second one is to monitor, in real time, the evolution of a dangerous environment.

1.1.3 Technology constraints of wireless sensor networks

In most applications, sensors have to be *small* and *cheap*. They need to be small because during their usage they must be "invisible" (military surveillance) or because patients in a hospital cannot be expected to carry a workstation on their back to be monitored. They also need to be cheap because in military or disaster monitoring sensors are dropped in areas where, probably, no one wants to enter. When a sensor breaks down, or its battery is over, no one will repair it and the device is gone forever.

Due to their nature of *small* and *cheap* devices, sensors have very limited resources. Current sensors are equipped with a small battery, a CPU with very low computational power (MICA-MOTEs from Crossbow Inc.* have an $8MHz$ processors with 8 bits words), memory limited to few hundreds of kilobytes, a wireless communicator (radio or IR) and some acquisition devices (for temperature, pressure, etc.). Figure 1.2.a depicts the standard architecture for a sensor node. The architecture is very simple and made up few components as said before. Figure 1.2.b depicts a real sensor. To give a scale factor for such devices its measures are $58 \times 32 \times 7$ millimeters and it weights 18 grams (excluding batteries).

* www.xbow.com



(a) Sensor architecture (b) A MICA-MOTE sensor node

Figure 1.2: Sensor node. (a) depicts the typical architecture of a sensor, made up of a processor, memory, some acquisition devices and a radio. (b) depicts the photo of a real sensor node.

Like in other wireless networks, in WSNs, packet collisions due to multiple communication taking place in the same space at the same time is more difficult to handle than in wired networks. In traditional wired networks, when an host sends a packet, the host is able to receive its own signal on the on the same cable. In this way, it is easy to notice interferences due to packet collisions. On the other hand, in wireless networks, we can experience the so called *hidden terminal* problem (or *hidden host* problem). As depicted in Figure 1.3, when two hosts (that are not in communication range of each other) send a packet at the same time to a common receiver, it will receive only a *junk transmission*. However, none of the senders is able to detect the collision without help. The hidden host problem is solved using MAC level protocols which exchange a few control messages to rule out all the possible collisions before starting a real sending operation (Fullmer and Garcia-Luna-Aceves, 1997).

A similar problem arises when a large number of devices try to communicate in the same area. In this case, the communications collide frequently and the problem known as *channel contention* may rise. In the channel contention problem many nodes share the same subarea and all the nodes are able to communicate each other. When two or more nodes try to communicate at the same time (also to different receivers) their requests to establish a communication clash and the nodes try again to send the requests. This problem has a domino effect on the sensor nodes: at the beginning it limits bandwidth because packets need to be transmitted again and then the retransmissions' cost is payed by the sensor via battery exhaustion. Again we can use suitable MAC protocol strategies to orchestrate the channel access in order to avoid collisions. Typical techniques used to avoid such clashes take care of waiting a random time before trying again to communicate. If the number of nodes trying to communicate is high the time range in which they randomize that time can be too

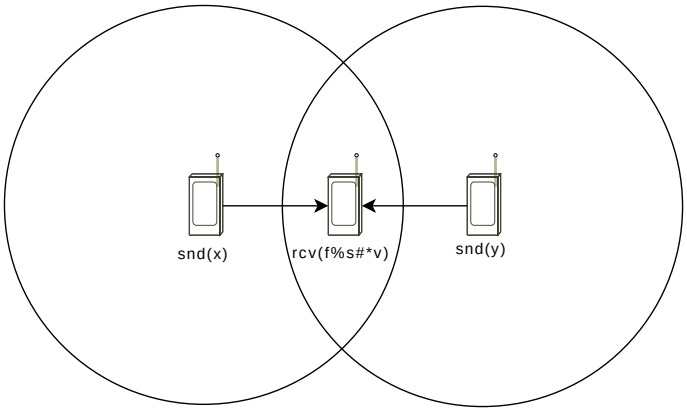


Figure 1.3: Hidden host problem in which the two transmitting nodes are unaware of the presence of each other and simultaneously transmit to the cental peer that is unable to receive correctly the two messages.

short and new clashes happen again. However, a collision free communication forces nodes to communicate one at the time drastically reducing the potential bandwidth of the network.

1.1.4 Wireless sensor networks' architecture.

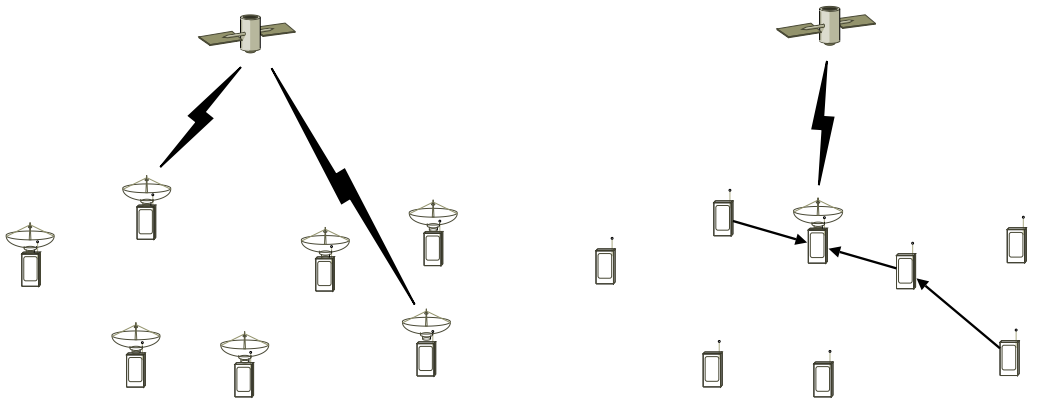
Each sensor, *per se*, is a too powerless device to be useful if used alone[†]. Actual WSNs are made up of hundreds (or maybe thousands) of sensors. All sensors *cooperate* to provide sensing, communication and reliability to all the system. More specifically, *cooperation* is used to achieve multiple results.

For instance, if each sensor is able to measure the temperature in less than $1m$ of distance from itself, thousand of nodes can be used to monitor the temperature of whole buildings of even cities. In the same way, the transmitting range of a single sensor ranges from $5m$ up to $30m$ but a WSN can route messages, hop by hop, for tenths of kilometers if needed.

We stated that each sensor is very unreliable and short living, but a dense network can overcome the problem of sensors ceasing to work simply with the number. If inside the network, each sensor is *anonymous*, in the sense that it can be replaced by any other in its neighborhood.

Architecture. As for any other *technological device*, the task to be accomplished defines the shape (or better, the architecture) of such *device*. WSNs are no exception to this rule. Since the main WSNs' applications are driven by the need to acquire, collect and communicate data, the WSNs must, at some point, communicate to the

[†] By the way, the first computer owned by the author of this thesis was as powerful as a single sensor, nevertheless the author had a lot of fun with it.



(a) WSN without sinks (b) WSN using a sink

Figure 1.4: WSNs architectures. (a) depicts a sensor network without a sink node and where each node is able to communicate to the user (in this case using a satellite up-link). (b) depicts a sensor network with a sink node that collect the data from the sensors, via multi-hop communication, and then it sends data to the user using a satellite up-link.

external world the acquired data. This can happen in two ways: either every node can communicate to the user or only few nodes can do that.

In the first case (Figure 1.4.a), we can imagine that all sensors have enough power to communicate out of the area of interest (e.g., using a satellite up-link) or, some times, the user can enter the area and use a sensor-like device, for instance attached to a laptop, to query the devices.

In the second case (Figure 1.4.b), we have one or more special nodes, called *sinks*, that collect and communicate data to the user. Sink nodes are special nodes, different from other sensors, with more computational power and battery. They can be always active, active only at pre-established times or active at request by the user. Sinks are *gateways* between the users of WSNs and the network itself. The kind of interconnection between the sink and the outside world is not part of the WSN and it can take place in a large number of forms.

Inside WSNs, the communications can take place in only two ways: single-hop and multi-hop. In single-hop communications, data are exchanged directly from the sender node to the receiver node. In multi-hop communications, one or more intermediate nodes route communications between senders and receivers not in transmission range.

Deployment. Roughly speaking, sensors can be deployed only in two ways[‡]: (*i*) "by hand" or (*ii*) randomly.

In the deployment "by hand", sensors are placed one by one in specific locations to create networks with desired characteristics, for instance sensors can be deployed to form a *mesh* or *tori* or, more simply, the sensors are placed to be aware of their position, of the position of the other sensors and of the position of the sink. This placement can be used to easily set up communications, find routes and keep track of the distribution and so on.

On the other hand, random deployment in the area of interest is performed with much less control. For instance, when sensors are dropped from an airplane or fired out from some fixed location, we could know the approximated distribution of the sensors in the sensing field but the exact position of each sensor (sometimes we are not aware of the distribution too) is definitely unknown. This kind of deployment generates a lot of problems: the nodes have to effectively self-organize to find out their relative positions and to find out the sink node(s) before the sensing operations can take place.

As a kind of *pervert version of the icing on the cake*, sensors can move during their lifetime. This usually happens by means of an external force, for instance if the sensors are dropped in a tornado to monitor air-flows or when a WSN, deployed to monitor a slope, starts to move with the mountains itself.

Apart from the "by hand" deployment, random WSNs deployment is a crucial part of the WSNs' architecture. The deployment introduces problems into a WSN configuration and performance. Partitioned WSNs can miss the connection to the sink, also for a large part of the nodes. Moreover, a non-uniform distribution of the nodes can bring to pathological states of unfair load distribution across the network if not properly managed.

1.2 An abstract wireless sensor networks model

In this section, we present our *reference model* for wireless sensor networks. The model is used throughout this thesis to develop our studies. We also discuss some problems and some simplifications with respect to the reality. The model that we are going to use is widely accepted and used by the scientific community (Bettstetter, 2004a), (Santi et al., 2001), (Santi and Blough, 2002), (Xue and Kumar, 2004), (Panchapakesan and Manjunath, 2001), (Gupta and Kumar, 1998) and (Dousse et al., 2002).

In our abstract model, we represent the sensors as *zero-dimensional* points in $\mathbb{R}^d$, with $d \in \{1, 2, 3\}$. Each sensor, s_i, can communicate with other sensors with its

[‡] All the other, "picturesque", ways in which sensors can be deployed by, fall in one of these two categories.

antenna. The communication range, r, can be fixed or can vary in some interval $(0, r_{max}]$ for each one or all the sensors.

The communication range r, is a critical factor in wireless sensor networks because the energy consumption in data transmission is very large and grows as a super-linear function of the range. In our model, the power p_i used by s_i to transmit correctly data to s_j must satisfy the following inequality

$$\frac{p_i}{\delta_{i,j}^{\alpha}} \geq \beta, \tag{1.2.1}$$

where $\alpha \geq 2$ is the *distance power gradient*, $\beta \geq 1$ is the *transmitting quality* and $\delta_{i,j}$ is the Euclidean distance between s_i and s_j in $\mathbb{R}^d$. Typically $\beta = 1$ and α depends from the environment in which the communication happens. In the ideal case $\alpha = 2$, but in more realistic conditions $\alpha = 4$ ($\alpha \in [2, 6]$).

Equation (1.2.1) describes the energy used by the sender, but we must remember, that also the receiver uses energy to receive and to decode the signal and this factor is taken in account in our experiments.

Thus, a WSN can be represented as a graph in which vertices represent sensors and edge exists between s_i and s_j if and only if $\delta_{i,j} \leq r$. When the sensors are deployed randomly in $\mathbb{R}^d$, this structure is called *Random Geometric Graph* (RGG for short (Bollobas, 1985)).

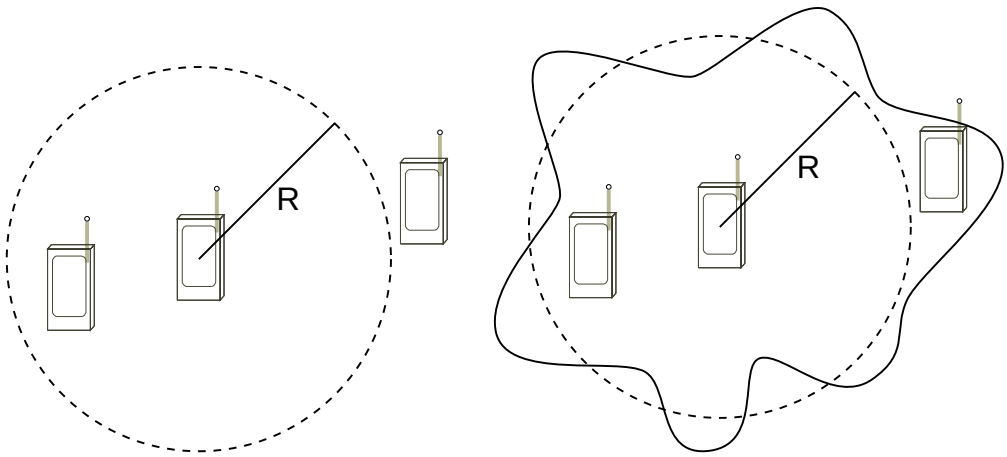
1.2.1 Missing features

The model previously presented is widely accepted by WSNs community as a reasonable approximation of the reality. However, communications in WSNs show some interesting properties which are not captured by the model. Here, we give a brief overview of these properties and we point out which part of the model should be refined to take them into account. The aim of this discussion is to make the reader aware of the approximations introduced.

A first, important, difference is that in the real world the communication volume that the transmitting range draws around the sensor is not a perfect sphere. In the reality, we must consider a spheroid of radius r with a *log-normal* perturbation

$$LN(x) = \frac{exp\left[-\frac{1}{2}\left(\frac{ln(x)-\mu}{\sigma}\right)^2\right]}{\sqrt{2\pi\sigma^2 x^2}} \tag{1.2.2}$$

Equation (1.2.2) highlights that the range r has a high probability to be lightly modified, and a very low probability to be modified significantly. Figure 1.5 depicts the differences between a circular communication range and a log-normally perturbed communication range. In Figure 1.5.a the communication range is a perfect circle



(a) Perfect circle communication range

(b) Log-normally perturbed communication range

Figure 1.5: Communication range models. (a) depicts the ideal case in which the communication range is a perfect disk around the sensor and (b) depicts the more realistic case in which the communication range is no more a perfect disk but it is a irregular area in which the neighborhood relations, with respect to the ideal model, can be changed.

and the central node can communicate only with one of the two neighbors. In Figure 1.5.b the communication range is no more a perfect circle and also the second node can be used to communicate with. This configuration can vary with time and other environmental conditions.

Another simplification introduced by the model is that, in the hidden host problem, a packet collision always causes a packet loss. This is not completely true because of the *frame capture* phenomenon (Arnbak, 1987,Ware et al., 2001). We use an example to show the phenomenon: let us suppose that many people try to talk to us at the same time and using approximately the same volume. In this situation, we hear only *noise* and we are not able to follow any conversation. However, if one speaker pumps up the volume, we are able to hear his/her words and all other voices became only background noise.

The same happens in radio transmissions. When many transmissions arrive at the same time, the receiver is not able to decode them. However, if one transmission is enough powerful, the others become only *white noise*. In order to be able to capture frame f, its power P_f must satisfy the following inequality

$$P_f > \gamma \sum_{i=1}^{N} P_i,$$

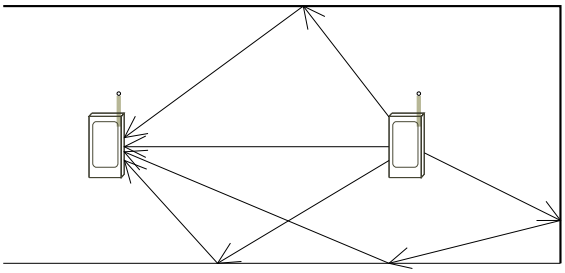


Figure 1.6: Multi-path fading example. The waves propagated by the antenna belonging to the sending device (on the right) arrive to the receiver (on the left), following multiple paths and in different times.

where N is the number of all the frames (excluding the frame f), P_i is the power of the i^{th} frame and γ is the *capture ratio*. Capture ratio is expressed in decibels (dB) and it is a characteristic of the antenna.

Another phenomenon typical of wireless networks is the *power fading*. In the real world, the communication range is not a fixed value r because the power of the signal decreases with distance from sender's antenna. The signal loss L is computed using the following formula

$$L = 32.4 + 20 \log F + 20 \log R$$

L is a function of the frequency F (MHz) and distance R (Km). The derivation of the formula, starting from the propagation model of waves in the vacuum is out of the scope of our research. However, it can be found in (Seybold, 2005).

The last simplification introduced by the model is the *multi-path fading* (Figure 1.6). In multi-path fading the signal can be corrupted by its own copies that arrive at the receiver at different times after that the environment has reflected them. Let us suppose that a radio enabled device communicates inside an obstacle's full area, in this case when a message is sent to a destination device, it is propagated all around the device and the radio waves start to hit the obstacles inside the area. Part of the waves are absorbed, and part of them are reflected by the environment. The reflected waves arrive at the destination device with a certain amount of delay with respect to the waves that proceed following a direct path and they arrive to the destination with a certain amount of delay. If this delay is too long and the reflected waves arrives with a strong enough signal they can corrupt each other.

1.3 Research issues and topics

Since their appearance, various aspects of WSNs have been investigated. WSNs research addresses all the typical problems of wired networks: routing, security, load

distribution and the like. Moreover, there are problems which are peculiar of WSNs. For instance topology control, in which the network structure is adjusted, setting communication ranges, to satisfy some requirement. However, any research in wireless sensor networks must take in account some specific constraints:

Low energy, sensors are equipped with small batteries. Since computations and wireless communications have a high cost, algorithms and protocols should have a low complexity and communications should be reduced as much as possible.

Limited bandwidth, the transmission power must be kept low to save sensor energy, therefore, only a few bytes of data can be exchanged in the time unit in order to save energy.

Unstructured and varying topologies, sensor nodes can *sleep* to save power or they can have faults or, more simply, they can move. In all the cases, as a result, we have a topology change and, for instance, the routes from a sensor to the sink may change and they must be computed again, and/or some datum can be lost.

Low-quality communications, low energy communications tend to be received with a lot of errors at long distance and WSNs must be able to cope with it using correcting codes and/or retransmissions.

Hostile environment, the nodes can be destroyed by the environment and/or environmental radiations can cause transmission failures. In any case the protocols must be tolerant to such events as in the case of topology changes.

In the rest of this thesis, we focus on *non-uniformity*, in its influence in WSNs and how we can overcome the problems that are originated by it. To show up the non-uniformity influence and the techniques that can be used to get rid of it we use a case study. Our case study is the non-uniformity issues in data management in WSNs.

Research on data management. Data management research covers the problems related to storing data in a balanced and fault-tolerant way in WSNs. The importance of these problems is due to the unreliable nature of sensors and by the low quantity of memory on board.

The central problem to be solved is to be able to locate data in an efficient way for both store and load operations. In WSNs, data can have two different ways to be located. Some data are localized in some area and must be stored there, for instance, the temperature measured in a given place can be kept in such a place. On the other hand, other kind of data are not related to a particular location, for instance, the average temperature of a region has not a particular position in which

it can be stored in. This last kind of data must be stored in a way such that: (*i*) it balances the load across the whole network and (*ii*) it is readily accessible when needed. Moreover, data produced at a high rate can easily fill the memory of a single sensor and it must be stored around the network in a balanced way.

A promising approach to data storage in WSNs is based on *geographic hash tables* (Ratnasamy et al., 2003). In this approach, a datum is identified by a *key* and there is a *hash function* that provides, given the key, the geographical coordinates where the datum must be stored in or can be loaded from. The hash function is used to distribute data in a balanced way, giving to each sensor the same amount of data to store. The protocol stores data in some sensors in the proximity of the coordinate identified by the hash function, in this way it adds fault tolerance because data is stored in multiple nodes.

As we will point out in Chapter 4, the geographic hash tables, as they are presented in (Ratnasamy et al., 2003), tend to have problems in presence of non-uniformity both for the load distribution and for the fault tolerance aspects. For these reasons, we have chosen to use them to point out some of the typical problems introduced by non-uniformity and to show up what is the path that we must follow to solve these problems.

1.4 Summary

In this chapter, we reviewed the principal features of WSNs, giving a gentle introduction to them. We reviewed the architectural properties of both single sensors and of the networks made up of these little devices.

WSNs are a special case of ad hoc networks: once deployed in an area of interest, the sensors have to self organize to provide the basic network services as routing. The main difference between ad hoc networks and WSNs is given by the tight constraints of such kind of networks.

While ad hoc networks are made up of devices, such as PDAs, whose computational power increases everyday, WSNs are made up of inexpensive devices with limited resources, both in terms of energy and computational power.

These constraints move all the ad hoc networks' problems to a more difficult level of resolution: algorithms need to be simple and very fast, protocols need few messages to be executed and brand new energy saving strategies have to be found to keep the network fully functional.

Then, we moved in the definition of an abstract model of WSNs useful for the analysis and for the formalization of our research in the field of WSNs. Our model is suited to catch the problems that we are going to present in the next chapters. We need a suitable model to study the influence of non-uniformity in WSNs. For this reason, we need to be free from all the characteristics that are not influenced (or not influenced very much) by non-uniformity. In our research, we need to focus on

the distinguishing characteristics of the *non-uniformity issues* that can influence, in a negative or positive way, the current results in WSNs that were build up on top of the uniformity assumption.

In the next chapter, we are going to give full suite of motivations to the reasons way the *uniformity dogma* is not suitable for WSNs' research, how we can characterize non-uniformity and how we can start to *think* in a non-uniform way to create *non-uniformity aware* architectures, protocols and systems.

Chapter 2

Non-Uniformity

Variety of mere nothings gives more pleasure than uniformity of something.
- Jean Paul (Johann Paul Friedrich Richter)

Abstract

Most of current WSNs research focus on uniform networks: networks in which sensors are equals to each other or networks that are distributed following the uniform distribution. In current WSNs research, uniformity is a dogma, a believed true assumption that no one wants to offend. In this chapter, we start from a slightly different point: we simply state that uniformity does not exist. Moreover, the uniformity assumption that can be found in the mainstream WSNs research produces solutions that can be non-tolerant with respect to non-uniformity and these solutions can be difficult to adapt to a non-uniform environment. In this chapter, starting from the consideration that uniformity does not exist, we begin to study non-uniformity and its influence in WSNs research.

In this chapter, we introduce non-uniformity and its influence in WSNs. Our intention is to show that the typical uniformity *assumption* found in almost all the research work in WSNs field is best suitable to be defined as an uniformity *dogma*: the assumption was never motivated by anyone, but was assumed as something true and reasonable.

In this chapter, we start with some general consideration about the uniformity which target is to *disrupt* the uniformity dogma: we show how the uniformity is rarely present in nature and, as a consequence of this, in every human artifact as WSNs.

Once this task is performed, we review non-uniformity: we describe it formally and we present four models of non-uniformity that may influence WSNs, namely

the *deployment non-uniformity*, the *geographical non-uniformity*, the *functional non-uniformity* and finally the *movement non-uniformity*.

These models present a rough taxonomy of non-uniformity and *we do not pretend that this taxonomy is the best one or the only one*: we define this taxonomy because it is an *aurea mediocritas* for the model developed in the rest of the thesis and a reasonable representation of the main characteristics of non-uniformity for WSNs.

Chapter organization. This chapter is organized as follows. Section 2.1 points out the importance of the non-uniformity as a substantial part of WSNs research. Section 2.2 presents our taxonomy of the non-uniformity in WSNs. Section 2.3 presents the (few) related work on non-uniformity in WSNs. Finally, Section 2.4 draws the conclusions of the work described in this chapter.

2.1 Why does non-uniformity matter?

In our opinion, non-uniformity matters because in "nature" *perfect* uniformity does *not* exist.

If we observe reality from a distant point of view, we can find uniformity almost everywhere. However, if we pay more attention in observing the world, for instance with a magnifying lens, we are no more able to find such uniformity.

As a common experience, if we observe the smooth marble surface of a beautiful renaissance statue we can argue that it is uniform: no asperities can be perceived by eyes or fingers. But, if we observe it using a microscope, this uniformity gets lost: the unveiled surface is full of asperities and it does not look possible that it could be the same surface as before.

In computer science, sometimes we manage data structures or algorithms that need to be uniform. For instance, when we use hash tables we need to provide uniform hashing algorithms to have a good distribution of data and in turn of this provide balanced tables. Moreover, in randomized and Monte-Carlo algorithms, we need uniform random number generators to explore in a random, but fair, way some result space. But, when we want to simulate, or analyze the real world, we cannot pretend that the simulated world behaves uniformly because in the real world uniformity is far from reality.

This happens in WSNs research too. Even if we distribute sensors "by hand" and we create structures with interesting properties as grids, meshes and so on, we cannot call them *uniform* because we may introduce some errors during the distribution and/or once distributed it may happen that their behavior (for instance the transmitting range) is not uniform.

If we drop sensors from above, using an airplane or a helicopter, we cannot obtain an uniform distribution at all. The sensors are more likely to fall accordingly to some more complex distribution such as Gaussian or Gaussian-like distributions.

Now, let us suppose that we can distribute sensors with perfect uniformity. Probably the environment itself is not uniform. Real applications of sensors include battlefields, harsh regions and urban scenarios. Scenarios too variegated and complex to be defined as uniform (or to find some kind of ruled non-uniformity as a Gaussian curve): these scenarios have obstacles that, for instance, change the behavior of transmissions, disabling the sensors to communicate each other.

2.2 Models of non-uniformity

In this section, we present the four models of non-uniformity that we have spotted out during the development of this thesis. The classification of the models of non-uniformity, that we developed, is a suitable high-level classification that catches the four main families of non uniformity. We do not think that this classification is the only possible one. As stated before in this chapter, these four families represents an *aurea mediocritas*, literally a *golden way in the middle*: this taxonomy catches the essential differences between the different kinds of non-uniformity and it defines only four models but it may be extended in multiple ways mixing up two, or more, of the presented models or creating sub models inside the ones presented here.

In a WSN, we can find different kinds of non-uniformity, appearing in different situations. Non-uniformity can be due to the method used for the deployment of the network (e.g., when the sensors are dropped from above) or it can depend on some functional patterns of the WSN itself. Moreover, non-uniformity can be due to the geography or by obstacles present in the deployment area. Finally, a last kind of non-uniformity is typical of WSNs in which the nodes can move, by themselves or simple because the environment can move.

We refer to these four models as: the *deployment non-uniformity*, the *geographical non-uniformity*, the *functional non-uniformity* and finally the *movement non-uniformity*.

	Intentional	**Accidental**
Static	Deployment	Geographic
Dynamic	Functional	Movement

Table 2.1: Our taxonomy of non-uniformity in WSNs.

We developed this taxonomy for the models of non-uniformity because, in our opinion, this is a reasonable high level representation of the possible kinds of non-uniformity. Moreover, our taxonomy catches four aspects of the possible distributions. Table 2.1 shows the four aspects. These aspects represent four main characteristics of distributions:

Intentional: the distribution that we are going to analyze (or its approximation) is known *a priori*. For instance, when we drop sensors from above, we are

quite confident of the fact that they will distribute following a Gaussian-like distribution.

Accidental: the distribution that we are going to analyze (or even its approximation) cannot be known *a priori*. For instance, when we drop sensors on a terrain full of hills and depressions, we cannot estimate what the distribution will be.

Static: the distribution that we have at the beginning will be almost the same during all the network lifetime. For instance, once we deploy the sensors, if the sensors cannot move, they will be in the same place for all the network life time.

Dynamic: the distribution of the sensors changes with time. If the sensors moves the physical distribution changes. If the network is made up of different kind of sensors, its behavior changes with the possible interactions between nodes, considering that this behavior is known only at the network run time it has to be considered dynamic.

2.2.1 Deployment non-uniformity

The way in which the sensors are deployed is the first cause of non-uniformity in our taxonomy. WSNs deployed by hand can be structured in very *fascinating* topologies with a lot of interesting properties, such as meshes, tori and many more *structured* distributed topologies. Unfortunately, real WSNs are usually deployed in a random fashion.

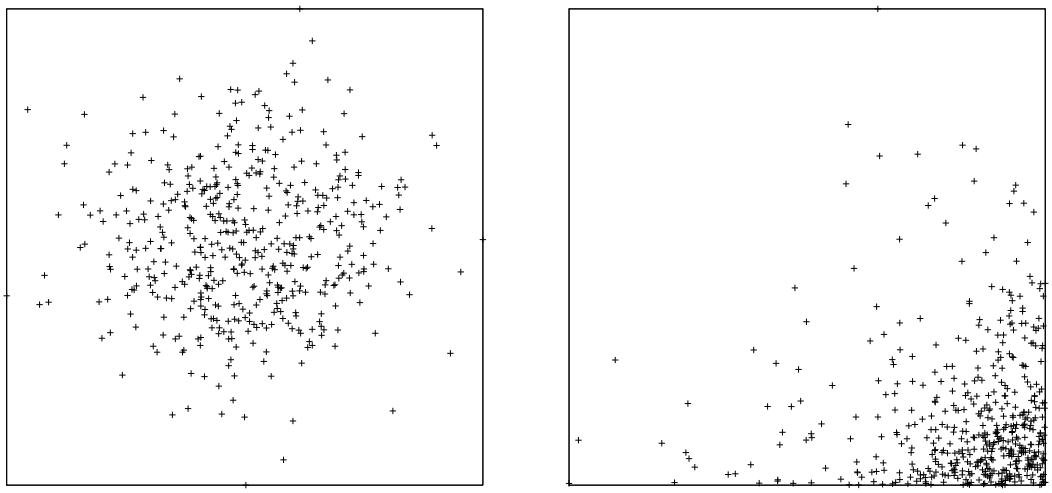
Random WSNs deployment follows two main kinds of non-uniform distributions: *Gaussian* and *skewed* (Zhou et al., 2005).

The Gaussian distribution, depicted in Figure 2.1.a appears when the WSNs are deployed from above, for instance when the sensors are dropped from an airplane or from any other kind of flying object. This kind of nodes distribution is well suited for military applications in which the sensors are deployed on the battlefield without any predefined pattern. The Gaussian distribution of sensors on a plane is described by the following Equation

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y}\, e^{-\left[\left((x-\mu_x)^2/2\sigma_x^2\right)+\left((y-\mu_y)^2/2\sigma_y^2\right)\right]}. \tag{2.2.1}$$

Equation (2.2.1) is the result of the product of two independent Gaussian distributions. The terms μ_x and σ_x represent the *mean* and the *standard deviation* for the Gaussian distribution on the x-axis; in the same way μ_y and σ_y represent the mean and the standard deviation of the distribution on the y-axis.

Typically Equation (2.2.1) is found in the form where both μ_x and μ_y at zero, to represent the fact that the distribution is centered in an ideal $(0, 0)$ point.



(a) Gaussian (b) Skewed

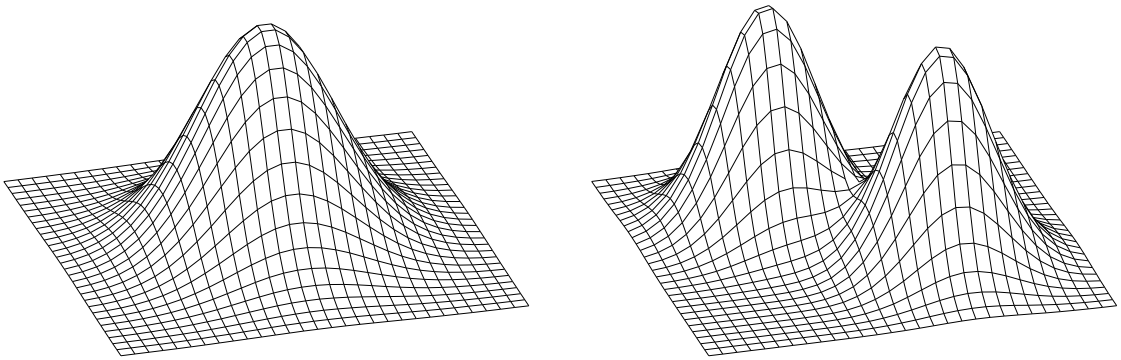
Figure 2.1: Deployment non-uniformity. (a) depicts the typical Gaussian distributed network more dense in the center and sparser on the borders of the distribution and (b) the skewed distribution with the majority of the nodes in an angle.

The values of σ_x and σ_y are the most important, because they give the *shape* of the distribution, low values produce a steep distribution and high values a flat one: if we drop sensors from a fixed point then $\sigma_x = \sigma_y$ (or more precisely $\mid \sigma_x - \sigma_y \mid \leq \epsilon$, with ϵ small); if we drop sensors from a moving point then $\sigma_x \neq \sigma_y$ (or more precisely $\mid \sigma_x - \sigma_y \mid > \epsilon$, with ϵ small). In other terms, the values of σ_x and σ_y can be combined to produce a large variety of shapes more or less suitable to describe different deployment situations as the deployment actuated by a fast moving objects or static ones.

The Skewed distribution, depicted in Figure 2.1.b, appears when the WSNs are deployed shooting them from a fixed point: for instance a cannon or any other suitable devices for such operation. In this case the sensors take position in a cone with maximum density at one corner of the deployment area (the impact point); then, their density exponentially decreases with the growth of the distance from the maximum density point. The skewed distribution on a plane is described by the following Equation

$$f(x,y) = \frac{1}{\beta_x \beta_y}\ e^{-[x/\beta_x + y/\beta_y]} \text{ with } x \geq 0 \text{ and } y \geq 0. \tag{2.2.2}$$

Equation (2.2.2) is the result of the product of two independent exponential distributions. The terms β_x and β_y represent the *shape* of the exponential on both axis



(a) Mono-modal (b) Multi-modal

Figure 2.2: Modality of distributions. (a) depicts a mono-modal distribution formed by one Gaussian distribution and (b) a multi-modal distribution made up of two independent Gaussian distributions.

x and axis y. When the shape decreases the function is more steep.

These two distributions, Gaussian and Skewed, can be found in two main variants: *mono-modal* and *multi-modal*, respectively depicted in Figure 2.2.a and in Figure 2.2.b.

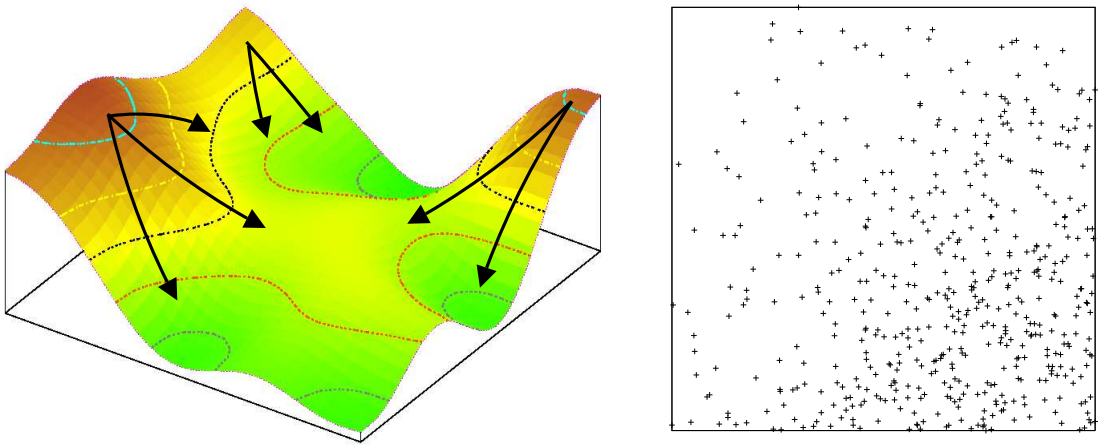
A distribution is mono-modal, when it is deployed from a single point and the nodes follow a single distribution.

On the other hand, a distribution is multi-modal when it is deployed from multiple points (in one or more times) using the same distribution or using different distributions. For instance, if we deploy a WSN dropping sensors from two distant points we obtain a distribution as the one in Figure 2.2.b, with two different peaks.

2.2.2 Geographical non-uniformity

WSNs' distributions can be affected by the geographical characteristics of the deployment area in which the sensors lay on. In this non-uniformity models, we can distinguish two cases.

In the first case, the deployment region is full of obstacles (walls, building, people, environmental radiations etc.) that prevent nodes physically close to communicate. In this situation, the sensors have an *altered* perception of the space that can bring to problems where, for instance, nodes are equipped with GPS antennas and interpret these connection errors as GPS errors or worst as security attacks from malicious users.



(a) Soil (b) Hill

Figure 2.3: Geographical non-uniformity. (a) depicts a sample geographic terrain conformation with a central valley and (b) an example of a network deployed accordingly to a Hill distribution with the sensors more dense in an angle.

In the second case, sensors dropped on a terrain tend to follow the terrain shape and to fall into soil depressions. This situation can bring to networks in which the nodes are more dense in the valleys and less dense at high points. If we add the fact that hills can represents obstacles that prevent the sensor-to-sensor communications, we can have an highly partitioned network.

For the case in which sensors follow the terrain shape, we explore two different distributions. The first one, fully generic and related only to the geography, and the second one, that is a distribution function that approximates reality.

In the first one, as stated before, the terrain geography can cause node redistribution as depicted in Figure 2.3.a. The nodes deployed on steep terrains tend to fall down following the direction of the slope. This effect is *per se* complex to examine and to evaluate but it can become very difficult to model if we use it in conjunction with the effects of the deployment non-uniformity.

The second one is a distribution function that approximates reality. The *Hill distribution*, depicted in Figure 2.3.b was first presented in (Orecchia et al., 2004a). The Hill distribution is very simple, and comes from an intuition and not from experimental results. This distribution simulates an uniform sensor distribution on a hill and the consequent fall of the sensors from the top of the hill to its base. In the paper, the authors provide only generator functions for the sensors coordinates generation: $x = \sqrt{u}$ and $y = \sqrt{v}$ where u and v are uniformly distributed variables in $[0, 1]$. From the generators, it is simple to find out the distribution function of the sensors on the plane

$$f(x, y) = 4xy \text{ with } x \in [0, 1] \text{ and } y \in [0, 1] \tag{2.2.3}$$

Equation (2.2.3) is very simple but reasonable and, at first sight, we can spot similarities with the Skewed distribution due to the maximum concentration in one corner of the distribution.

2.2.3 Functional non-uniformity

The functional non-uniformity is related to *functional patterns* of the network itself. A single network can be made up by multiple kinds of sensors, with different capabilities and constraints, or can be formed by a single kind of nodes behaving in different ways. These functional patterns are related to two basic concepts.

In the first one, the nodes of a WSN can adopt some kind of sleeping pattern to enlarge life time as in (Ye et al., 2002), (Ye et al., 2004), (Cerpa and Estrin, 2002), (Chen et al., 2002) and (Xu et al., 2001). The effect of these sleeping patterns is to create an effect of dynamic non-uniformity in which the sensors' distribution is not only a function of the deployment method but also a function of time, the sensor "disappears" for some time and than "appears" again. Depending on the sleeping pattern, we can experience an effect of the same region that is alternatively monitored by two or more networks sharing only few nodes.

In the second one, a single WSN can be made up of different kind of nodes with different capabilities and constraints. An example of this kind of networks in which two kind of nodes are present:

1. *Sensing nodes*, which are small and with low power, and perform data acquisition from the environment;

2. *Communication nodes*, which are more powerful and with a high power communication system, that can communicate at a larger distance providing a bridge-like service between sub-areas of the network.

In this way, the sensing nodes can chose two ways to communicate: if they want to communicate with close peers, they use the usual multi-hop communication but if they need to communicate with nodes that are far away, they can use the communication nodes as *communication highways*. To have a perfect collaboration, that is a situation in which the two kinds of nodes works together to perform a monitoring task, we require a load balancing between the sensing and communication nodes because in this case all communication nodes need to manage the same number of sensing nodes.

One example of functional non-uniformity is provided by networks using three layers of nodes: the sensing nodes, the sink nodes and the proxy nodes (Desnoyers et al., 2005). in this kind of networks, the proxy nodes collect data from the sensors, do some computation on these data and finally send data to sink nodes.

Another example is provided by *Mules* (Shah et al., 2003). The mules are moving proxies that go around the network collecting data from sensors and bring data to one or more sink nodes.

2.2.4 Movement non-uniformity

The last non-uniformity model that we want to present is the movement non-uniformity. This non-uniformity model can be due to two different factors. In the first one, the sensors are free to move in an autonomous and predictable way, for instance, the sensors that are piggybacked on moving objects such as industrial robots that the sensors are intended to monitor and control. In the second one, the sensors are supposed to be stable and/or the movement is not predictable, for instance, the sensors are dropped in the water to track the flows or sensors that are attached to animals and are used to monitor the wildlife.

The sensors distributions that describe movement non-uniformity are functions of both time and space. Moreover, we can have situations in which the sensors can converge to a single point and create a unbalanced network for a short period of time, or the sensors can move randomly changing network topology continuously.

In literature we find few studies aimed to find out the characteristic distributions for movement uniformity. One of these, few, works is (Bettstetter, 2001). In the paper, we can find a relevant study of this phenomenon limited to nodes that move accordingly to the *random waypoint* strategy (Johnson and Maltz, 1996). These sensors tend to create a node distribution (in function of the time) similar to a Gaussian distribution. In random waypoint strategy, at the beginning each node is stationary in a random location, then it randomizes a point in the plane and moves there, waits a random time and than moves again. This movement creates a concentration of nodes in the middle of the area, because the nodes, to go from one point to another, cross the center of the area or pass nearby the center of the area.

2.3 Related work

Due do its nature of *dogma*, the uniformity assumption prevails in almost all the research work on WSNs. As a consequence of this, the study of non-uniformity in WSNs is an emerging topic. By now, only a few papers approach different problems in WSNs taking into account networks deployed in a non-uniform way (Bettstetter, 2004b), (Orecchia et al., 2004a), (Tilak et al., 2003), (Zhou et al., 2005) and (Bash et al., 2004). Between them, we can find non-uniformity depicted in two ways.

The first way addresses WSNs that are deployed in non-uniform fashion. In this area, we can find two main contributions. The first one is (Bettstetter, 2004b), in which the author studies the connectivity properties of sensor networks with uniform and Gaussian distributions. For Gaussian distributions, given the transmitting range of sensors and the shape of the distribution function, the author finds out the radius around the maximum of the function in which all sensors are connected. A second result is (Orecchia et al., 2004a), in which the authors study the problem of broadcast in sensor networks distributed with both uniform distribution and "Hill" distribution, which is a quite simple but reasonable distribution that we will

study and use in the rest of the thesis. Finally, in (Zhou et al., 2005), the authors propose a construction scheme of *small worlds* for the physical topology of Multiple-Input Multiple-Output (MIMO) wireless networks. In this work, the authors take into account various topologies (uniform, Gaussian, skewed and grid) to study the possibility to create *short cut channels* to optimize the network communications.

The second way does not use non-uniformity in terms of deployment. In (Tilak et al., 2003) the non-uniformity is related to the concept of *non-uniform information granularity*. The non-uniform information granularity is related to the concept of accuracy of information. The paper states that the required accuracy, or precision, of information is proportional to the distance between the producer and the consumer of the information. Another result is (Bash et al., 2004), in which the non-uniformity is related to information sampling from the network itself. The authors provide a method to perform data sampling from a sensor network with *approximately uniform* methods. Even when a network is non-uniformly distributed (they study the case in which the nodes are hand distributed into a building to monitor it), the authors pick data form the network uniformly. This is possible because the sensors space is normalized to an uniform space using Voronoi regions computed in a distributed fashion.

2.4 Summary

In this section, we introduced non-uniformity and we present some keys to motivate our work.

At the beginning, we presented the need for non-uniformity in current WSNs research and we pointed out how the *uniformity dogma* is widely used in research while the world itself, that we are going to explore with our research, is, by its own nature, *non-uniform*.

Then, we moved to the classification of various models of non-uniformity. The model that we propose out are general and each of them covers a large part of possible problems and networks. Our taxonomy of the models of non-uniformity is, intentionally, coarse gained. We preferred to give an intuitive division of the possible sources of non-uniformity without being too formal. We know well that our taxonomy can be rewritten in a finer, or simply different, way but we think that a finer grain in the taxonomy does not help the comprehension of the problem itself and it risks to be a pure formalization and style exercise.

Our classification presents four possible sources of non-uniformity. The *deployment non-uniformity* is related to the way in which the sensors are deployed in the interest area by the user. The *geographical non-uniformity* is based on the idea that the environment that the sensors are intended to monitor influences the way in which the sensors can communicate. The *functional non-uniformity* catch the different behaviors of the sensors. Finally, the *movement non-uniformity* is related to the idea of movement of the sensors themselves.

In the related work section, we presented the few papers that walked inside the non-uniformity world and we realized that with the help of these few papers alone, we are not able to comprehend non-uniformity. Moreover, with the only help of these papers, we are not able to provide new solutions and, from our point of view more important, a new way of thinking solutions that are able to deal with non-uniformity.

Chapter 3

Data Management

"Data! Data! Data!" he cried impatiently. "I can't make bricks without clay."
- found in "The Adventures of Sherlock Holmes"

Abstract

All the possible uses of WSNs deal with the idea of *data acquisition* and *data retrieval*. These two concepts are strictly related because data retrieval is the response to a user query. The user should be able to actively program the network, via control programs, to retrieve data that is considered useful. In traditional WSN systems, these tasks consist in transmitting sensed data to a powerful node (the sink) which performs data analysis and storage. However, these models resulted unsuitable to keep the pace with technological advances which granted to WSNs significant (although still limited) processing and storage capabilities. For this reason, more recent paradigms for WSN introduced *data base* approaches to define the tasks of data sampling and processing, and the concept of *data-centric storage* for efficient data access. In this chapter, we revise the main research contributions in the data management field.

All the possible uses of wireless sensor networks deal with the idea of *data acquisition* (by the sensors) and *data retrieval* (by the user). These two concepts are strictly related because data retrieval is the response to a user query. The user should be able to actively program the network, via control programs, to retrieve data that is considered useful. This can be achieved in many ways, following various paradigms.

WSNs applications can be classified in two different families.

The first family is the one in which sensors continuously check the environment for some particular data pattern (as in an intrusion detection system). In these applications, the cooperation of nodes is be essential in providing extra information to the end user. Nodes that are aware of their own position can communicate to produce a

localization of the event using triangulation or other systems (Bejar et al., 2001, Krishnamachari et al., 2002). This collaboration is essential to provide meaningful data to the end user.

The second family is the one in which sensors are employed to monitor some physical characteristic of the environment. For instance, sensors can provide the sensed temperature to the user. This use of the network produces data streams from a single sensor to the sink. In this case, the real challenge in network data management is to find an optimal strategy to delivery this (maybe) large amount of data to the user, in the most efficient way. The simplest approach to this problem is *data aggregation*. In data aggregation, the nodes used to perform routing (also called *relay nodes*) take packets that are sent to the sink and combine the different pieces of information in a single packet to reduce the total amount of transmissions. Data aggregation still works with small amount of data because the maximum size of packets is fixed and cannot be exceeded.

Modern approaches move the computation of data in the network itself (Madden et al., 2002b, Madden et al., 2002a). Sensed data are not directly sent to the sink and the computation is not done off-line: sensors that acquire data also process them to extract information. The nodes can actively cooperate to produce highly meaningful correlated data (such as the average, the variance and other statistical measures or spacial and temporal associations of data). Moreover, these modern approaches place the storage itself inside the network. In cases in which the sink can be unavailable for long periods of time (or in cases in which the sink is not present inside the network and the user has to enter the network to query it) these approaches are able to distribute data inside the network in such a way that the storage is fault tolerant and load balanced. Some of these systems provide to the user a high level distributed query language and engine (*a là* SQL) to query the network.

Chapter organization. This chapter is organized as follows. Section 3.1 provides a brief introduction to the data management issues and models. Section 3.2 points out which services have to be provided by a data management system in WSNs and their underlying needs (e.g. routing). Section 3.3 describes the state of the art of the database implementation inside WSNs. Section 3.4 gives a wide range introduction to the data centric storage (DCS) model, that will be the central topic of the research work that we present in Chapter 4. Finally, section 3.5 summarizes and draws some conclusions regarding the research work described in this chapter.

3.1 Data management in wireless sensor networks

In current WSNs, data management tasks are performed accordingly to different *network storage models*. A network storage model defines how to manage data inside a WSN, starting from data acquisition up to data retrieval by the user. For this

reason, each network storage model needs to define how to implement, even partially, the basic services necessary to the good work of a data management system: (*i*) *network programming*, in which the sensors are programmed to replay to queries (set up of routes, data aggregation strategies etc.). (*ii*) *data acquisition and in-network aggregation*, in which the data is physically acquired by transceivers and refined to produce more useful streams. (*iii*) *data retrieval*, that provides the collection of data by an external entity.

In the literature, we can identify four main models to data management: namely *external storage*, *directed diffusion*, *data centric storage*, and *database model*. Each one of these models implements a part or all the above services. The services provided by these models are summarized in Table 3.1.

	External Storage	Directed Diffusion	Data Centric Storage	Database Model
Network programming	no	yes	no	yes
Data acquisition and in-network processing	partially	partially	yes	yes
Data retrieval	yes	yes	yes	no

Table 3.1: Correspondence between models and services.

External storage is the simplest storage model and provides node cooperation only by data aggregation and data forwarding. Directed diffusion is able to program the network to optimize the data flow from the sensors to the sink, providing a better usage of the resources with respect to external storage. Data centric storage is able to use the network to easily store and retrieve a datum using its meta-datum (an unique *name* for the datum) as a key. Data centric storage does not need, or provide, network programming facilities for the data management. Such facility can be added on an upper layer laying on this one. Finally, the database model is able to abstract the network as a relational database to perform complex queries that are executed by the sensors themselves.

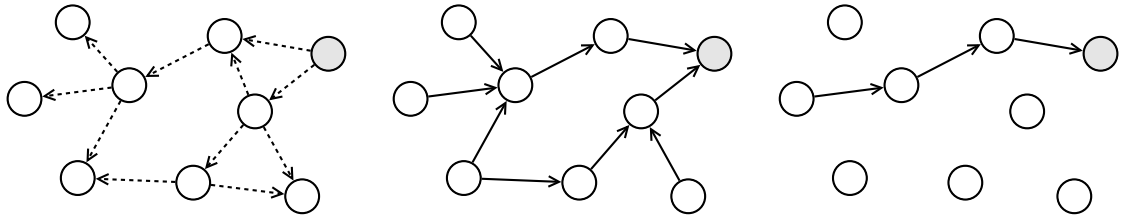
In the rest of this section, we give more details on these four models. We will focus more on directed diffusion and external storage because the data centric storage and the database model will be the focus of the rest of this Chapter.

External storage model. In the External Storage (ES) model, the nodes send the sensed data, in both raw or more refined forms, to one or more sinks which perform the actual aggregation, processing and storage. In this model, the sensor nodes perform only data acquisition and send sensed data to the sink. To this purpose, they also have some routing capability. Each sink node collects data from the sensors and implements an interface for the user. For instance, a sink might be a PC acting as router between the sensor network and other networks.

The main advantage of this model is that it is simple to implement. Moreover, if the network produces few data it can be energy efficient.

The drawbacks of ES are represented by the poor usage of the network's computational resources because the sensors, that can be capable of more complex operation could refine data. Moreover, data correlation is due to the sink and off line processing can require multiple sink-to-sensors communications while in an in-network computation the sensors can efficiently self organize to provide correlated data. For instance, a user who wants to correlate various temperature sampling from a region (far from the sink) needs to query all the nodes of that region while an in-network processing strategy is able to program the sensors of the same region to communicate and provide only the correlated data.

Diffusion model. Directed Diffusion (DD) is one of the first approaches proposed for data management in sensor networks (Intanagonwiwat et al., 2000). The user requests some specific data with a query including a tuple $\langle key, value \rangle$ and a *data rate*, which specifies the amount of data that must flow in the unit of time.



(a) interest dissemination (b) gradients (c) delivery of a datum

Figure 3.1: Directed Diffusion: (a) the interest dissemination from the sink node (in gray) to the sensor nodes (in white), (b) the gradients that are built in response to the interest dissemination and (c) the delivery of data from a sensor node to the sink following one gradient.

The first step in data retrieval is represented by the *interest dissemination* phase in which the sink broadcasts the query in the network (as depicted in Figure 3.1.a). The broadcast used to disseminate the query sets up a directed acyclic graph (DAG) in the network which is rooted at the sink (as depicted in Figure 3.1.b). A node x is connected to a set of *upstream* nodes (the *gradients*) which are its relay nodes when x sends packets to the sink. Each node receives the query, records the query data rate and sets up the corresponding *gradient* toward the sink. The nodes that receive or sense some data that match one or more interests forward them up the route created by the gradients according to the data rate (as depicted in Figure 3.1.c). The sink, at the reception of messages with the queried data, can exploit *reinforcement* of the paths that bring data with higher data rates sending a packet down to the data stream. Reinforced paths may augment their data rate, while paths that are not reinforced expire after a given amount of time.

The main advantage of the DD model is that the interest propagation happens hop by hop and the network does not need to use long communications to set up or modify paths' characteristics. DD is able to provide multi-path routing and delivery using the reinforcement of multiple paths. The nodes are capable to repair or tune the paths in a local way sending local reinforcement messages.

DD presents also two main drawbacks. The first one is related to the poor load distribution of the protocol: the nodes that are closer to the sink are burdened with data and control packets. The second drawback is the limited opportunity for in-network data processing because data may pass through different paths and aggregation and processing often happens only at the sink or at its neighbors.

Some recent improvements on DD have been proposed in (Marcucci et al., 2005) and (Liu et al., 2003, Liu et al., 2005).

In (Marcucci et al., 2005), the authors present Directed Diffusion *Light* (DDL). The improvement of DDL acts on interest propagation: each node can have only a limited set of gradients (while in DD the size of this set was not limited). In this way, the DAG is sparser and it is not formed by all the nodes in the network. When reinforcement is performed the DAG may change to include previously discarded nodes. This offers the opportunity of replacing depleted nodes with fresh spares and thus extends the network lifetime.

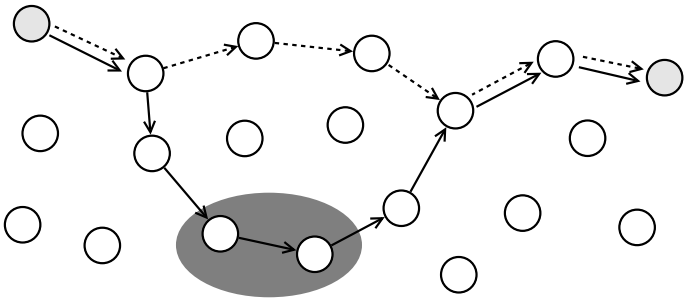


Figure 3.2: Information Directed Routing: a message from the query proxy node to the exit node, does not follow the shortest path (the dashed line), but follows a longer path (solid line) to get data from the nodes inside the dark gray area.

In (Liu et al., 2003, Liu et al., 2005), the authors present the Information Directed Routing (IDR). IDR can operate in two ways. In the first way, a source node injects a query with a particular interest and such query is routed throughout a path build upon interest gradients spread by the possible query receivers. In the second way, depicted in Figure 3.2, it is able to find paths between a source node and a destination node (called respectively the *query proxy* node and the *exit* node in the paper) with maximum information gain. When the source node injects the query into the system, it is routed as close as possible to the areas that have a high probability to be information rich (the nodes inside the dark gray area in Figure 3.2) to collect as much data as possible.

Data centric storage. In the Data Centric Storage (DCS) model, the sensors network itself provides support to data storage (Ratnasamy et al., 2003). Data is stored in the network according to keys (meta-data) which are also used for data retrieval. Given a data and its key, DCS stores the data in a subset of sensors which is uniquely determined by the key.

Database model. The Database model (Madden et al., 2002b, Madden et al., 2002a, Yao and Gehrke, 2002, Amato et al., 2005a, Amato et al., 2006a, Amato et al., 2006b) is a more recent model that offers a high abstraction. In this model the user can issue SQL-like queries on the network. The WSN is abstracted as a distributed database and the user can specify both queries and their duration (to provide temporal aggregation). In these systems, the queries are translated in data acquisition, data processing and data transfer operations that are performed in a distributed way by the nodes of the network.

3.2 Data centric storage and database support systems

Both the data centric storage and the database models need some basic systems to operate in a correct and efficient way. These support systems can be grouped into three main areas: the *localization* to be able to associate a position in space to data or to nodes, the *routing* to find paths to move data from one node to another and finally *redundancy* to achieve fault tolerance.

Localization is necessary to relate data with the physical location where they have been collected, but it can also provide support to geographic routing (which uses geographical position of the nodes to perform routing). Routing is essential because sensor networks are generally multi-hop, hence data must be routed through multi-hop paths to reach the sink or a storage point. Redundancy enables a more reliable data storage because each datum is stored in multiple copies inside the network. In this way, an unique node does not represent a single point of failure.

We address localization in Section 3.2.1, routing in Section 3.2.2 respectively and redundancy in Section 3.2.3.

3.2.1 Localization systems

Localization systems are used to find the position of a sensor. The position of a sensor can be useful for two different reasons:

1. Queries request data from specific locations.
2. A datum must be stored in a specific location for later retrieval.

The localization systems are grouped into three main categories: *GPS-based*, *approximate*, and *virtual*.

GPS-based systems. GPS-based systems assign real geographical coordinates to sensors. In this model, each sensor is equipped with a GPS antenna and it is able to receive the GPS signal and to locate itself.

This solution is not always applicable in current WSNs because GPS receivers are expensive both in terms of money and energy and they may fail if the nodes are located inside buildings or in areas in which obstacles prevent the GPS signal to arrive (Kaplan, 1996).

The growing capabilities in hardware could provide, in a near future, cost effective and energy inexpensive GPS receivers. If this happens, solutions that now seem poorly applicable could become future standards.

Approximate coordinate systems. The physical coordinates of a sensor can be approximated using special hardware and/or network topology. In these systems, the geographical coordinates of some of the nodes must be known (for instance using GPS), and a distributed algorithm exploits this information to infer the approximate position of other nodes. To this purpose the algorithm exploits information about relative distances or angles between nodes which can be obtained equipping the sensors with specialized hardware. More specifically, as in (Savvides et al., 2001), (Nasipuri and Li, 2002), (Niculescu and Nath, 2003b), (Niculescu and Nath, 2003a), (Bulusu et al., 2000), (Nagpal et al., 2003) and (Niculescu and Nath, 2003b), the nodes can estimate the relative distances between themselves using different techniques: *signal power*, which estimates the distance based on the power of the received signal, and *difference in arrival time* which exploits messages sent on different communication medium (e.g. radio and acoustic) that have different propagation times, and estimates the distances based on their inter-arrival time. A different approach exploits the *angle of arrival* of the messages to estimate the relative position of a sender. This approach however must be supported by directional antennas.

A general classification of approximate coordinate systems divides them in *range based* and *range free*.

In range-based systems some nodes (called anchors) know their physical position. Non-anchor nodes compute their position using multilateration techniques which minimize the square of the distance between them and the anchors. This computation returns the position of the nodes.

In (Savvides et al., 2001) a node can use multilateration techniques using both anchors and nodes that used anchors previously to estimate their position. In (Nasipuri and Li, 2002) the anchors are special nodes equipped with powerful radios with narrow and rotating beams. The non-anchor nodes use both the signal strength and the angle of messages sent by the anchors to locate themselves. In (Niculescu and Nath, 2003b) and (Niculescu and Nath, 2003a) less accurate systems are provided: they use an approximate strategy in which only a fraction of the nodes has a GPS and such nodes are used as landmarks for the other nodes to find out their position.

Range free systems can be used by nodes equipped with cheaper radios that are not able to compute the distance of the received signals. In (Bulusu et al., 2000) the nodes compute their position as the centroid of the positions of the anchors that they can receive. This approach requires a high number of anchors. In (Nagpal et al., 2003) and (Niculescu and Nath, 2003b) the anchors flood their position in the network. The number of hops (that is related to communication range) is used to find a node's position.

Virtual coordinate systems. Virtual coordinate systems define a coordinate space that is unrelated to the physical position of the nodes but it is related to other concepts as neighborhoods. In such systems the focus is in providing an efficient support for routing systems and for geographic routing too, with the only difference that the coordinates used in routing are virtual and not real ones.

Multidimensional scaling (MDS) (Shang and Ruml, 2004, Ji and Zha, 2004) is a technique that maps proximity information of the nodes to the element of a matrix: devices have connectivity information (whether or not two devices are in range). The distance between two connected nodes is defined to be 1, while the distance between two nodes not in range is set to the number of hops in the shortest path between them. In this new space the Euclidean distance between nodes is related to the original proximity information. Each node has knowledge of its neighbors at various hops of distance (for energy saving reasons, typically this distance is no more than 2 hops), and it creates a matrix containing the shortest paths between itself and its neighbors. These local matrixes are then merged to produce a global map. The global map can be then used by anchor nodes (which know their position) to give to all the nodes a physical position.

In (Rao et al., 2003), the authors introduce a family of coordinate assignment protocols with increasing complexity suitable for different scenarios. In the first scenario, the coordinate of the nodes in the network are computed, in an iterative fashion, as the averages of the coordinates of the nodes on the external perimeter of the network because these last nodes are the only ones that know their own position. In the second scenario, the nodes on the border do not know their own position and the first step of the algorithm consists in the construction of a *perimeter vector* with the distance from each node of the network with all the nodes on the borders. In the third scenario, the sensors are not aware to be on the border or in the central part of the network, thus two nodes start a bootstrap phase in which the nodes on the border are identified because they are the most distant from the two bootstrapping nodes. All the three systems in (Rao et al., 2003), use a large amount of energy and memory to compute the coordinates of the nodes. Energy is used by the iterative systems that are used to compute the average of the coordinates of the nodes. Memory is used to maintain the perimeter vectors used in such computation.

It is possible to provide less expensive solutions that use only the distance in hops from some of the nodes on the border. In (Cao and Abdelzaher, 2004), a reconfigurable number of nodes are used as anchors. The anchors are chosen randomly,

and the coordinates of each node are the hop distances between the node and the anchors. However, since anchors are selected randomly, some of them might be close to each other, and this may result in an unbalanced coordinate system in which too many nodes share the same coordinates. To overcome this problem, (Caruso et al., 2005) presents a result for uniformly deployed networks. In this solution the size of the areas of nodes sharing the same virtual coordinate is minimized because the anchors are selected as far as possible from each other. To select distant anchors, the authors propose a distributed algorithm that elect three nodes on the border of the network in such a way that their position approximates the tips of a equilateral triangle and then, they compute the virtual coordinates of all the nodes in a distributed fashion. The authors also show that proactive routing protocols, perform similarly with virtual and physical coordinates.

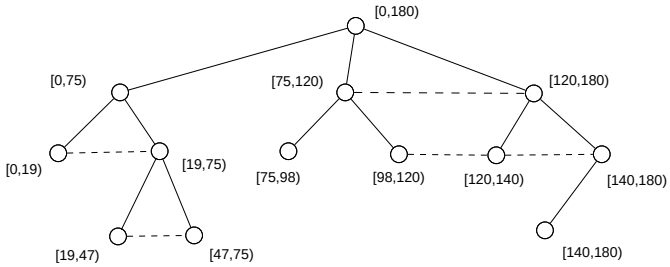


Figure 3.3: A VPCS angle range assignment (Newsome and Song, 2003). The root has range $[0, 180)$*. Each first level neighbor has an assigned range proportional to the number of nodes in its sub tree.*

Virtual Polar Coordinate Space (VPCS) is the virtual coordinate system used by the GEM storage system (Newsome and Song, 2003). In VPCS a ringed tree graph is embedded in the network topology. Each node of the tree has an identifier that is formed by two values: the first one is the level in the tree and the second one is a virtual angle that identifies the node in the level. The virtual angle is assigned in such a way that it is consistent with the network topology. In this way, the coordinates represent a polar system whose origin is a single node (usually the sink). The VPCS algorithm uses several steps to build the polar coordinates. The first step builds the ringed tree (as the one depicted in figure 3.3). To build the tree, the root node assigns to itself the level 0 of the tree. All the nodes that are at one hop distance from the root have level 1. At this point, the nodes at level 1 broadcast the information to all their neighbors. The procedure continues until all the nodes are assigned with a level. After the tree is built, the size of each sub-tree is sent back to the root of the tree in a recursive way starting from the leaves. The nodes at level i collect the information about the size of the sub-trees formed by the nodes of level $i + 1$ or greater. When the root has collected all the sub-tree sizes for each node of level 1 it begins to assign the virtual angles. The root uses for itself the whole angle range, let us say the range $[0,\ 180)$. Then it assigns to each one of its neighbors a

sub-range of its angle. The size of the sub-range is proportional to the size of the sub tree. This system gives to larger sub-trees a wider angle, balancing the coordinate system. Each level 1 node assigns to its neighbors of level 2 a sub range of its range proportional to the size of the sub-trees. This procedure is repeated until the leaves are reached as presented in Figure 3.3, where the root of the tree has the whole virtual range [0, 180) and the three first level neighbors the virtual ranges [0, 75), [75, 120) and [120, 180). This ranges assignment reflects the sizes of the sub-trees rooted into the first level neighbors.

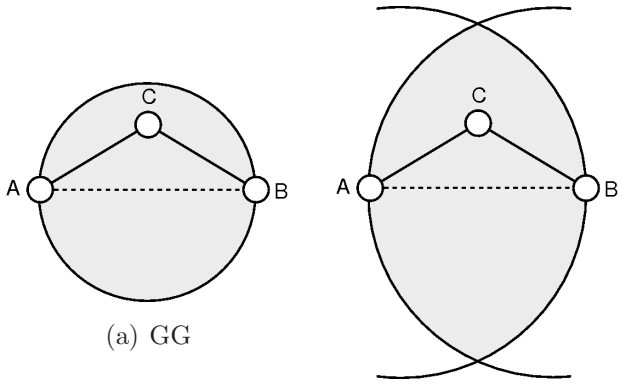
At this point, VPCS aligns the computed topology with the geographical information of the network to provide a more efficient structure. The alignment provides the routing system to jump from one branch of the tree to another using cross-links in the embedded tree structure (which are represented by the dotted lines in Figure 3.3).

To perform the alignment in a distributed fashion, each node must find the *order* of its children. The order is given by the way in which the nodes are mapped on the tree structure, and then, we must be able to identify, on the tree, the nodes on the *left* and the nodes on the *right* of one node. The order is found out using the relative distance between the nodes at a particular level of one sub-tree. To compute such distance the authors first propose to use the signal attenuation of the transmission. However, this system can introduce errors, and for this reason the authors themselves propose an alternative method based on triangulation. The coordinate system based on the triangulation is computed starting from the root node. The root node selects two other nodes in such a way that the three nodes are not collinear and then, using spanning trees from the root and these other two nodes identify the relative position of the other nodes.

3.2.2 Routing systems

Traditional reactive approaches (Johnson et al., 2001, Perkins and Royer, 1999) to routing in ad hoc networks appear unsuitable to wireless sensor networks due to the limitations of the sensors in terms of energy, memory and processing capabilities. For this reason, recent approaches in WSN focus on geographic, stateless routing which does not burden the sensors with routing tables and caches. Among these protocols, we mention GPSR and VPCR, which are the most widely used in WSNs.

Greedy Perimeter Stateless Routing. The Greedy Perimeter Stateless Routing (GPSR) (Karp and Kung, 2000) is the routing protocol used by GHT (Ratnasamy et al., 2003) system to route and store messages in the network. The GPSR protocol uses nodes coordinates to route packets in the network. The key idea in GPSR is that the packet is routed in a *greedy* way, reducing the distance to the receiver at each hop, when possible. When no improvement is possible, the protocol switches to *perimeter* mode that grant to find a closer hop.



(a) GG

(b) RNG

Figure 3.4: Areas used to choose to insert or not an edge inside the planarized graph. (a) depicts the area used by the Gabriel Graph planarization. It is a disk of diameter equal to the distance between A and B and centerd in the middle point with reapect to A and B. (b) depicts the area used by the Relative Neighborhood Graph planarization. It is a lens-shaped area given by the intersection of two disks of radius equal to the distance between A and B and each one of them is centered in A and B

In in perimeter mode the protocol uses the graph planarization of the graph to prevent loops. The graph planarization (Jaromczyk and Toussaint, 1992) is implemented with a distributed algorithm and creates a graph in which there are no crossed edge between the nodes. The planarization used by GPSR is the Gabriel Graph (GG) planarization, but Relative Neighborhood Graph (RNG) planarization can be used too.

In the planarization of a graph an edge is inserted between two nodes, say A and B, if and only if we cannot find a third node C that can connect A and B. The difference between GG and RNG is the size and the definition of the area in which we must look for node C.

As depicted in Figure 3.4.a, to compute the GG of a graph the edge (A, B) is included into the planar graph if and only if the disk with diameter (A, B) does not contain any other nodes of G.

On the other hand, as depicted in Figure 3.4.b, to compute the RNG of a graph G the edge (A, B) is present if and only if the lens defined by the intersection of the two disks of radius (A, B) and centered in A and B does not contain any other nodes of G.

Both the GG and the RNG of a graph G guarantees some property such as connectivity: a GG (or a RNG) of G is connected if G is connected.

The algorithm to compute the GG is described in the following distributed algorithm (Bose et al., 2001):

Algorithm 1 GG(v)

Require: A node v of the graph
Ensure: A GG planarization of v's neighborhood

```
for all u ∈ Neighborhood(v) do
  if disk(u, v) ∩ (Neighborhood(v) \ {u, v}) ≠ ∅ then
    delete(u, v)
  end if
end for
```

In the Algorithm 1, each node v looks in its neighborhood $N(v)$ to find the nodes that are part of the GG: for all the nodes u in its neighborhood it creates the disk of diameter (u, v) and centers it in the middle between u and v and then verifies that no other node in the neighborhood of v is inside the disk. The algorithm complexity is $O(d^2)$, where d is the node degree. Using some additional information, as the Voronoi diagram and Delaunay triangulation (Okabe et al., 1992) the complexity can be reduced to $O(d \log d)$.

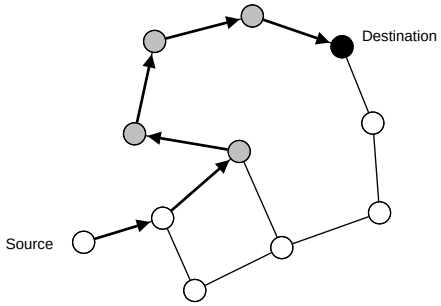


Figure 3.5: Perimeter mode of GPSR protocol. The grey nodes are the ones that belong to the perimeter built to move the message from the source to the destination.

As said before the routing is performed using two different modes, used in two different situations.

Greedy routing is used to move quickly the packet from the source to the destination. The greedy routing works as follows. When a node receives the packet and the node is not the destination of the packet it must forward it to one of its neighbors. The node forwards the message to a node which distance from the destination is less than its own distance and less than the distance of any other node in its neighborhood. If the forwarding node is the closest to the destination between the nodes of its neighborhood and it is not the destination node the routing mode is switched to switch *perimeter* mode (Figure 3.5).

The first step of the *perimeter* routing consists in the planarization of the graph around the forwarding node. Once the planarization is computed, the message is forwarded to the first node of the planarized structure (as depicted in Figure 3.5)

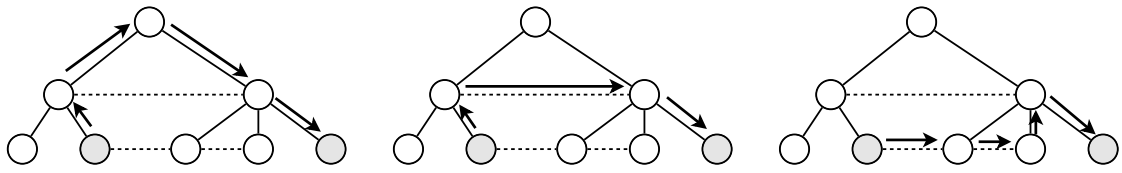
in counterclockwise (or clockwise) direction. If also the receiver node is not able to forward the message in a greedy way, it planarizes its neighborhood and proceeds in perimeter mode too.

The perimeter routing starting from a node v, to arrive to destination d, continue to move on the perimeter until the forwarder node finds out a suitable node to perform greedy routing again. If such node is not found, the packet returns to the node that begun the perimeter mode triggering some user defined action (as we will see in Section 3.4.1, GHT uses this event to start the storage of a datum on such perimeter, or in the case of data retrieval, to signal that the datum is not present inside the network).

Virtual Polar Coordinate Routing. The GEM (Newsome and Song, 2003) system uses the Virtual Polar Coordinate Routing (VPCR) to route messages on its VPCS located network that uses a virtual coordinate system structured as the tree-like structure presented in Figure 3.3. VPCR uses three different routing techniques: the *naive-tree* routing which does not use cross-links (represented by the dotted lines in Figure 3.3), the *smart-tree* routing that introduces optimizations on the naive routing and the *greedy* routing which uses greedy forwarding on cross-links. We describe the three models below.

In the *naive-tree routing*, as depicted in Figure 3.6.a, a packet is forwarded by a node upward to its parent in the tree in a recursive way until the packet reaches the root. However, if along this path the packet reaches an ancestor of the destination node, then the ancestor forwards the packet downward in the tree, directly to the destination. During the journey from the ancestor to the destination two factors are taken in account: the first is that the level must increase at each forwarding and the other is that each node sends the packet in the sub-tree whose angle range contains the virtual angle of the receiver.

In the *smart-tree routing*, as depicted in Figure 3.6.b, each node checks if an ancestor of the destination is in its neighborhood. In this case, the packet is forwarded



(a) naive-tree (b) smart-tree (c) greedy

Figure 3.6: VPCR routing techniques: (a) the naive-tree routing *that forwards messages to a common ancestor of both sender and receiver, (b)* smart-tree routing *that forwards the messages to an ancestor of the destination as soon as the protocol finds such ancestor and (c)* greedy routing *that forwards the message to nodes closer to the destination without using ancestors.*

directly to that ancestor. In some cases, this may save the cost of reaching a common ancestor in the tree. This method can be improved using 2-hop neighborhoods. In this case, if the ancestor is as far as two hops the packet is directly forwarded to the ancestor using the connecting neighbor. The method can be generalized using a proactive area of n-hop neighborhoods. However, as n grows, the cost in terms of space (required to store the proactive area) and messages (to keep updated this information) also grows, and this approach results impractical even for limited values of n. In general, the smart-tree optimization reduces costs if the source and the destination are close to each other, but it does not improve significantly the efficiency for arbitrary pairs of nodes.

Finally, *greedy routing*, as depicted in Figure 3.6.c, can be used effectively when the source and the destination are far away from each other. In the ringed tree's rings, the virtual angles are assigned in strictly increasing, or decreasing, order. When the smart-tree routing is forced to route the message upward in the tree the forwarder node checks if some of its neighbors' virtual angle and if one of them is closer to the destination's angle range than its own, the node greedily forwards the message to the neighbor which is closer to the destination. If the message reaches a node that is no more able do greedily forward the massage , the routing algorithm switches to the smart-tree routing. If smart-tree routing is also impossible, it switches to naive-tree routing. However, the routing, when possible, can be switched back to greedy or smart tree routing. This is the case depicted in Figure 3.6.c, in which the message is greedily forwarded for the first two steps and then, when greedy routing is no more possible, it is forwarded upward in the tree using naive-tree routing.

3.2.3 Redundancy systems

Redundancy models are used to assure data availability, i.e. the system ensures that the stored data are still available despite of sensors faults. The fundamental redundancy technique used in sensor networks is the *pure replication*.

Pure replication is the simplest way to achieve data availability. It ensures that a datum is available until all the sensors storing a copy of that datum fail. In a storage system, pure replication can be used in an *uncontrolled* or *controlled* way.

Uncontrolled replication. In uncontrolled replication, the number of replicas of a datum is not known a priori. This system is widely used in data centric storage systems such as GHT (Ratnasamy et al., 2003), that we detail in Section 3.4. The general idea behind uncontrolled replication is the following. Once a location, or a node, is found to store the datum this last one is stored in the *neighborhood* of such location, or node. Since this neighborhood can be arbitrary large, it is not possible to determine the number of replicas which will be stored in. In some cases, this leads to excessive replication while in other cases it results in a very poor replication of a datum. Moreover, the nodes in which each datum is replicated can be far from the

intended location for the datum and after some faults the datum is unreachable, also if it is still present inside the network. This high variability, for both the number of replicas and their distance from the intended datum location, cannot guarantee that the datum will be available for a long time, or for an expected time. This can also produce bring to states of unfair load distribution in the network.

Controlled replication. On the other hand, controlled replication systems (Albano et al., 2007) are based on the concept of Quality of Service (QoS). Each datum has an associated QoS value. This value can also be the same for all the nodes. The number of copies of each datum is a function of its QoS level. Let us suppose a datum d needs $n = QoS(d)$ copies. A system using controlled replication will find n nodes according to some data distribution rules that will store a copy of the datum. In this way, each datum can be stored in a predictable number of nodes and both the reliability and load distribution are acquired.

3.3 Database model

The database model enables the user of the network to abstract a WSN as a database. Using this abstraction, the user can perform queries that program the sensors to retrieve and refine data.

This approach provides a network abstraction which is completely independent of the network details, the sensors do not need to be preprogrammed for specific tasks and the user can easily change the behavior of the network by injecting new queries.

Specifically programmed applications can be more efficient, but they are very specialized and they are tuned only for a given task. These applications must be programmed from scratch each time a new query is needed. In particular when the query changes, the design, coding and debugging steps must be performed again. On the other hand, in a database abstraction the system is always the same and only the queries change to perform different tasks with only little design, code and debug efforts.

3.3.1 TinyDB

TinyDB (Madden et al., 2003, Madden et al., 2002a) is a WSN database implementation developed at UC Berkeley. TinyDB provides a SQL-like language extended to use both duration and sample ratings, to provide averages on time spans. The database is able to run queries over a single table that contains all the data collected by the network. Each sensor is represented by a row of the table and continuously updates its own data. TinyDB supports a large set of operators (Madden et al., 2002a): spatial aggregation, filtering based on patterns and union between row sub-sets. An example of TinyDB query is the following one (Madden et al., 2003):

Example 1 Query example in TinyDB.

```
SELECT nodeid, light, temp
FROM sensors
SAMPLE INTERVAL 1s FOR 10s
```

This query specifies that each sensor of the WSN, that is abstracted as the virtual table *sensors*, must report its id, light and temperature readings (*nodeid*, *light* and *temp*) once per second for 10 seconds. The results of the query are sent to the requester as a stream of records.

TinyDB provides power aware optimizations of the queries and the execution plan of the query is performed on the basis of the type of datum (its meta-data) and of the parameters needed by the operators. Then, the operators are scheduled to provide a suitable environmental sampling to get all the data needed to compute the query. The query is distributed using Semantic Routing Trees (SRTs). SRTs are routing trees rooted at the sink. A SRT reaches all nodes needed to cover the attributes of interests for a given query. In general, it may cover the entire network.

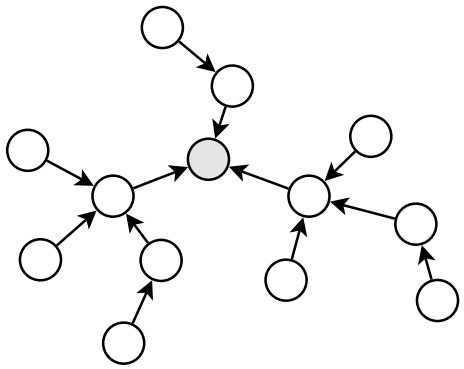


Figure 3.7: SRT collecting data from the whole network.

The SRT depicted in Figure 3.7 is a possible instance of SRT for the query presented in Example 1. The SRT spans the whole network and the data, composed by the tuples $\langle nodeid, light, temp \rangle$, flow to the sink (the gray node) from each sensor (the white nodes), each second for 10 seconds.

Although SRTs provide very efficient routing, their use forces a limitation of TinyDB because the queries can only go from the sink to the leaves. This is an obstacle to more complex queries that involve more complex communication patterns. For instance, this prevents comparison of data produced in different subtrees of the SRT. Also, for this reason, TinyDB is a parallel (but not distributed) query processor that does not perform in-network joins. All the information flow to the sink and the main operators that are supported by the in-network query processor are selection and the union of the streams. However, joins can be performed only at the sink.

3.3.2 Cougar

Cougar (Bonnet et al., 2000, Bonnet et al., 2001, Yao and Gehrke, 2002) is a WSN database system developed at Cornell University and exhibits many similarities with TinyDB. The query language is a SQL dialect in which the nodes of the network are represented by Abstract Data Types (ADTs) with interface methods (as in object oriented programming) to retrieve data stored in sensors.

An example of Cougar query is the following one (with similar semantics of the one presented for TinyDB):

Example 2 Query example in Cougar.

```
SELECT R.s.getNodeID(), R.s.getLight(), R.s.getTemp()
FROM R
WHERE $every(1);
```

This query specifies that each sensor of the WSN must report its id, light and temperature readings (in ADT notation) once per second (`$every(1)`).

In Cougar's queries the FROM clause can refer to a WSN's relation. The relation includes nodes attributes and nodes ADTs. In Example 2, the FROM clause refers to all the nodes that are collected in a set of nodes called `R`.

The SELECT and WHERE clauses can refer to specific node's data and can use methods defined in the node's ADT. For instance, in Example 2, the temperature of a sensor `s` in the relation `R` is denoted by the method invocation `R.s.getTemp()`.

The query optimizer is run on a workstation and generates the query execution plan that specifies communication patterns, computational activities and tree operations as joins. For each method invoked on ADTs in the query, a *virtual relation* is created. A virtual relation is a tabular representation of a method that contains the input values and the output argument of the method it is associated with. Following Example 2, the method `getTemp` is associated with all the sensors `s` in the set of sensors `R`, it has no input and returns a value of some suitable type, for instance a `float` value to represent the sampled temperature.

3.3.3 MaD-WiSe

MaD-WiSe (Amato et al., 2005a, Amato et al., 2006a) is a recent database model for WSN. Differently from TinyDB and Cougar, its query processor is fully distributed, so that any query can be completely processed inside the network. MaD-WiSe defines an SQL-like query language, a query algebra, and a data model based on streams.

An example of MaD-WiSe query is the following:

Example 3 Query example in Mad-WiSe.

```
SELECT roomB.Temperature
FROM roomA, roomB
WHERE roomA.Temperature > roomB.Temperature and
      roomA.Temperature > 50
EVERY 10 seconds
```

The query specifies that the temperature of the sensors placed in two locations, called `roomA` and `roomB`, must be compared and only when the condition in the WHERE clause is true, it outputs the `Temperature` in `roomB`. The sampling of the temperatures is performed every 10 seconds.

MaD-WiSe defines heuristics for query optimization which take into account the transducer sampling costs, predicate selectivity and transmission costs. In the optimized query plan, each sensor involved in the query is assigned with a subset of the operators. For this reason, the communication pattern between sensors involved in a query can be arbitrary, and it requires general routing strategies.

The MaD-WiSe architecture comprises a user-side module and a network-side module. The user-side module implements the query parser, the query optimizer, and the tools to inject queries and collect results. The network-side module implements a layered architecture with a *network* layer, a *stream system* layer and a *query processor* layer.

The *network layer* implements both connection oriented and connectionless services as well as a general point-to-point routing. It also embeds an energy efficiency module which governs the duty cycle of the sensors.

The *stream system* layer (Amato et al., 2006b) provides a stream abstraction to the query processor. The streams abstract the transducers, the remote communications between nodes and the local data exchange between operators allocated on the same node, providing a unifying view of all these sources for the underlying query processor that can use them in the same way.

The *query processor* layer implements all the operators of the algebra, which include selections, projections, spatial and temporal aggregates as well as unions and joins. Note that the ability to perform joins within the network is unique to MaD-WiSe and allows comparison of data from different source nodes in the network itself without the need of collecting all the data on the sink before the processing.

3.4 Data Centric Storage model

Storage systems can be used either with simple data acquisition mechanisms, or in combination with more sophisticate models such as databases. DCSs provide support

for efficient storage of sensed/processed data and for data retrieval, and they exploit data redundancy to ensure data reliability.

DCS addresses all the problems concerning the topology changes due to node failures, energy efficiency and fair load distribution using Geographic Hash Tables (GHT) (Ratnasamy et al., 2003), which are based on physical coordinates of the sensors. Other approaches to DCS, such as Graph EMbedding (GEM) (Newsome and Song, 2003), use virtual coordinates instead of real ones. Cell Hash Routing (CHR) (Araujo et al., 2005), is another DCS system based on hash tables. CHR clusters nodes in cells of predefined and globally known shape using a distributed protocol (e.g. dividing the sensor field in a mesh of squares) and stores each data in a cell. Moreover, K-D tree based Data-Centric Storage (KDDCS) (Aly et al., 2006) is an in-network data storage that addresses the problem of load distribution using a distributed K-D tree (Bentley, 1975).

3.4.1 Geographic Hash Tables

A Geographic Hash Table (GHT) (Ratnasamy et al., 2003) provides a $\langle key, value \rangle$ associative memory abstraction of the sensor network. Data are represented by $\langle key, value \rangle$ pairs and GHT offers two operations: `put` and `get`. `put(k,v)` stores the value v according to a geographic position (x, y) that is computed using the key k. `get(k)` retrieves the value v that was stored using the the key k looking in the position (x, y) computed using k.

The central part of GHT is a deterministic and uniform hash function that enables both `put` and `get` to find the same geographical location starting from the same key k. The hash of the key k ($h(k)$) is a point (x, y) of the WSN deployment area: a $\langle key, value \rangle$ pair is stored on the nodes that are closer to the point (x, y). To this purpose GHT exploits GPSR (Karp and Kung, 2000) to route a pair $\langle key, value \rangle$ towards the point $(x, y) = h(key)$.

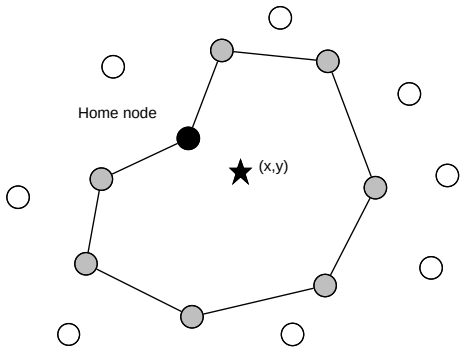


Figure 3.8: The point (x, y) (the star), the home node (the circle in black) and the home perimeter (the circles in gray).

In general, (x, y) is a generic coordinate in the deployment area and, with high probability, does not match with any real node, as for the case depicted in Figure 3.8, where the nodes of the network are the circular points and the point (x, y) is represented by a star. To deal with this problem, GHT uses the concept of *home node*. The home node of (x, y) is defined as the node that is closest to the point (x, y), and it is the rendez-vous point for both `put` and `get` operations. In Figure 3.8 the home node is represented by the black colored node, which is the closest one to the point (x, y).

GHT finds the home node of (x, y) using the perimeter mode of the GPSR protocol. Once a packet arrives in the proximity of the point (x, y), it is routed around that point. At the end of the perimeter mode the packet returns at the starting node of the perimeter mode because there is no other way to reach point (x, y). This event triggers the *home perimeter* building procedure that replicates the pair $\langle key, value \rangle$ on all the nodes that surround the point (x, y). The sensor that is closest to (x, y) becomes the home node. In Figure 3.8, the nodes that belong to the perimeter and that store the datum, as well as the home node, are the gray colored nodes.

GHT provides a DCS system that is scalable and robust because in a WSN, which nature is unreliable, these two characteristics are fundamental. GHT features *data persistence*, *data consistency*, *load distribution*, *database increase* and *topological generality*. Data persistence means that a pair $\langle key, value \rangle$ that is stored in the system is available to queries, despite of sensors failures. Data consistency means that a query for a key k, is correctly routed to a node that hosts the right pair $\langle key, value \rangle$. Load distribution means that the pairs $\langle key, value \rangle$ are stored in fair way such that data is not concentrated in any particular node. Database increase means that when new nodes are added to the system data is spread also on the new nodes. Topological generality means that the system does not need a particular topology to work properly.

To the purpose of *data persistence* and *data consistency* all the nodes in the home perimeter store the pair $\langle key, value \rangle$. However, when one or more topology changes of the network occur, a home node can disappear or home perimeters can be broken. In this case, data consistency can fail because a request for a datum can be taken to the new closest node to the point (x, y), that is neither the home node nor part of the home perimeter. For this reason, GHT implements the *Perimeter Refresh Protocol* (PRP). PRP periodically generates refresh packets. Each t seconds, the home node performs a `put` operation to the (x, y) point. The packet enters in perimeter mode and refreshes the data on the home perimeter. If the perimeter is broken, due to some node failure, a new perimeter is built. In the same way, if new nodes are deployed in the network, the PRP is able to compute a new home node and a new home perimeter using both the new and the old nodes.

Each time that a node receives a refresh packet it stores the data again and sets a timer, to be notified after t seconds. This operation is necessary because, if no refresh messages arrive after t seconds, the node itself starts the refresh procedure. This ensures that a home node failure is recovered as soon as t seconds expire.

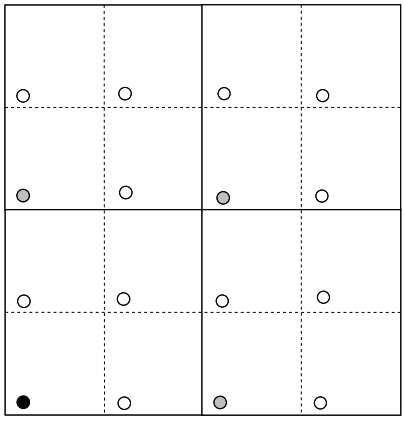


Figure 3.9: Structured Replication (SR). The black node is the original hashed coordinate, the grey nodes are the level one copies and the white nodes are the level two copies.

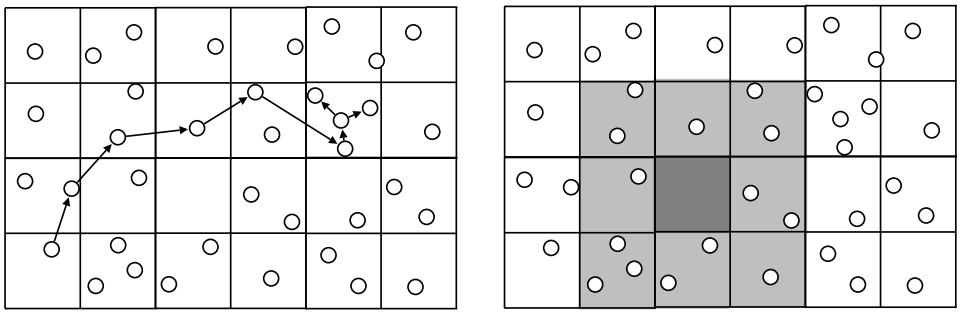
In GHT, if a pair $\langle key, value \rangle$ is very popular the nodes that store that value, and the routes to them are heavily stressed. To resolve this scaling problem, GHT uses the *structured replication* (SR) strategy (Figure 3.9). SR uses space decomposition. The plane is divided into a square grid and groups of contiguous areas are grouped in larger squares in a hierarchical fashion. The hierarchy is represented as a tree. The root of this tree is the whole area and the leaves are the smaller squares. For instance, the hierarchical decomposition depicted in Figure 3.9 shows a sensing area divided in 16 squares. The squares are aggregated in 4 macro-squares (solid lines). With SR a pair $\langle key, value \rangle$ is still stored only in one position, this position is not $(x, y) = h(key)$, but can be one of the in the $4^d - 1$ *mirror points* of the location (x, y). The point (x, y) is located in a squared sub-area of the grid and occupies a relative position in it, let us call it (x', y'). A mirror point is a point, belonging to a different sub-area, that has the same relative coordinate (x', y') of the point (x, y). In Figure 3.9 the black node represents the hashed position, the gray nodes are the mirrors of the second level and the white nodes are the mirrors of the third level. A node that performs a `put` operation thus stores the data on a mirror node, and the mirror, informs all the ancestor mirrors about the actual position of the data. The `get` operation starts querying the root mirror, which propagates to its child mirrors the request, until the request reaches the mirror actually storing the data, that is than returned to the querying node.

The SR does not replicates the actual datum. SR replicates the *pointers* to the actual position of the datum in a hierarchical way.

3.4.2 Cell Hash Routing

The Cell Hash Routing (CHR) (Araujo et al., 2005) is an evolution of GHT. CHR divides the space in *cells* of equal size. Cells are squared and divide the space in

a grid. Each cell defines a *cluster* of sensor nodes. The choice of the size of the cells is based on the communication range of the nodes. Each node in a cell must be able to receive messages from all the nodes in its cell and from the nodes in all the adjacent cells. With this restriction CHR guarantees that, if the initial network was connected, the graph interconnecting the cells is connected too. Let r be the communication range of a node of the network. To enable the node to communicate with all the nodes in its cell and with the eight adjacent cells, CHR requires that the side of a cell is at most $r/\sqrt{8}$. CHR uses cells to solve the problem of node mobility, that was not addressed by GHT. A cell is considered a super-node that stores some data. If a node moves into a cell, the other nodes inside the cell replicate the data in the newcomer's memory.



(a) Greedy routing on cells (b) Cells home-perimeter

Figure 3.10: CHR cell division of the sensors space. (a) depicts the greedy routing on the cell structure: the message is forwarded from one cell to another and when it arrives to the destination cell it is copied on all the nodes belonging to the cluster. (b) depict the case in which home cell is empty (the dark gray cell) and the data must be stored on the home perimeter of the home cell (the light gray cells).

CHR uses a modified version of the GPSR routing protocol to route messages from one node to another. The modified version of GPSR works on cells and not on single nodes. As depicted in Figure 3.10.a, to route a message from one cell to another one, the modified GPSR routing protocol forwards the message from one cell to the following one using a single node per cell and, when the destination cell receives the message, the receiver node copies the message in all the nodes belonging to the cell.

The choice of CHR, to use of small cells, guarantees that a node in a cell communicates with all the nodes inside its cell and in the surrounding cells. The routing protocol takes advantage of this because (i) the routing protocol is able to know if the cells around the actual node are empty and if it is no more possible to forward the message in a wanted direction and (ii) the next node, during routing from a cell to the following one, is found using a simple randomization scheme between all the nodes in the next cell.

The cell structure itself provides two major benefits to geographic routing (in a more general fashion) because the geographic routing performs better in low density networks: each hop covers a larger space and the clustered network is a less dense version of the underlying network because each cell may contain more nodes.

Moreover, geographic routing performs better in networks which have a smaller number of edges and enables a simpler planarization during perimeter mode. Also in this case the clustered network is a good choice because each cell is connected with all the neighbor cells.

CHR manages data storing it in a whole cluster, that is all the nodes belonging to a cell. To find out the cell that will store a datum (that is called the *home cell*), CHR uses a hash function to return the position where the datum must be stored to (or retrieved from). The hash function used by CHR, differently from the GHT hash function, does not return a point (x, y) inside the space defined by the network. The CHR hash function returns a *cell index*, that refers to one of the cells in which CHR divided the space.

In the case in which the hashed cell index points to an empty cell, as for the case depicted in Figure 3.10.b, in which the home cell is the one in darker gray, using the GPSR perimeter mode, the messages are routed to the home cell. At this point, the messages are routed around the perimeter of the empty cell and data are stored in all the cells that belong to the *home perimeter* (the cells in light gray in Figure 3.10.b).

3.4.3 Graph EMbedding

Graph EMbedding (GEM) (Newsome and Song, 2003) is an infrastructure for data centric storage that does not need to use of geographical coordinates. GEM is based on VPCS and the routing protocol used on this coordinate system is the VPCR.

In GEM, all the nodes have a label that enables them (i) to route the messages from one node the other and (ii) to map data names to the existing labels. The nodes are labeled using the VPCS algorithm, presented in Section 3.2.2.

In GEM, data centric storage is enabled using a hash function that maps the pair $\langle key, value \rangle$ to the embedded graph's labels. The system uses a function $h(key)$ to achieve this result. The function $h(key)$ depends only on the parameter passed enabling a consistent output for all the possible senders.

The node-to-node communication is enabled using a lookup mechanism based on data centric storage itself. If a node x wants to communicate with a node y, it needs to know the label of y. Let us call such label $L(y)$. At network setup, a node y computes both $L(y)$ and $h(y)$. Then it routes the message $\langle y, L(y) \rangle$ to the node with label $h(y)$. When node x wants to communicate with y it sends a message to the node $h(y)$ to query the value $L(y)$. After this step the node x can send the message to y routing it to $L(y)$.

With GEM, the user can easily add new nodes and the network is able to reconfigure to use the newcomers. When a new node enters in the network, GEM uses

VPCS to assign the new node a level and an angle. In particular, a new node selects a parent node in its neighborhood and the assigned level is the level of the parent plus one. At this point, the parent removes part of the angle range from one of its children and assigns it to the new node. This change affects all the child of the node whose range was decreased to keep the whole tree consistent.

GEM enables the user to add nodes when the network operates. Adding nodes to the network can be used to substitute broken nodes inside the network. When a new node is added to the system, the new node chooses a parent from its set of neighbors. The parent assigns the new node a level (the parent's level plus one) and an angle range taken away from one of the children nodes. The new node can become the new parent of previously disconnected nodes. In this case, the previously disconnected nodes start again the procedure to acquire a level and an angle from the new parent.

3.4.4 K-D tree based Data-Centric Storage

The K-D tree based Data-Centric Storage (KDDCS) (Aly et al., 2006) is an in-network data storage that addresses the problem of load distribution. The authors claim that non-uniform sensor readings, due to the presence of information *hot spots* inside the network (such as nodes more frequently queries than others), can bring to states of unfair load distribution in both storage and queries execution features. The general idea of KDDCS is that data are stored using a distributed K-D tree (Bentley, 1975). A K-D tree is a data structure which allows logarithmic storage and retrieval of data that are identified by geographical coordinates. The K-D tree uses the coordinates to build a balanced tree and to guarantee poly-log access. In KDDCS, the K-D tree is used to provide virtual coordinates to the nodes. A virtual coordinate is a bit string of the form 11001 in which each one of the bits represents the branch of the K-D tree in which the nodes is located.

KDDCS builds the K-D tree in a distributed fashion. In particular, it builds the levels of the tree using partition lines which are constructed by a *weighted split median* algorithm. This algorithm divides the area in regions that contain the same number of sensors. The weighted split median algorithm is built on top of a distributed *Breadth First Search* (BFS) algorithm (Cormen et al., 1990). A node is first elected root of the BFS algorithm in a distributed way.

The election is done independently by each sensor using a probability to be the root of the BFS. If more than one node becomes a root node, each node starts to build its own tree. To overcome the problem of having multiple trees, at the end only the tree built from the root with higher id is kept.

Once the tree is built, the root proceeds in the retrieval of the coordinates of the nodes in the network, computes the median value, and communicates it to the other sensors.

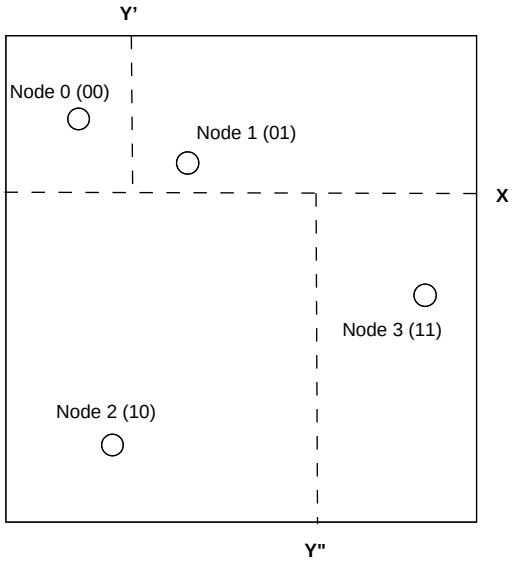


Figure 3.11: K-D tree built on top of a four nodes network. In the first step, the area is first divided in two using line X, in the second step each resulting sub-area of the first division is divided using the lines Y' and Y''. (The last step is performed in parallel on each sub-area)

The algorithm partitions recursively the network, dividing the sensor set in two balanced parts using alternatively the x and the y values. A result of such partitioning is depicted in Figure 3.11.

The algorithm starts partitioning the whole area horizontally. Once the median value is computed and spread on the network, the sensors with y coordinate lesser than the median value assign to themselves the coordinate 0, and the sensors with y coordinate larger than the median value assign to themselves the coordinate 1. At the second step the weighted split median algorithm is applied locally in each sub-region. The sensors with x coordinate lesser than the median value left-shift to their coordinate the value 0, the sensors with x coordinate greater than the median value left-shift to their coordinate the value 1. The algorithm continues to divide, alternatively on the x and the y values, until each node has an unique id.

The tree that is built using this procedure is balanced because at each step the sensors set is split in two sub-sets having half of the nodes each.

Following the previous algorithm, Figure 3.11 shows an example of network that is divided in a K-D tree structure: the first step of the algorithm divides the plane in two halves following line X, thus both $Node$ 0 and $Node$ 1 have the first bit of their address set to 0 and both $Node$ 2 and $Node$ 3 have the first bit of their address set to 1. The second step divides each one of the previous halves in two, following line Y' and Y'' respectively. This last step sets the second bit of the address.

The logical coordinates built using the K-D tree are used to balance the load of

the network. To provide fair load distribution the system uses a mapping between the events that must be notified and this coordinate system. The mapping requires the events' probability distribution. This distribution is computed during the network lifetime and the tree is adjusted to keep the load balanced and to pace with the current distribution. At the beginning the network assumes that the events' generation probability is uniform. The mapping assigns the coordinates to specific event ranges. Consider, for instance, a network that must control temperature. KDCCS initially guesses a possible range of values for the temperature and this range is mapped on the the virtual coordinates: when the situation evolves, and the actual range is discovered and the mapping is corrected to balance the load again.

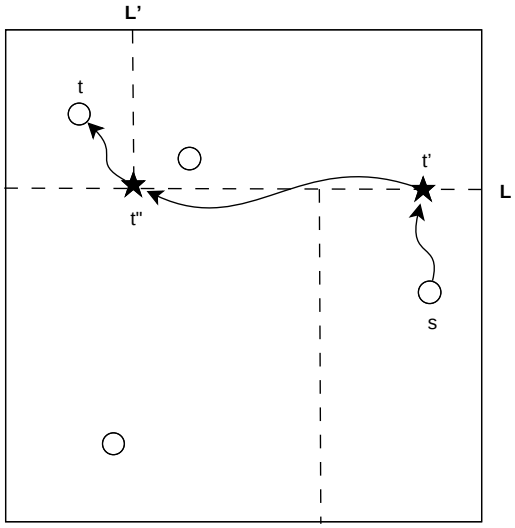


Figure 3.12: LSR routing from node s to node t using points t' and t'' belonging to L and L' respectively.

KDDCS uses a routing protocol, defined into the paper itself, called *Logical Stateless Routing* (LSR) to route the messages on the network. LSR uses the tree structure of the network to perform GPSR routing in multiple *rounds*. LSR uses $\log n$ rounds to route a message from one sensor to another. A sensor s that wants to send a message to a node t must identify the least common ancestor (LCA) of both itself and the receiver. Let R be the region that contains the LCA and let L be the line that splits R in the two sub-regions containing s and t. s knows the position (only one component) of the line L because part of its address belong to this line. The message is routed from s to the point t' of line L that is perpendicular to that line using GPSR. Once arrived at the node closer to the point t', the procedure is repeated, and the message is forwarded to a point t'' belonging to L' that is the LCA of t' and t. Figure 3.12 shows an example of routing performed by LSR. To send a packet to node t, node s, must follow the tree structure and must use the points t' and t'' to route the message. Both t' and t'' are points belonging to the division lines

that where used to divide the network.

3.5 Summary

In this chapter, we have presented a review of the principal topics and results in data management for WSNs.

Early WSNs were able only to perform data acquisition and to send raw data streams, without processing, to a special node (the sink) which task was to collect these streams and provide them to the user. More recent data management strategies use the network itself to provide data computation. In this way, the user is able to program the network as a database. The network organizes itself to manage sensor-to-sensor communications to compute more complex functions and to enable in-network data storage.

At the beginning of this chapter, we provided a brief introduction to the models of data management systems in WSNs. We actually identified four models: *external storage*, *directed diffusion*, *data centric storage*, and *database models*.

Then, we moved into the description of the building blocks that are essential in the construction of an effective and efficient data management system. First of all, we described the localization systems (Savvides et al., 2001, Nasipuri and Li, 2002, Niculescu and Nath, 2003b, Niculescu and Nath, 2003a, Bulusu et al., 2000, Nagpal et al., 2003, Niculescu and Nath, 2003b, Shang and Ruml, 2004, Ji and Zha, 2004, Rao et al., 2003, Caruso et al., 2005, Cao and Abdelzaher, 2004, Newsome and Song, 2003) that are in use to provide a coordinate system to the whole network and to the sensed data. Then, we described the routing systems (Karp and Kung, 2000, Bose et al., 2001, Newsome and Song, 2003) that are essential for the reliable and efficient transport of messages to both communicate data and to organize the infrastructure of data management. Finally, we described the redundancy mechanisms (Ratnasamy et al., 2003, Albano et al., 2007) to guarantee a more reliable storage.

After that, we described the database model (Madden et al., 2003, Madden et al., 2002a, Bonnet et al., 2000, Bonnet et al., 2001, Yao and Gehrke, 2002, Amato et al., 2005a, Amato et al., 2006a) for data management into WSNs. The database model enables the user of the network to abstract a WSN as a database. Using this abstraction, the user can perform queries that program the sensors to retrieve and refine data. Then, we pointed out the two faces of the database model. The first face shows that the database model is easy to use i.e., the ability to change the network behavior via queries redefinition without the need of (re)programming the network. The second face shows the limited efficiency of the system itself i.e., the cost of the flexibility must be paid in terms of complex operations executed by the network. Some database systems try to limit the inefficiency (e.g., disabling the in-network capability to perform *join* operations in a distributed fashion) at the cost of lowering of the flexibility of the systems themselves.

Finally, at the end of this chapter, we focused on the *data centric storage* which enables the network to work as a data storage for the data that are collected by the sensors. This use of the network is suitable for applications in which the sink node can be unavailable for long periods of time and between such accesses the network must store the data in a reliable way. We presented different protocols that implement such a model, namely *Geographic Hash Tables* (Ratnasamy et al., 2003), *Cell Hash Routing* (Araujo et al., 2005), *Graph EMbedding* (Newsome and Song, 2003), and *K-D tree based Data-Centric Storage* (Aly et al., 2006). All these systems are based on the capability to associate a datum to a specific location inside the network. The value used to locate a datum inside the network is given by its associated meta-datum.

Still focusing on the DCS, all the systems that we have seen until now do not consider the problem of the non-uniformity in WSNs. The only solution that mentions non-uniformity is KDDCS, in which the non-uniformity is seen as the presence of *hot spots* inside the network.

In the next chapter, we will focus on the problem of DCS in non-uniform WSNs. Our approach to the design of a DCS system that can deal with non-uniformity can be summarized as follows: we want to provide a system starting from the assumption that the network is non-uniform. Such system must work also in uniform network, without any modification and the cost of dealing with non-uniformity must be as small as possible.

To define our non-uniformity resistant system, we start from GHT (that, as we will see, is able to work properly only in uniform networks): we analyze GHT finding the non-uniformity weaknesses and than we rethink it from scratch focusing on dealing with non-uniformity. This process will bring us to a system (Q-NiGHT) that has two major features: (*i*) it is able to deal with non-uniformity and (*ii*) it is able to provide a high level of fault-tolerance, using QoS associated to the data.

Apart from providing a non-uniformity proof solution to the DCS problem, we intend to use Q-NiGHT as a Trojan-horse to breakthrough into the structure of the non-uniformity in WSNs. To provide a solution that is able to deal with non-uniformity, we *need* to know how non-uniformity *can* influence the behavior of a WSN and we *must* provide countermeasures to block such influence *without* loosing efficiency.

From this point of view, the design of a non-uniformity proof system is similar to a chess game between the designer and the non-uniformity: a player need to prevent the winning moves of the other player without loosing the opportunity to make the good move to win. The designer need to prevent the influence of the non-uniformity without loosing too much the opportunity to provide an efficient protocol.

Chapter 4

Q-NiGHT: Non-uniformity Aware Data Management

Resistance does not start with big words
But with small deeds [...]
Asking yourself a question
And then asking that question to others
That is how resistance starts.
- Remco Campert

Abstract

The data management in WSNs, as presented in the last chapter, is a wide topic and it is made up of different approaches and techniques. In this thesis, we choose focus on the data-centric storage model. Our main contribution in this area is Q-NiGHT. Q-NiGHT originated by the analysis of the experimental results that we performed on GHT in non-uniform sensor networks. These results pointed out the inability of GHT to provide good results in non-uniformly distributed WSNs. In this chapter, we revise Q-NiGHT using the following approach. We start with the analysis of GHT in both uniformly and non-uniformly distributed networks. Our aim is to point out that GHT is unable to provide good load balance in uniform networks and things get worse in non-uniform environments. GHT's problems are due to the poor control of the perimeters where the data is stored on and the lack of flexibility of the hash function used to provide the coordinates of the point used to store data. Then, we move into the the study of our improvements of GHT to provide a better load distribution. This task is achieved using two different techniques. The first one dismisses the use of perimeters to store data and the second one uses a new, adaptive, hash function to deal with non-uniformity.

In this chapter, we present Q-NiGHT*, our approach to non-uniformity aware data-management in WSNs.

In the first part of this chapter, we analyze the characteristics of the GHT system described in Chapter 3.4.1, simulating its behavior with uniform, Gaussian and Hill distributions of sensors.

Results show the pathological problems of GHT in non-uniform networks i.e., the lack of fair load distribution in non-uniform networks due to the *religious* use of an uniform hash functions for the data distribution and the lack of control on fault tolerance issues due to the uncontrolled replication of data used by GHT.

Then we describe Q-NiGHT. We present our solution to the problem of the lack of fair load distribution with the introduction of a new class of hash functions. Then we present our solution to the problem of the lack of control of perimeter length with the introduction of a better strategy to distribute data.

In this chapter, we do not cover the problem of how the network distribution is found out by Q-NiGHT. We leave the description of this problem to Chapter 5, in which, we present a suite of protocols to find out networks distribution in a efficient way. These protocols are independent from Q-NiGHT and can be used in any distribution-aware sensor system.

This chapter notation. In this chapter we use a recurrent notation to identify the *actors* of our research (i.e., sensors, their number, communication range and so on). Theses symbols are used across all the chapter with the meaning defined in Table 4.1. In the text sometimes we redefine some of these symbols and/or use them with a slightly different meaning. In all these cases, we make the reader aware of such changes.

s s_i s_j	generic sensors
n	number of sensors in the network
r	communication range of a sensor
A	deployment area in which sensors are located
f	geographical distribution of sensors in A
h	hash function used to locate data
D	datum to be stored/retrieved, in the system
M	name (meta-data) for D

Table 4.1: Recurrent symbols

* The name Q-NiGHT stands for "Quality of service in Non-uniform Geographic Hash Tables" and it is pronounced as "knight".

A note about simulations. In this chapter, we make a large use of simulations to (*i*) study the influence of non-uniformity on GHT and (*ii*) study the performance and effectiveness of Q-NiGHT. Our simulation environment is a self-made simulator that is able to deal with large quantity of nodes. All the experiments are repeated until we achieve a statistical confidence of 95%, with a precision within 5%.

Chapter organization. This chapter is organized as follows. Section 4.1 presents the weaknesses of GHT, that motivate our work. Section 4.2 describes our main contribution to data management in non-uniform networks: Q-NiGHT (published in (Albano et al., 2006a, Albano et al., 2006b, Albano et al., 2007)). Section 4.3 describes an application of Q-NiGHT in the field of resource localization inside an heterogeneous WSN (published in (Nidito et al., 2007)). Finally, Section 4.4 draws the conclusions of the research work described in this chapter.

4.1 Why do we need Q-NiGHT?

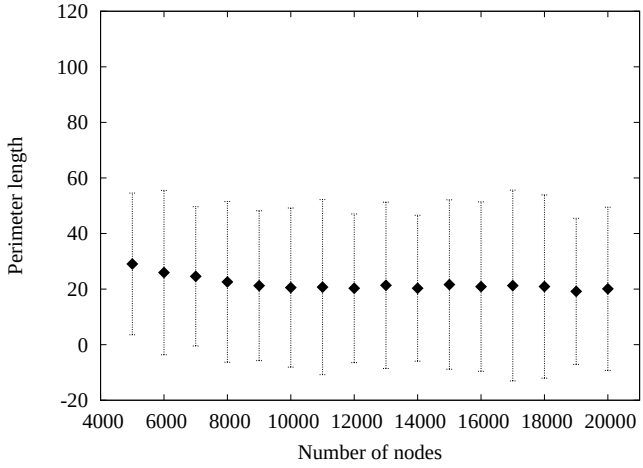
GHT (Ratnasamy et al., 2003) implements Data Centric Storage using Geographic Hash Tables and the GPSR protocol. We have already discussed how the protocol works in Chapter 3.4.1. Here, we discuss the behavior of GHT, with respect to load distribution. In particular, we describe a simulation experiment that gives better insights on the protocol behavior and settles some motivations for Q-NiGHT.

The first thing that we want to measure is the mean and the variance of the perimeters found by GHT. As GHT stores a datum on all the perimeter surrounding the corresponding hash coordinates, this measure gives us an idea on the number of copies of each datum stored by GHT.

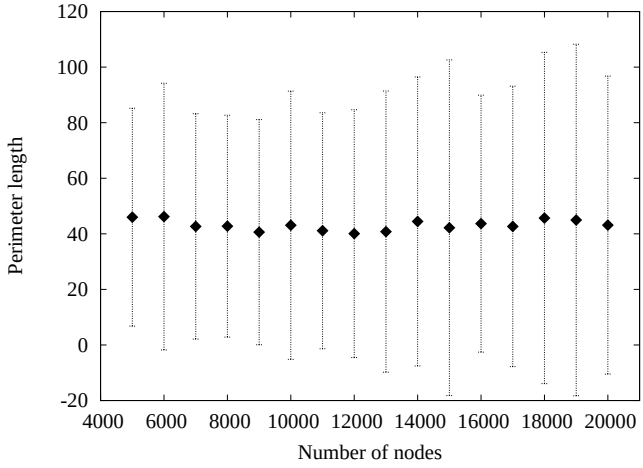
Our experiment is organized as follows. In order to measure the degree of unbalance of GHT, we simulated a flat square sensing field, with a $1000m$ side. Each node has circular transmission range with $30m$ radius. In this area, we simulated several WSNs ranging from 5000 to 20000 sensors. For each network size, we randomly generate 100 networks with uniform distribution. For each network, we compute the mean and the variance of the number of nodes found in a GPSR perimeter as follows. For each sensor network the simulator uniformly selects 1000 points and, for each point, it computes the number of nodes in the perimeter surrounding the point. GPSR need to work with a planar graph in order the perimeter mode to behave correctly, thus, in our experiments we use both GG and RNG planarization.

Figure 4.1.a and Figure 4.1.b show the mean and variance of the perimeter length used to store data by GHT. Figure 4.1.a shows the length of such perimeter found out using GG planarization and Figure 4.1.b shows the length of such perimeter found out using RNG planarization.

Both Figure 4.1.a and Figure 4.1.b show that the lenght of the perimeters is highly variable. This is even more evident if we compare the standard deviation of



(a) GG



(b) RNG

Figure 4.1: Mean and variance of perimeters (number of nodes) measured for different sensors numbers with GG and RNG planarization. The figure shows that the variance of the length of the perimeters is very high with respect to the number average number of nodes forming the perimeters.

the lenght of the perimeters with the average of the length of the perimeters. This variability is partly due to the behavior of nodes in the outer part of the sensing area, since in that area the probability of having very long perimeters (i.e., following the whole border) is high. With low densities, the probability that a random point belongs to the exterior of the network (and thus it is associated to the external perimeter) is not negligible.

To provide a better understanding of the load unbalancing problems that can be introduced by the variability of such perimeters, we perform another set of experiments, using the same setting as before, to measure how many pieces of data are stored by GHT on each node of the network.

The results of this last set of experiments are presented in Figure 4.2–4.4 and they present the results for the network size 5000, 12000 and 20000 and using both GG and RNG planarization.

In all charts, the x axis shows the different load (i.e., the amount of data for each sensor) on a node and the y axis shows the number of nodes storing exactly this number of data. Values on the y axis follow a logarithmic scale for better comprehension.

All the figures show the situation of load unbalance of the data distribution provided by GHT. In all the cases we can spot the nodes of the network stores an high variable quantity of data. We have a large number of nodes holding few data and a smaller number of nodes storing a lot of data.

All the results show how the uniform hash function used by GHT is not able to guarantee a control over the perimeter length and, as a consequence of this it is not able to provide a fair distribution of data inside the network.

Perimeters on the border In order to understand this *border effect*, we performed another set of simulations in which the sensing networks are generated in the same way as above but the external part of the sensors area is not used to store data. We "cut away" the 5% of the area from each border. Figure 4.5 shows the area that we are going to "cut away" in grey and the area that we use to store the data in white.

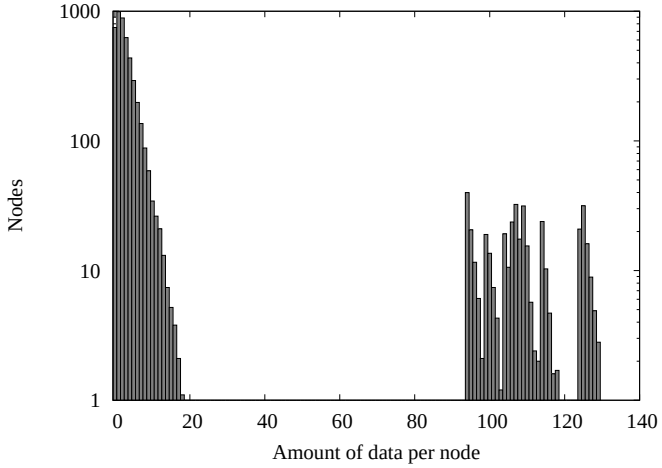
We randomly generate 1000 points in the white area and again measure the length of each perimeter and compute the mean and the variance. Figure 4.6.a and Figure 4.6.b show the results of this new experiment. Figure 4.6.a shows the length of such perimeter found using GG planarization and Figure 4.6.b shows the length of such perimeter found using RNG planarization.

The mean and the variance, as depicted in both Figure 4.6.a and Figure 4.6.b, improve if the border nodes are let out but standard deviation remains high.

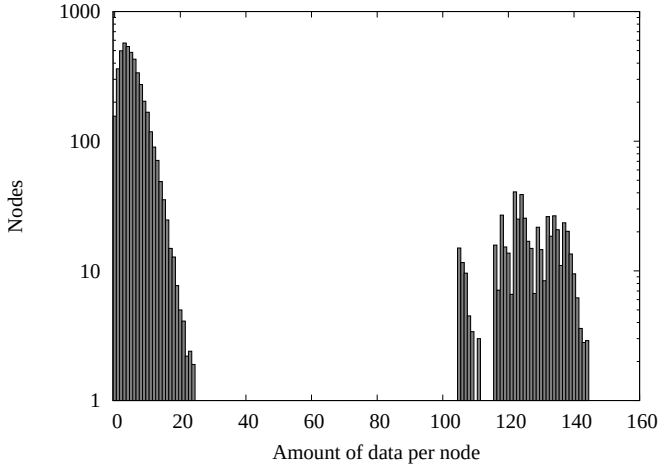
To show the benefits of the perimeter exclusion form the perimeter computation, we perform another set of experiments to measure how many piece of data are stored by GHT on each node of the network if we exclude the border of the network.

The results of this last set of experiments are presented in Figure 4.7–4.9 and they present the results for the network size 5000, 12000 and 20000 and using both GG and RNG planarization.

All the figures show the high benefits introduced by the exclusion of the border region of the network. If we compare these last results with the one shown in Figure 4.2–4.4, it is evident how the exclusion of the external part of the network and thus the exclusion of the external perimeter can bring to better load distribution performances.



(a) GG

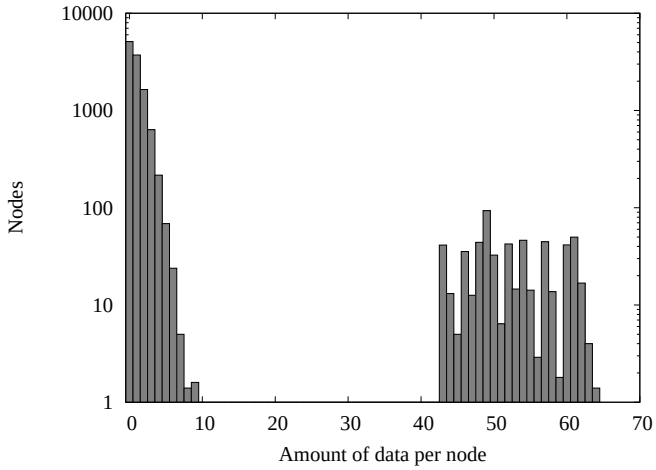


(b) RNG

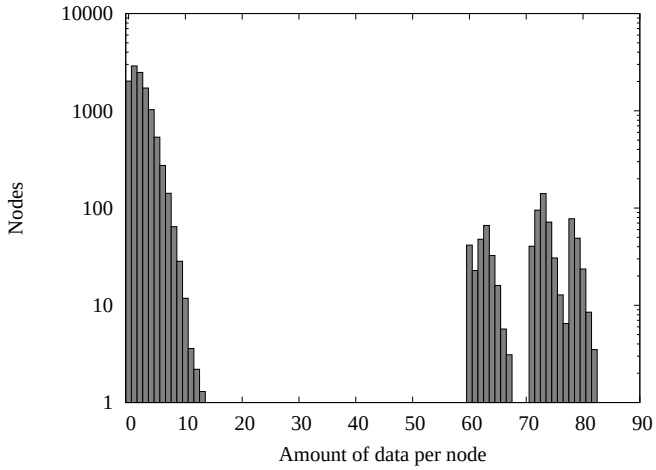
Figure 4.2: Amount of data stored in each node for uniform sensors distribution of 5000 *nodes with GG and RNG planarization.*

Non uniform sensor distribution In order to understand the behavior of GHT with non-uniform sensor distribution, we repeated our experiments using both a Gaussian function and a Hill function (Orecchia et al., 2004b) for distributing sensors. The Gaussian function has $\sigma = 1$ with maximum on the center of the area. The function is structured to have the 99 percentile matching the area.

The results concerning the perimeter lengths for the Gaussian distribution are shown in Figure 4.10.a and in Figure 4.10.b and the results for the Hill distribution are shown in Figure 4.11.a and in Figure 4.11.b.



(a) GG



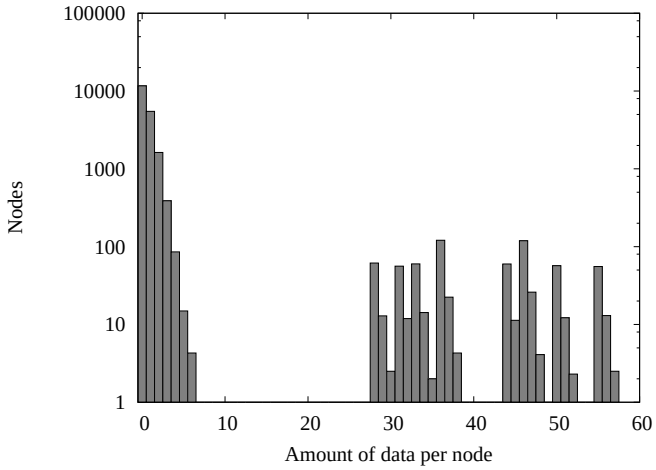
(b) RNG

Figure 4.3: Amount of data stored in each node for uniform sensors distribution of 12000 *nodes with GG and RNG planarization.*

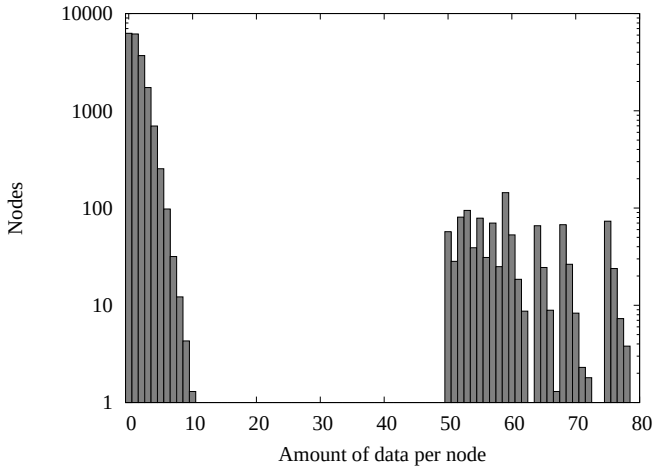
All the results show a behavior that is much worse than the one shown with uniform distribution because GHT uses a uniform hash function independently of the real distribution of the sensors.

The results concerning the load distribution are presented in Figures 4.12–4.14 and they present the results for Gaussian and Hill distributed networks of size 5000, 12000 and 20000.

All the figures show how the use of the uniform hashing function bring to a state of high unbalance in load distribution, much times worse than the one presented in



(a) GG



(b) RNG

Figure 4.4: Amount of data stored in each node for uniform sensors distribution of 20000 *nodes with GG and RNG planarization.*

Figures 4.2–4.4.

This brings to a pathological state of *load unbalance* that is due to the different amount of data that must be managed by an equal number of sensors: a sensor belonging to a sparser region the deployment area must manage a quantity of data that is larger than the quantity managed by a sensor inside a denser part of the network.

Load unbalance and QoS. Another issue with GHT is that there is no way to control the QoS provided for each datum. Since the point (x, y) on which a datum

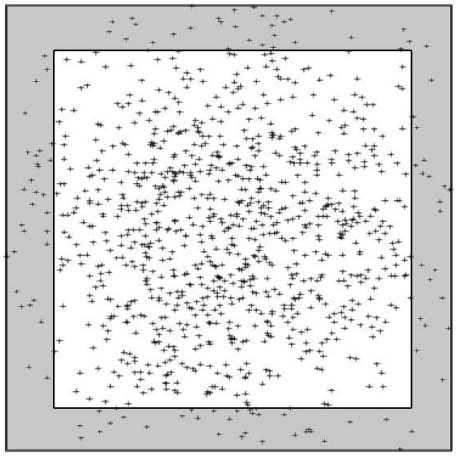


Figure 4.5: The border area (grey) and the storing area (white).

should be stored is obtained computing a hash function h on its associated meta-data M, the selection of the sensors candidate for storage is in practice independent from the importance of the datum.

In principle, this ensures the same treatment for each stored datum. However, if the meta-datum M is particularly popular and many sensors generate the data described by M, the sensors located in the perimeter around $(x, y) = h(M)$ would be burdened with a high storage and communication load.

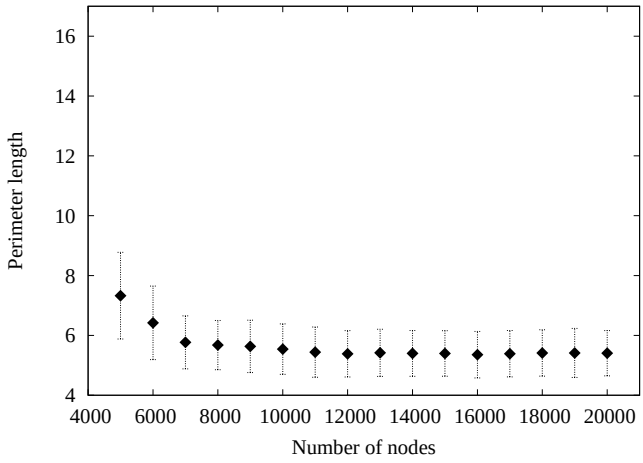
For this reason, the authors of GHT introduce the technique of *structured replication* (Ratnasamy et al., 2003), that replicates the same datum in different sub-areas of the sensing field. However neither GHT nor structured replication ensure that the level of redundancy associated to a datum is related to the importance of the datum itself: GHT assures only the same average treatment of each stored datum.

Another aspect is that, although the average level of redundancy of the meta-datum is constant, in practice it can vary significantly (due to the fact that each geographic point is surrounded by a different perimeter), even in case of uniform distribution of the sensors.

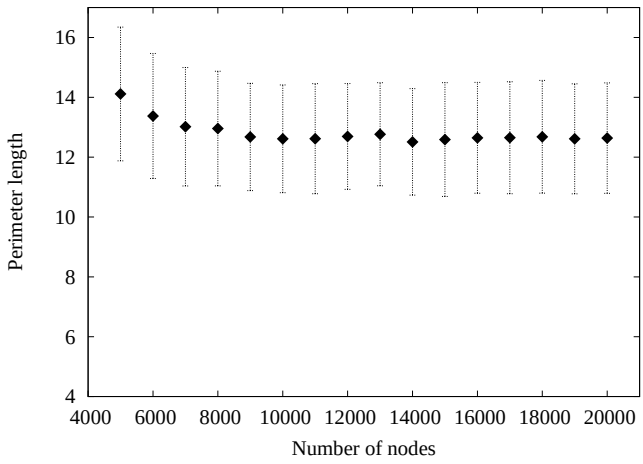
4.2 Q-NiGHT

In this section we present Q-NiGHT, a novel DCS protocol which moves from GHT incorporating QoS control and featuring good load balance among sensors distributed in a non-uniform way.

Q-NiGHT uses a strategy similar to the *rejection method* (Neumann, 1951) to build a hash function biased with sensor distribution. This spreads data more evenly among nodes. In addition, Q-NiGHT can provide QoS with different redundancy techniques. We first detail the protocol using pure replication, allowing the user to



(a) GG

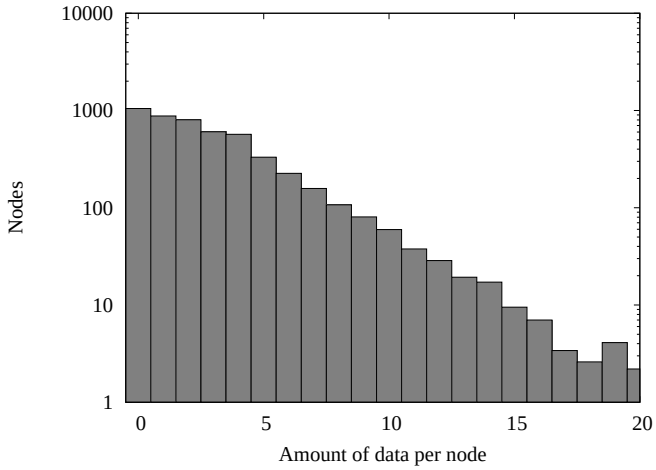


(b) RNG

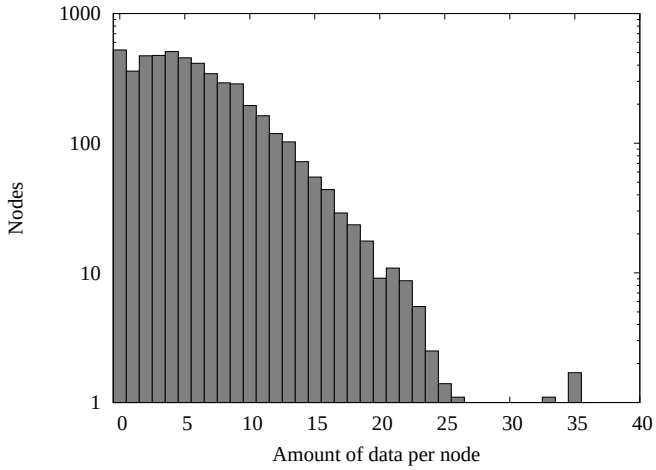
Figure 4.6: Mean and variance of perimeters (number of nodes) measured for different sensors numbers with GG and RNG planarization without considering the borders of the network. The figure shows a lower length of the perimeters with respect to the results presented in Figure 4.1.

choose the number of replicas required for a given datum. however all replication methods discussed in Chapter 3 can be used.

Then, we conduct detailed simulations of Q-NiGHT and analyze the results obtained with respect to the load of each sensor (i.e., the number of data stored in each node) and the number of messages needed for data storage and retrieval. Results show a good performance of Q-NiGHT on different sensors distributions



(a) GG



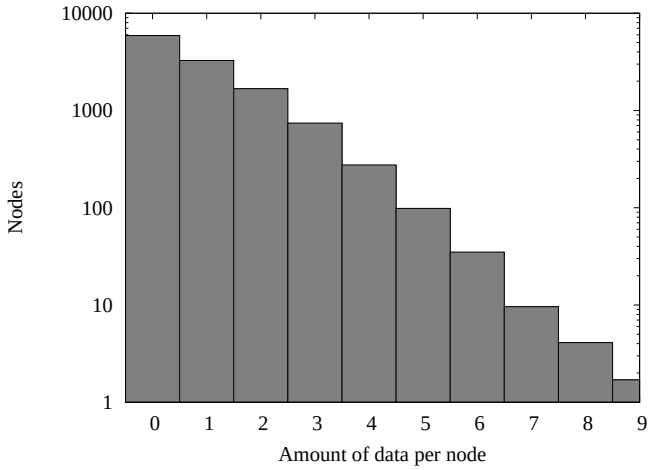
(b) RNG

Figure 4.7: Amount of data stored in each node for uniform sensors distribution of 5000 *nodes with GG and RNG planarization without considering the borders of the network.*

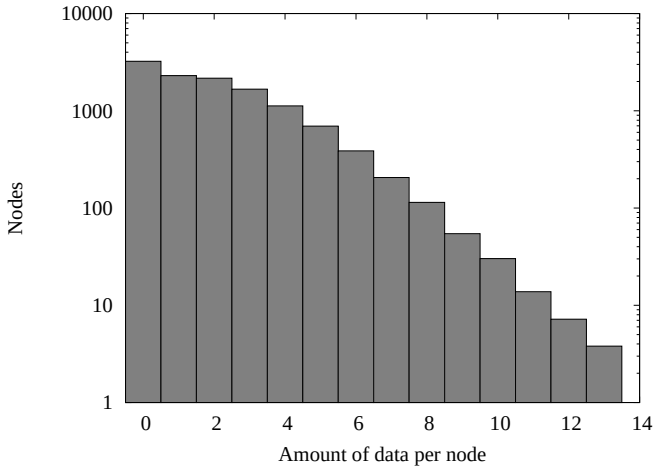
on terms of both protocol costs and load distribution.

4.2.1 A first step: GHT with non-uniform hashing

As we have seen, a serious problem with GHT is the fact that it uses a uniform hash function independently of the real distribution of sensors. This leads to a poor control on the perimeter lengths as shown in Figures 4.10–4.11 and to the pathological state



(a) GG

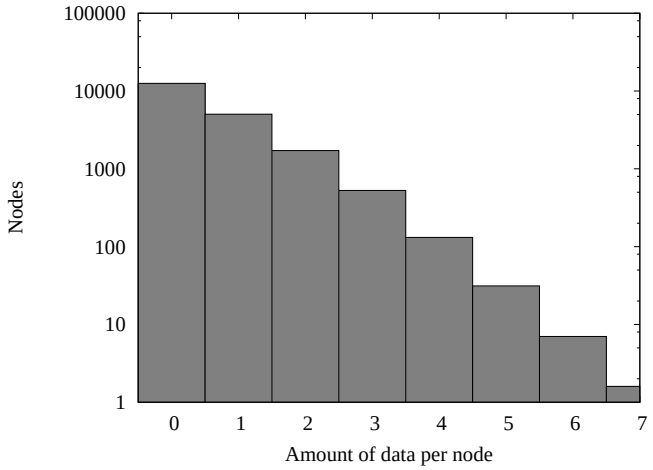


(b) RNG

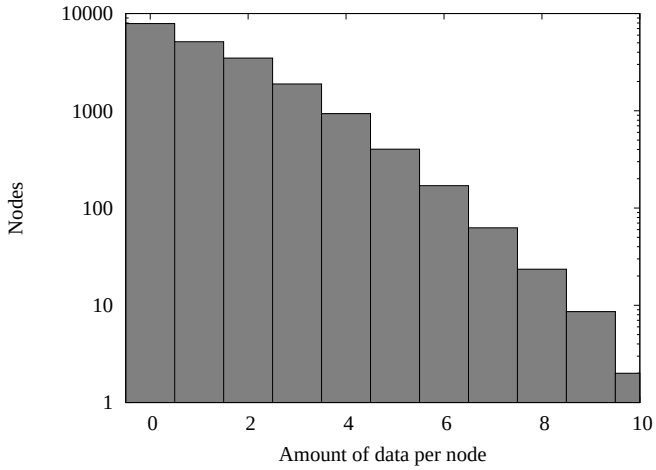
Figure 4.8: Amount of data stored in each node for uniform sensors distribution of 12000 *nodes with GG and RNG planarization without considering the borders of the network.*

of unbalancing in load distribution, as shown in Figures 4.12–4.14.

In this section, we explore what happens if we use hash functions which scatter data approximately with the same distribution of the sensors. We can observe that a hash function $h(k)$ is a kind of pseudo-random number generator: starting from a seed (in our case the key k) it produces an output (in our case in a value in $\mathbb{R}^2$) such that for *near* values of the key the hashed values must be *distant*. With this consideration in mind we define a new hash function, whose pseudo-code is



(a) GG

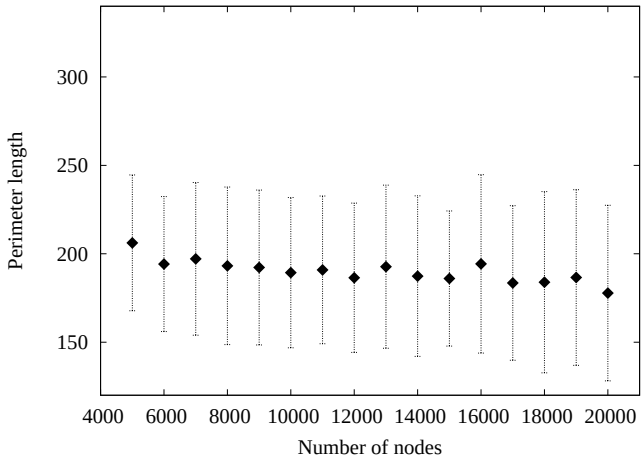


(b) RNG

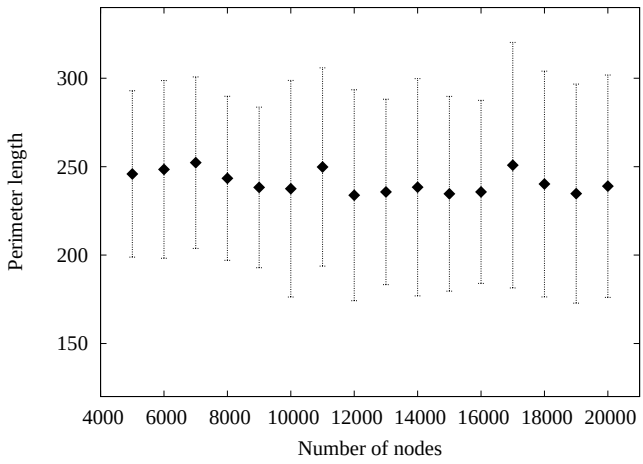
Figure 4.9: Amount of data stored in each node for uniform sensors distribution of 20000 *nodes with GG and RNG planarization without considering the borders of the network.*

shown in Algorithm 2. This function uses a strategy similar to the one used in the *rejection method* (Neumann, 1951), but with some differences. Rejection method is a technique used in random number generation to produce random numbers following any probability distribution, with limited dominion.

The basic idea is the following. Let us suppose to have a probability function and an interval (a, b) in which we want to generate random numbers, as depicted in Figure 4.15.



(a) GG

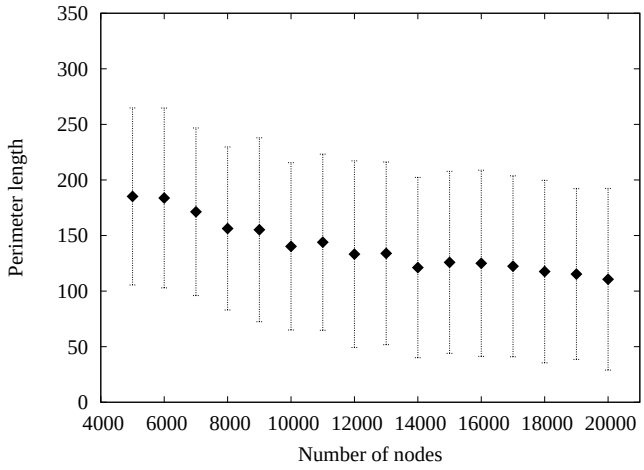


(b) RNG

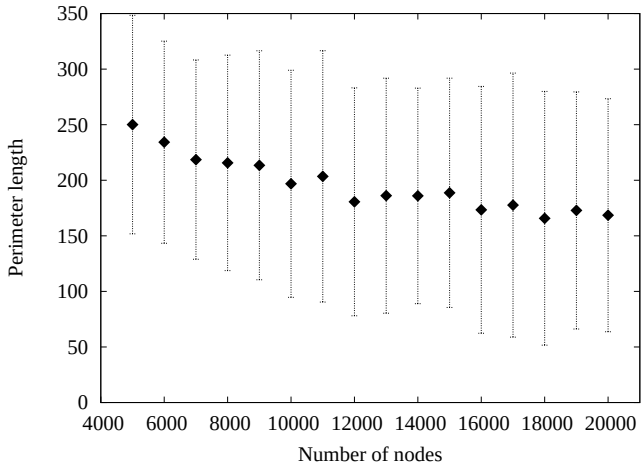
Figure 4.10: Mean and variance of perimeters (number of nodes) measured for different sensors numbers with GG and RNG planarization using the Gaussian distribution.

The probability function is boxed and we generate uniform random values in the box. If the value generated is below the distribution function the value is accepted and returned. Otherwise, we randomly generate new points in the box until a value is below the function.

Notice that in principle there is a non-null probability of non termination because the values can be generated all above the function. In practice a good uniform hash grants to generate values in all the box.



(a) GG

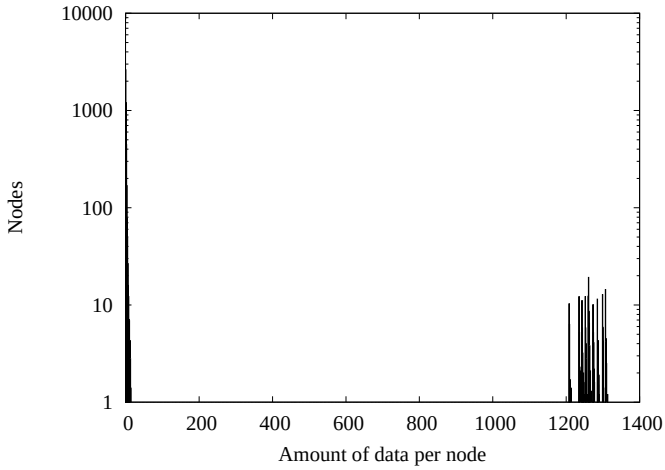


(b) RNG

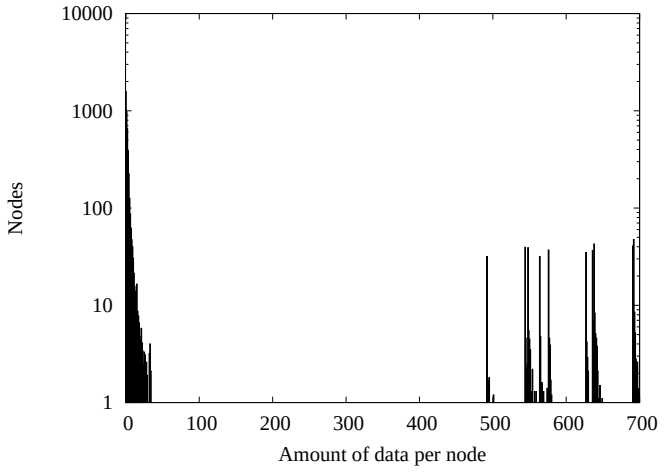
Figure 4.11: Mean and variance of perimeters (number of nodes) measured for different sensors numbers with GG and RNG planarization using the Hill distribution.

From its key k, the function RejectionHash returns a pair (x, y) of coordinates where to place data, belonging to distribution f. Instead of using uniform randomization to provide the points inside the box, RejectionHash uses uniform hashing on the key. At each iteration, if necessary, it changes lightly the key in a deterministic way. This operation is represented by the increment of the key k of a quantity i that is increased at each iteration. This operation is needed to guarantee that at each iteration we provide a new, and very different, uniformly hashed vale.

In order to understand the goodness of RejectionHash function we repeated



(a) Gaussian



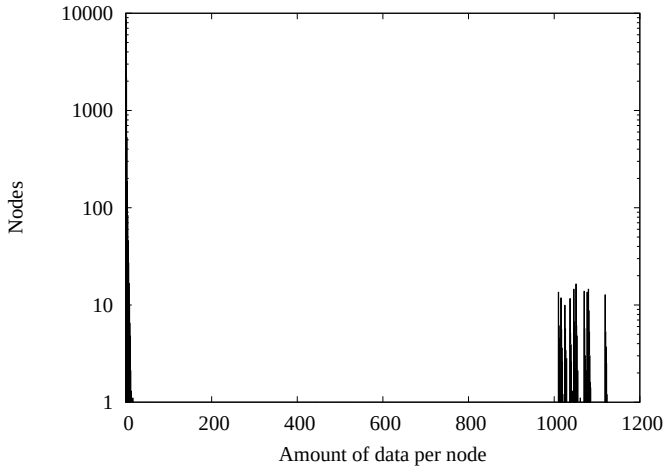
(b) Hill

Figure 4.12: Amount of data stored in each node by GHT for the Gaussian and Hill sensors distribution of 5000 *nodes.*

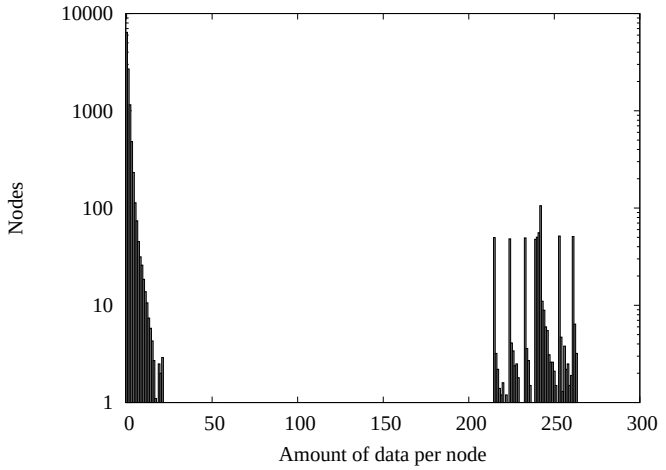
our experiments presented in Figures 4.10–4.11 and in Figures 4.12–4.14 using the RejectionHash function instead of the uniform GHT hash function.

The results showing the mean and the variance of the length of the perimeters for the Gaussian distribution are shown in Figure 4.16 and the results for the Hill distribution are shown in Figure 4.17.

Both Figure 4.16 and Figure 4.17 show the good behavior of the non-uniform hash function. RejectionHash fits well the sensors distribution in the data dissemination strategy with a good global load distribution.



(a) Gaussian

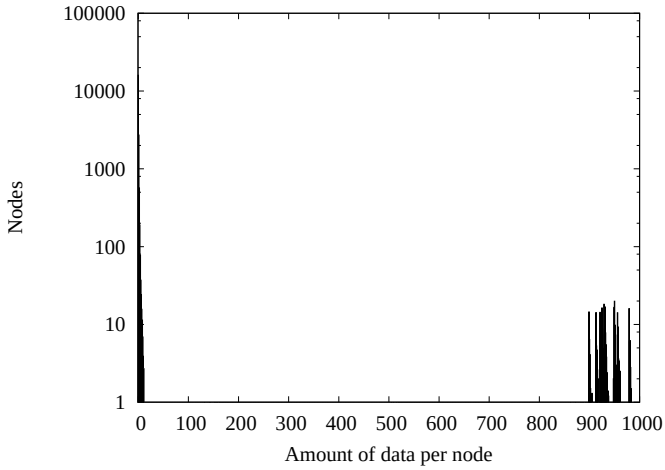


(b) Hill

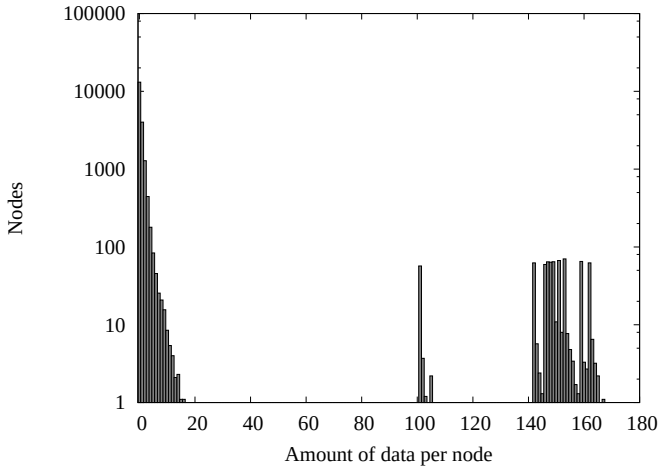
Figure 4.13: Amount of data stored in each node by GHT for the Gaussian and Hill sensors distribution of 12000 *nodes.*

These results are even better than the results provided with uniform distribution and uniform hashing (Figure 4.1 and Figure 4.6). This is due to the nature of non-uniform distributions themselves.

A non-uniform distribution tends to have parts of the network that are more dense than others. RejectionHash function distributes more data in these region. As happened in the uniform case, the perimeters get shorter in larger, and thus denser, networks because the nodes are closer and few nodes are needed to form a perimeter around a point.



(a) Gaussian



(b) Hill

Figure 4.14: Amount of data stored in each node by GHT for the Gaussian and Hill sensors distribution of 20000 *nodes.*

The evidence of these results is more clear if we look at Figures 4.18–4.20. This last set of experiments show the quantity of data stored in each node by GHT if it uses RejectionHash instead of the uniform hash function.

Figures 4.18–4.20 show a big improvement in the load distribution of GHT, with respect to the results presented in Figures 4.12–4.14. Even if GHT still the perimeters to store data in the network, the use of a hash function that is able to provide the data coordinates with a distribution biased on the network distribution shows big improvements.

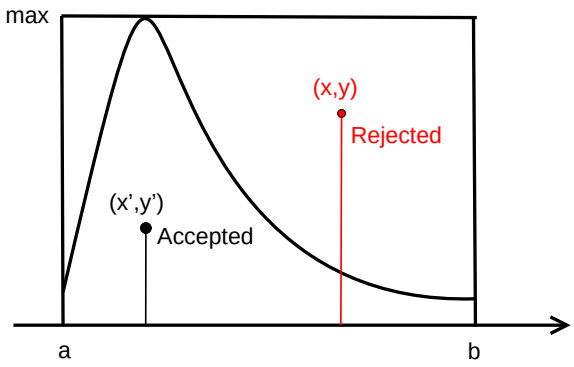


Figure 4.15: Rejection method: The probability function is boxed and we generate uniform random values in the box. If the value generated is below the distribution function the value is accepted and returned. Otherwise, it is rejected.

Algorithm 2 RejectionHash(k, f)

Require: A key k and a function f.
Ensure: A coordinate pair (x, y).

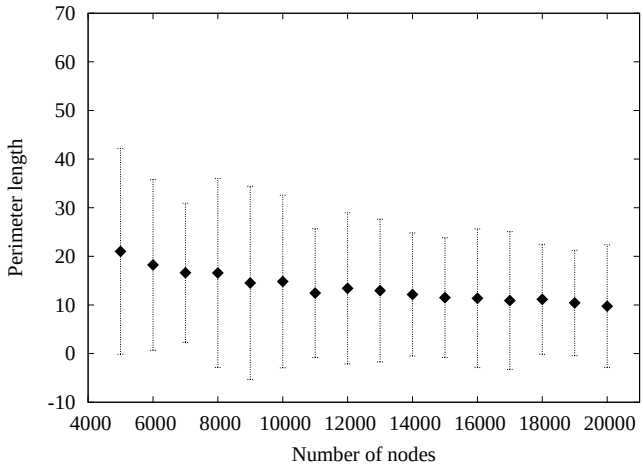
$i \leftarrow 0$
loop
  $(x, y, z) \leftarrow Hash(k + i)$
  $i \leftarrow i + 1$
  if $z < f(x, y)$ **then**
    return (x, y)
  end if
end loop

4.2.2 Q-NiGHT: adding quality of service to Geographic Hash Tables

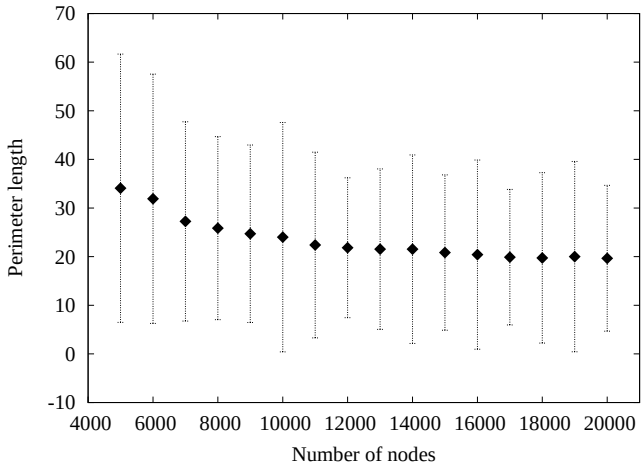
Exactly as GHT (Ratnasamy et al., 2003), Q-NiGHT is built atop the GPSR (Karp and Kung, 2000). Q-NiGHT provides data insertion (via `put`) and data retrieval (via `get`) on the sensor network.

To our purposes, the interface of the `put` includes, along with the meta-data M and the data D, also a parameter Q expressing the desired QoS. Q gives a measure of the dependability required for the data, and may be expressed using different metrics and ranges according to the particular redundancy technique used (see Section 3.2.3).

For instance, if Q-NiGHT adopts pure replication then Q can express the number of sensors on which the data should be replicated, or, if Q-NiGHT adopts n out of m redundant encodings (Rabin, 1989, Rizzo, 1997), then Q can express the number of fragments in which partition the data (each fragment to be stored in a different



(a) GG

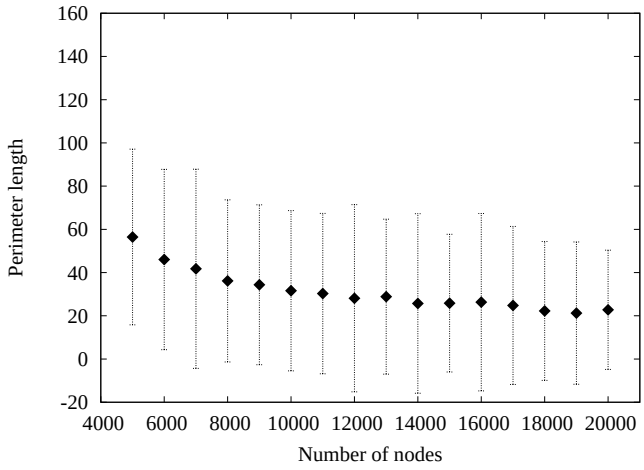


(b) RNG

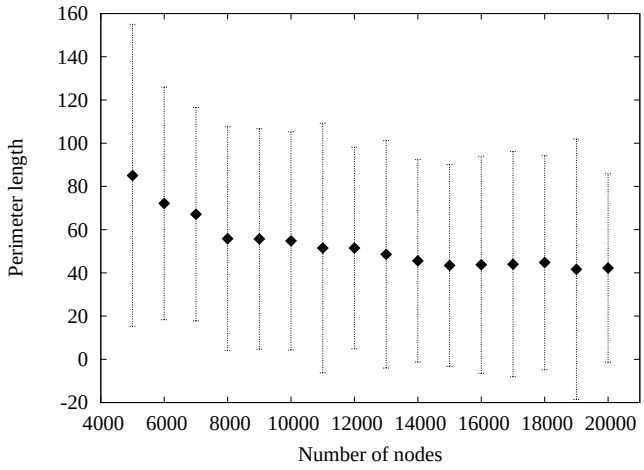
Figure 4.16: Mean and variance of perimeters (number of nodes) measured for different sensors numbers with GG and RNG planarization using the Gaussian distribution and the RejectionHash *hashing function.*

sensor) and the number of redundant fragments.

In the following, we describe Q-NiGHT assuming pure replication of the data. Data insertion is expressed with `put(M,D,Q)`. We assume Q ranges in $[1, Q_{max}]$ and gives the number of sensors on which the data should be replicated. Let s be the source node of a `put(M,D,Q)` operation. s first computes $h(M)$, where h is the RejectionHash function conditioned with the sensor distribution function, f, in the sensing field, as discussed in Section 4.2.1. $h(M)$ returns a pair of geographic coor-



(a) GG

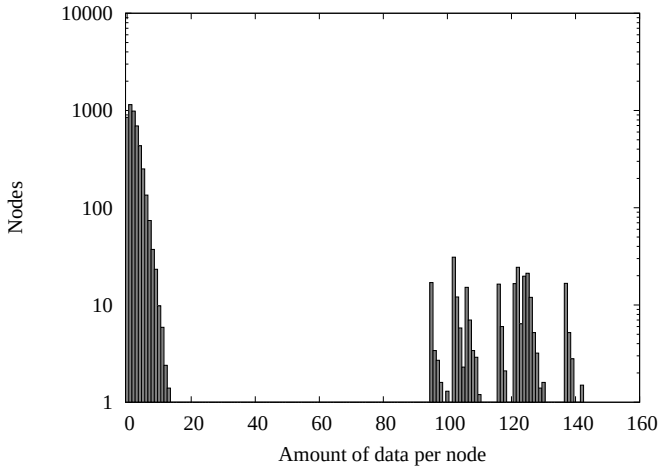


(b) RNG

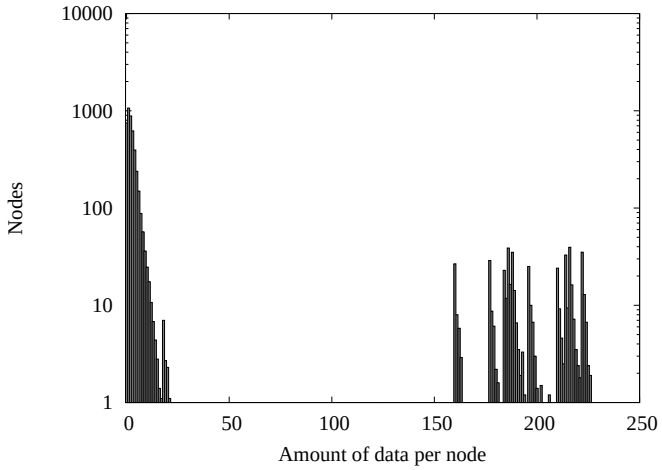
Figure 4.17: Mean and variance of perimeters (number of nodes) measured for different sensors numbers with GG and RNG planarization using the Hill distribution and the RejectionHash *hashing function.*

dinates (x, y) as the destination of the packet `P`$_p$`=<(x,y),<M,D,Q>>`. The packet in turn is sent to the destination using the GPSR protocol. As in GHT, we call *home node* the sensor s_d (of coordinates (x', y')) geographically closest to the destination coordinates. The home node naturally receives the packet as a consequence of applying GPSR. Upon the reception of packet `P`$_p$, sensor s_d begins a *dispersal protocol* which selects Q sensors to store a copy of `<M,D>`.

The dispersal protocol is iterative and uses the concept of *ball*. Given a sensor s_d



(a) Gaussian



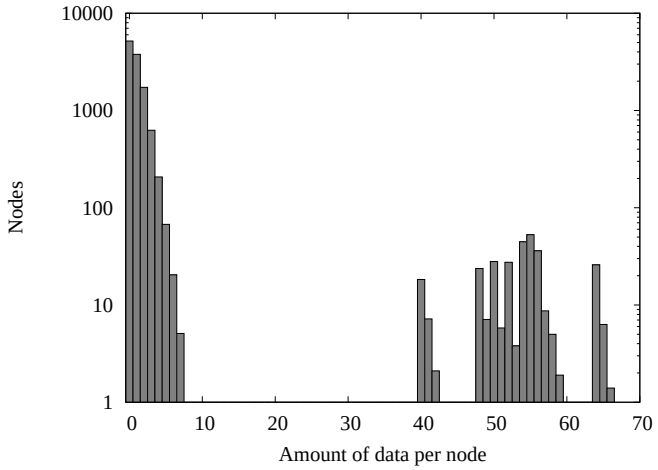
(b) Hill

Figure 4.18: Amount of data stored in each node by GHT (using RejectionHash*) for the Gaussian and Hill sensors distribution of* 5000 *nodes.*

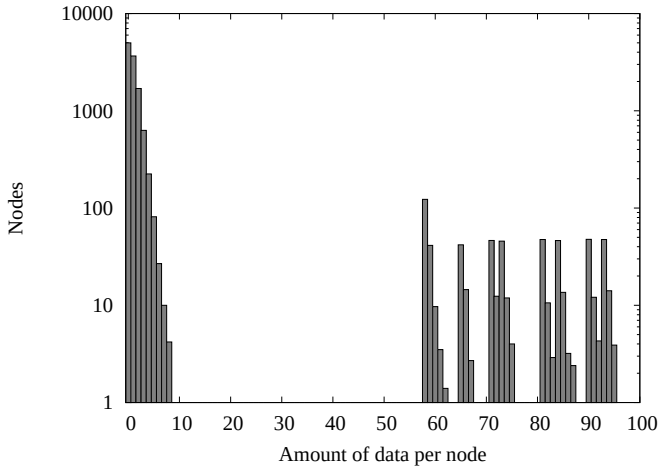
of coordinates (x', y'), we denote with $B_{(x,y)}(\overline{r})$ the *ball* centered in (x, y) of radius $\overline{r}$, that is the set of sensors that are within a Euclidean distance $\overline{r}$ from (x, y). As depicted in Figure 4.21.a, in the first iteration s_d broadcasts a replica of `D` to all the sensors included in the ball $B_{(x,y)}(\overline{r})$. $\overline{r}$ is chosen in order to reach the Q sensors closest to (x, y) with high probability[†].

Each sensor receiving a replica responds with an acknowledgment to s_d as de-

[†] A more detailed description of the computation of $\overline{r}$ with such a property is described in the next subsection.



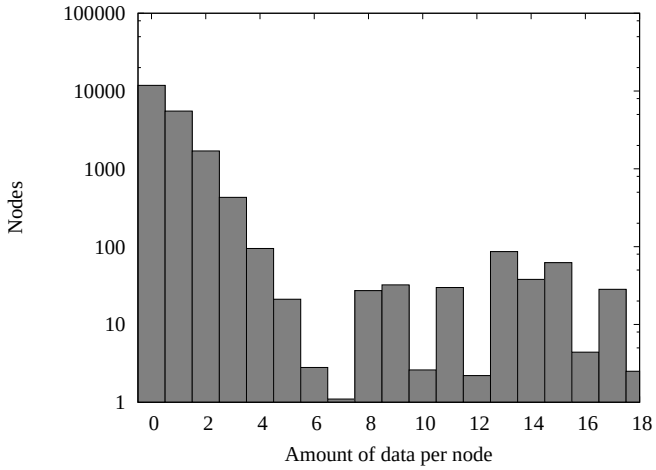
(a) Gaussian



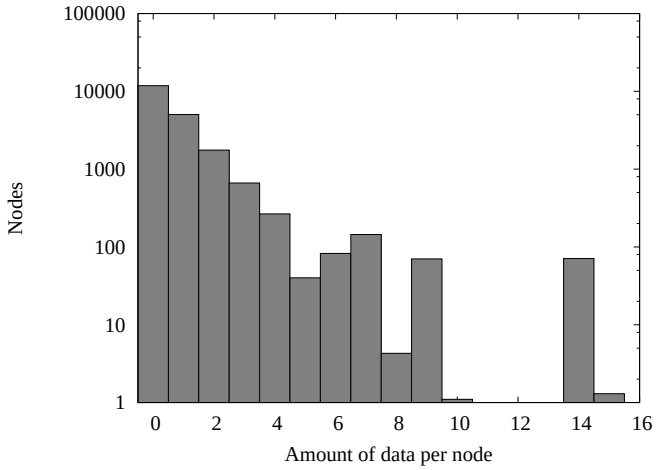
(b) Hill

Figure 4.19: Amount of data stored in each node by GHT (using RejectionHash*) for the Gaussian and Hill sensors distribution of* 12000 *nodes.*

picted in Figure 4.21.b. Finally, sensor s_d confirms the $Q-1$ acknowledgments received from the sensors geographically nearest to (x, y) and disregards the others (Figure 4.21.c). The confirmation requires an extra packet sent by s_d. Sensors which receive the confirmation keep the datum while the others will disregard the data after a timeout. If s_d receives $Q' < Q$ acknowledgments, then it executes another iteration of the dispersal protocol with $\overline{r} = 2\overline{r}$ in which it considers only the sensors in $B_{(x',y')}(2\overline{r}) - B_{(x',y')}(\overline{r})$. The dispersal protocol stops as soon as Q sensors have been hired or the outermost perimeter has been reached. The dispersal protocol is



(a) Gaussian



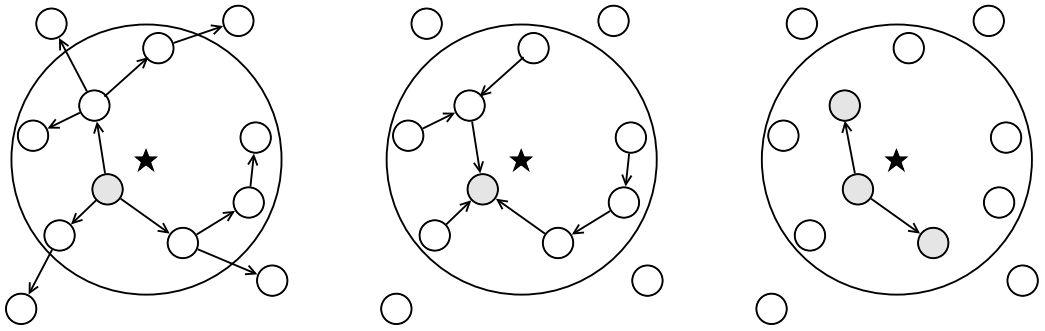
(b) Hill

Figure 4.20: Amount of data stored in each node by GHT (using RejectionHash*) for the Gaussian and Hill sensors distribution of* 20000 *nodes.*

a simple implementation of a geo-casting protocol (Seada and Helmy, 2006).

When a node s_g of coordinates (r, z) executes `get(M)`, it first computes $(x, y) = h(M)$, and then sends a query packet $\mathtt{P}_g$`=<(x,y),<(r,z),M>>` using the GPSR protocol. In turn, packet $\mathtt{P}_g$ will reach the perimeter surrounding (x, y) and it will start to move across the perimeter. Eventually, the packet will reach either the home node or another node containing a replica of the data D associated to M. This node will stop packet $\mathtt{P}_g$ and will send the required data back to s_g.

The complexity of the `put` protocol clearly depends upon the choice of $\overline{r}$ as this



(a) Broadcast phase

(b) Replay of the nodes inside $B_{(x,y)}(\overline{r})$

(c) Choice of the closest sensors

Figure 4.21: Dispersal protocol of a datum D with $Q = 3$ and $(x, y) = h(M)$ (represented by the star in the three pictures). (a) The home node (shaded) broadcasts D up to distance $\overline{r}$. (b) The nodes inside the $B_{(x,y)}(\overline{r})$ replay to the home node. (c) The home node sends the confirmation to the $Q - 1$ closest nodes.

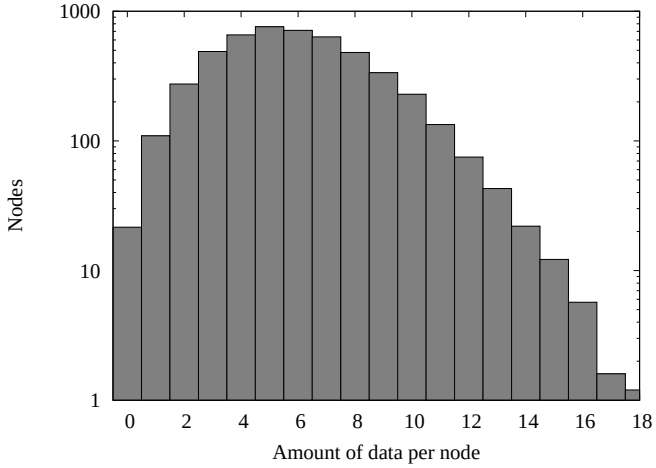
determines the number of iterations made to successfully place the Q replicas.

In order to evaluate our solution, using the *ball* instead of the perimeter to store data, we perform simulations to check the quantity of data stored in each node. As before, we simulate a flat square sensing field, with a $1000m$ side. Each node has circular transmission range with $30m$ radius. In this area, we simulated several WSNs made up of 5000, 12000 and 20000 sensors, using both RNG and GG to planarize the networks during GPSR perimeter modes. For each network size, we randomly generated 100 networks with uniform, Gaussian and Hill distributions, and performed 2000 `put` operations with uniformly generated meta-data using both GHT and Q-NiGHT. In these trials, Q-NiGHT uses a pure replication QoS with 15 replicas for each datum.

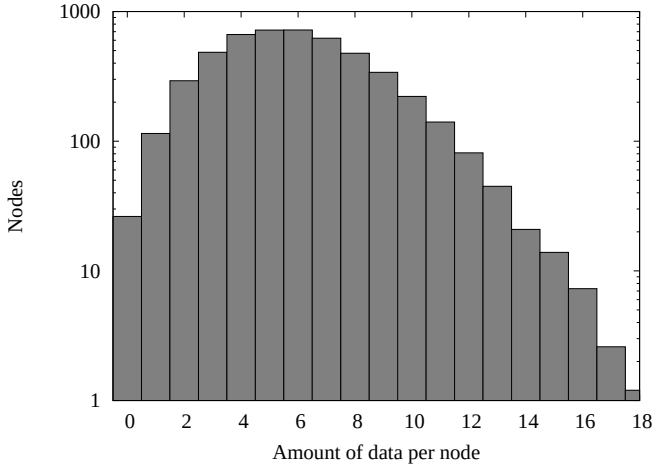
In all charts, the x axis shows the different load (i.e., the number of pieces of data stored) on a node and the y axis shows the number of nodes storing exactly this number of pieces of data. Values on the y axis follow a logarithmic scale for better comprehension.

Figures 4.22–4.24 show the quantity of data stored in each node for uniform, Gaussian and Hill sensors distributions of a network made up of 5000 sensors using RejectionHash and the Q-NiGHT dispersal protocol.

If we compare Figures 4.22–4.24 with their counterparts that do not use the dispersal protocol, namely Figure 4.2, Figure 4.7, Figure 4.12 and Figure 4.18, we can notice that Q-NiGHT reaches better load balance than GHT, even in the uniform case. In all the cases, GHT shows its unbalance problems, while Q-NiGHT manages to balance the load. The reason of such a behavior, is that GHT stores data on perimeters and since such perimeters are chosen arbitrarily, GHT produces very unbalanced storages.



(a) GG

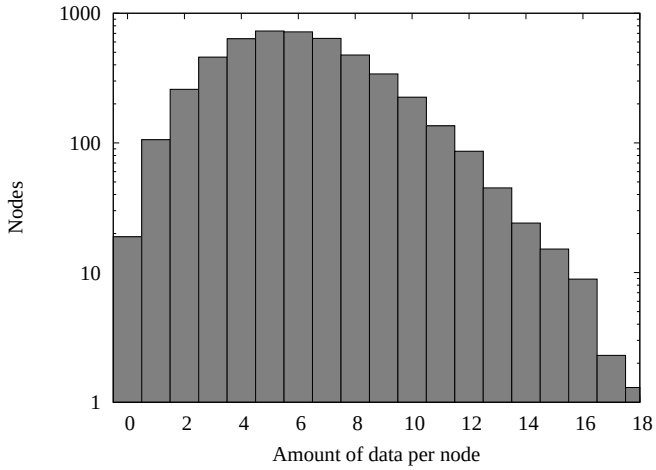


(b) RNG

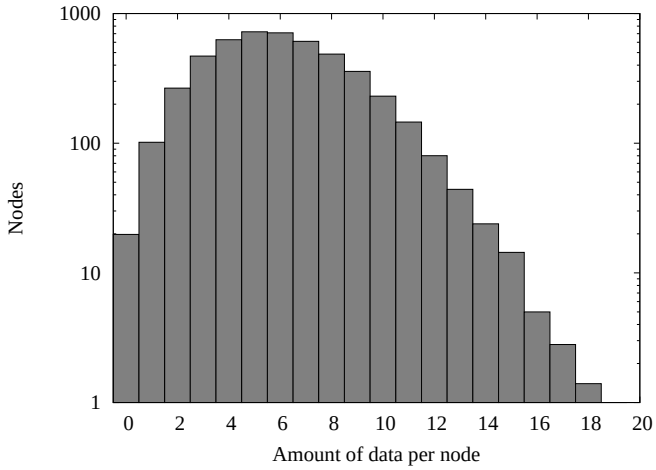
Figure 4.22: Amount of data stored in each node for uniform sensors distribution of 5000 *nodes using* RejectionHash *and the* Q-NiGHT *dispersal protocol.*

For instance, comparing Figure 4.22 (the uniform case) with Figure 4.2, the differences are very large. We can spot the following interesting situation: a large number of nodes store few data (approximatively between 0 and 25 data for each node) and at the same time, in the same network, we have good number of nodes storing a lot of data (approximatively between 100 and 150 data for each node). This storage configuration finds its explanation in the use of the external perimeter to store data.

On the other hand, Q-NiGHT stores almost the same number of data on each node (approximatively between 0 and 18 data for each node) because Q-NiGHT



(a) GG



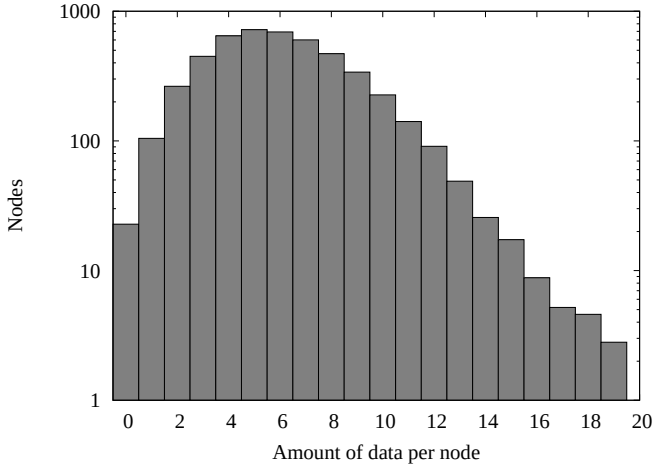
(b) RNG

Figure 4.23: Amount of data stored in each node for Gaussian sensors distribution of 5000 *nodes using* RejectionHash *and the* Q-NiGHT *dispersal protocol.*

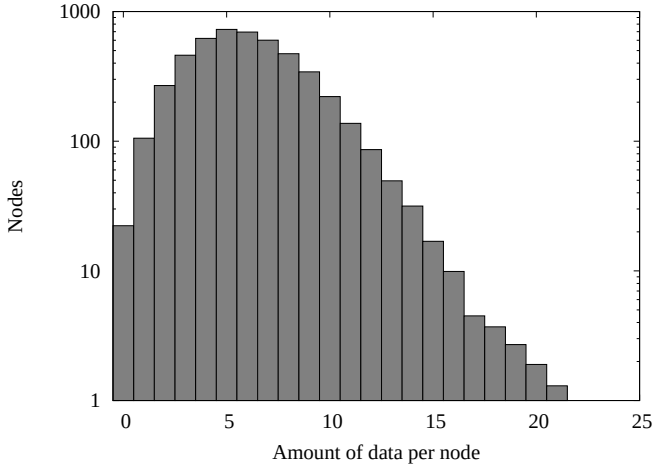
looks for a fixed number Q of storage nodes for each datum and it is much more difficult for the same node to store a large number of different data.

The difference is smaller if we compare Figure 4.22 with Figure 4.7, in this last case the network do not use the external perimeter to store data and thus, GHT finds smaller perimeters, with size comparable to our Q.

The different behaviors of Q-NiGHT and GHT are more evident in non-uniform networks (Figures 4.23–4.24). If we compare Figures 4.23–4.24 with Figure 4.12 and Figure 4.18, we can easily see the great advantage of using both the Rejection-



(a) GG



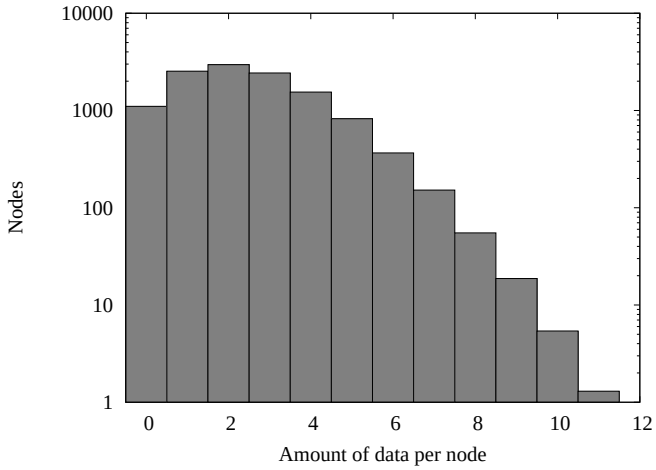
(b) RNG

Figure 4.24: Amount of data stored in each node for Hill sensors distribution of 5000 *nodes using* RejectionHash *and the* Q-NiGHT *dispersal protocol.*

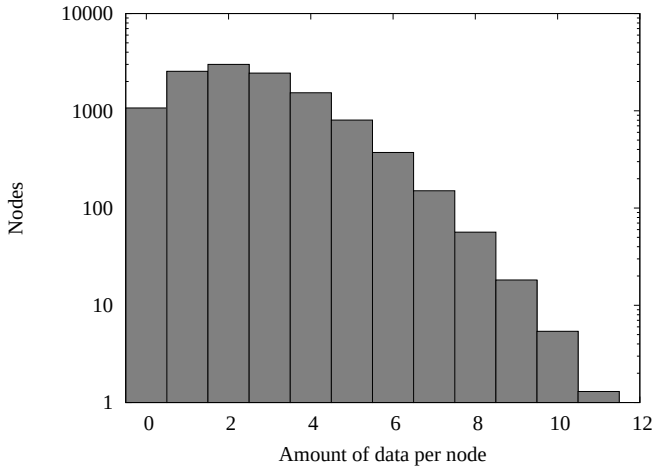
Hash, to locate data accordingly to the network distribution and thus preventing to use sparser regions of the network and the use of the dispersal protocol to spread data on a limited number of nodes.

Figures 4.25–4.27 show the quantity of data stored in each node for uniform, Gaussian and Hill sensors distributions of a network made up of 12000 sensors using RejectionHash and the Q-NiGHT dispersal protocol.

Finally, Figures 4.28–4.30 show the quantity of data stored in each node for uniform, Gaussian and Hill sensors distributions of a network made up of 20000 sensors



(a) GG



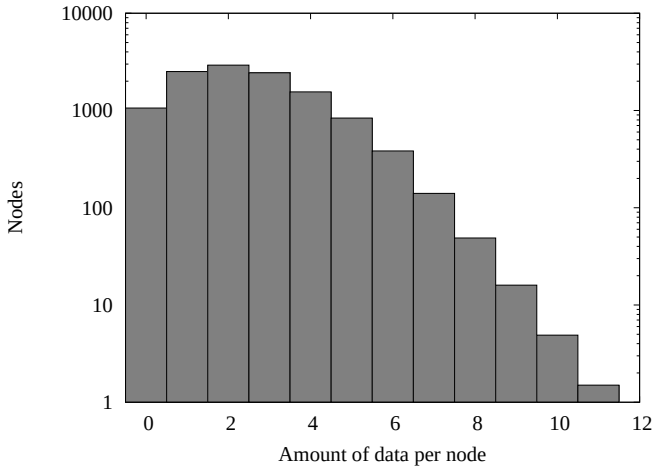
(b) RNG

Figure 4.25: Amount of data stored in each node for uniform sensors distribution of 12000 *nodes using* RejectionHash *and the* Q-NiGHT *dispersal protocol.*

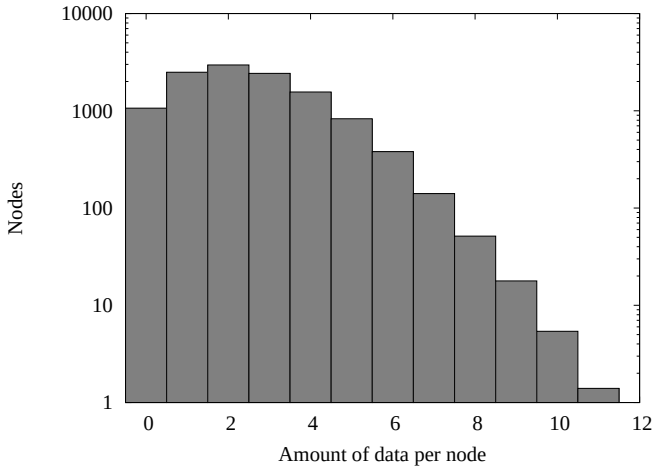
using RejectionHash and the Q-NiGHT dispersal protocol.

All the considerations made for the 5000 nodes networks still apply when we move into the analysis of larger networks. Also in the case of 12000 nodes networks (Figures 4.25–4.27) and 20000 nodes networks (Figures 4.28–4.30), we can notice that Q-NiGHT reaches better load balance than GHT.

On the other hand, Q-NiGHT, reaches better performance while the network size increases. Once fixed the number of data to be stored and the parameter Q, the probability that a same node stores multiple data become smaller as the number of



(a) GG



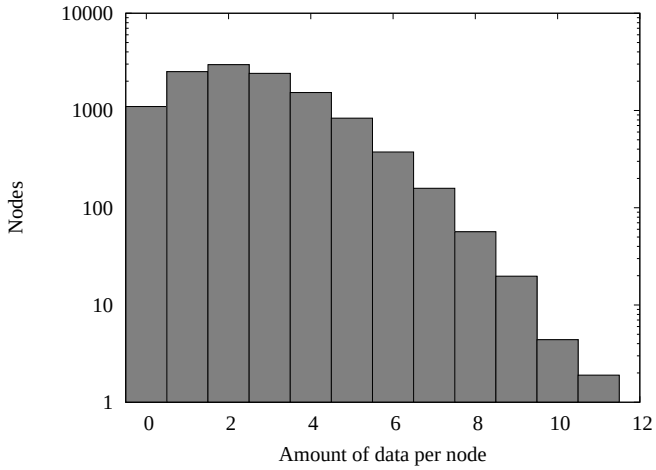
(b) RNG

Figure 4.26: Amount of data stored in each node for Gaussian sensors distribution of 12000 *nodes using* RejectionHash *and the* Q-NiGHT *dispersal protocol.*

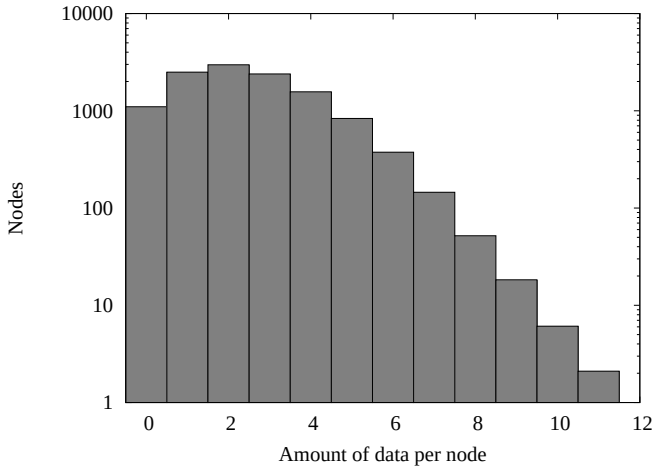
nodes inside the network increases.

$\overline{r}$ computation. We now discuss how to compute the radius $\overline{r}$ of the ball used by the Q-NiGHT dispersal protocol to choose the nodes to store data in.

Let f be the sensor probability distribution function, Q an integer in $[1, Q_{max}]$, and (x, y) a point in $\mathbb{R}^2$ we want to fix $\overline{r}$ in such a way that, called S_B the set of sensors laying in the space defined by the ball $B_{(x,y)}(\overline{r})$, $|S_B| \geq Q$ with high probability.



(a) GG



(b) RNG

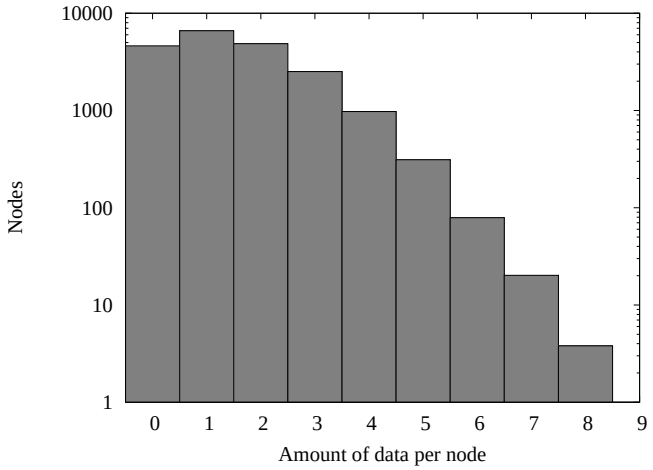
Figure 4.27: Amount of data stored in each node for Hill sensors distribution of 12000 *nodes using* RejectionHash *and the* Q-NiGHT *dispersal protocol.*

We first discuss the intuition behind our approximation schema. Let A_r denote the circle of radius r, we want to fix r such that

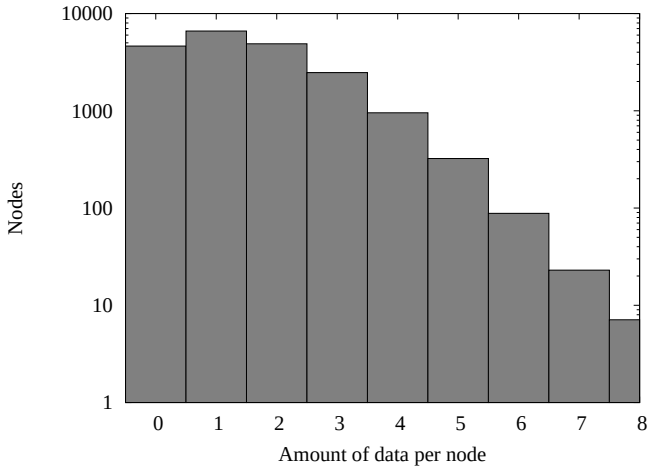
$$E[number\ of\ sensors\ \in\ A_{\bar{r}}] = Q \tag{4.2.1}$$

Equation (4.2.1) can be rewritten in terms of the probability f

$$n \cdot \int_{A_{\bar{r}}} f(x,y)\mathrm{d}A_{\bar{r}} = Q \tag{4.2.2}$$



(a) GG

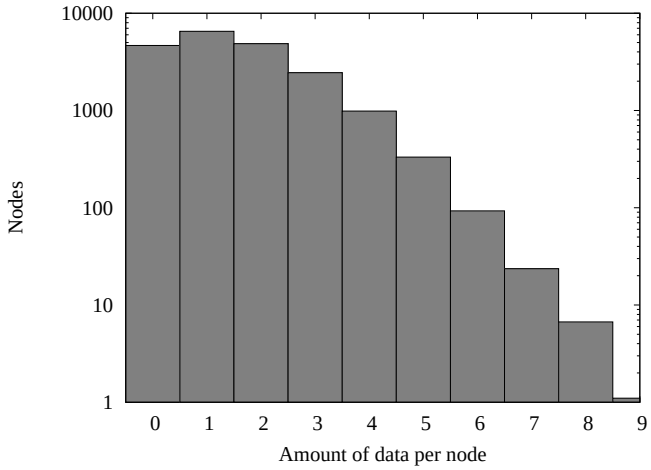


(b) RNG

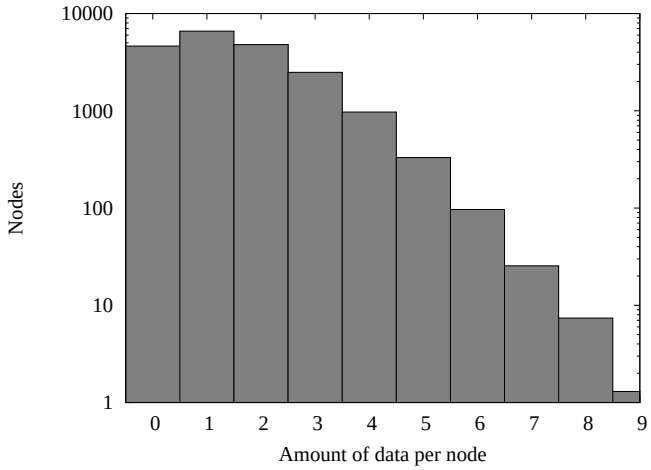
Figure 4.28: Amount of data stored in each node for uniform sensors distribution of 20000 *nodes using* RejectionHash *and the* Q-NiGHT *dispersal protocol.*

where n is the total number of sensors in the network and the integral represents the probability to have a sensor in $A_{\overline{r}}$. Direct computation of the integral above is likely to need a large number of floating point operations in practical distributions.

To simplify the computation of $\overline{r}$, we use the following strategy. Instead of using $A_{\overline{r}}$,we use the square inscribed in the circle of radius $\overline{r}$. This square has edge $\overline{r}\sqrt{2}$ and area $A'_{\overline{r}} = 2\overline{r}^2$. If we impose to have Q nodes in $A'_{\overline{r}}$, we grant at least Q nodes



(a) GG



(b) RNG

Figure 4.29: Amount of data stored in each node for Gaussian sensors distribution of 20000 *nodes using* RejectionHash *and the* Q-NiGHT *dispersal protocol.*

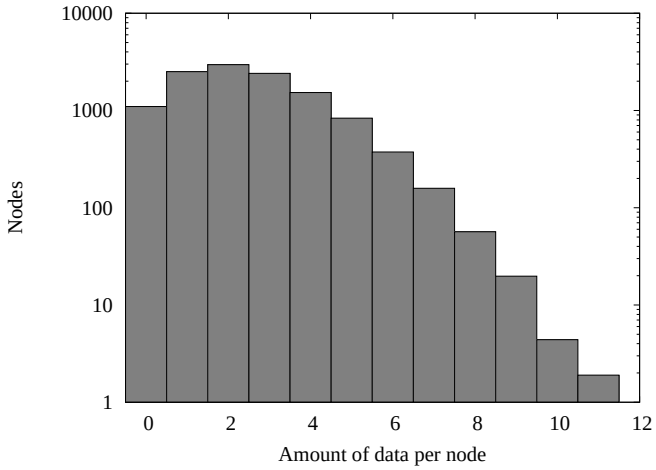
in $A_{\overline{r}}$. Equation (4.2.2) becomes

$$n \cdot \int_{A'_{\overline{r}}} f(x,y)\mathrm{d}A'_{\overline{r}} = Q.$$

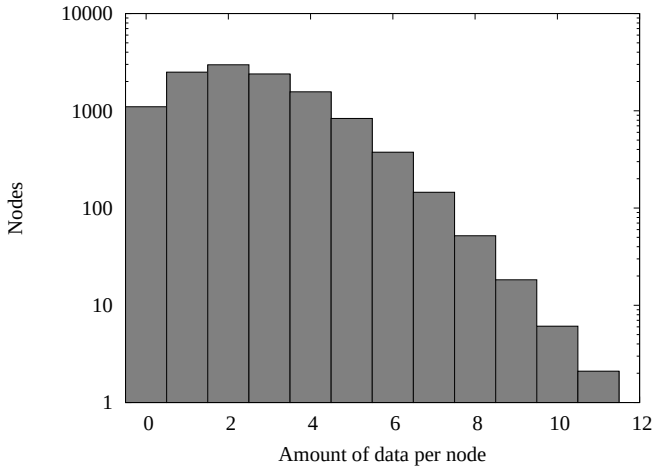
The volume identified by the integral can be represented in terms of $\overline{r}$ as follows

$$n \cdot 2 \cdot \overline{r}^2 \cdot h = Q \tag{4.2.3}$$

where h is the ideal height of the cuboid which volume is equivalent to the one of integral.



(a) GG



(b) RNG

Figure 4.30: Amount of data stored in each node for Hill sensors distribution of 20000 *nodes using* RejectionHash *and the* Q-NiGHT *dispersal protocol.*

We use the height h to simplify the computation of $\overline{r}$: sensors do not compute the integral but have a stepped version of the nodes distribution function f. A sensor chooses h between the heights of the steps that include A'. h is the the minimum of such heights. If the minimum of such heights is 0, the second minimum is chosen. This means that the value of $\overline{r}$ identifies a number of sensors greater than Q. Now, from Equation (4.2.3) we can derive the minimum $\overline{r}$ that identifies a number of

sensors greater than Q

$$\overline{r} = \sqrt{\frac{Q}{2 \cdot n \cdot h}}. \tag{4.2.4}$$

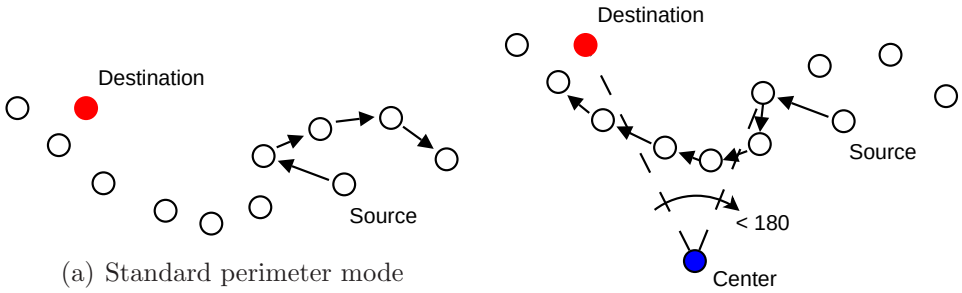
The complexity of the computation of Equation 4.2.4 is low, even for a sensor. The computation involves two multiplications, a division and a square root operation. The multiplications and the division are simple operations supported by hardware. Then, the square root operation can be computed using tables and/or approximated methods, which complexity depends on the memory usage and the wanted approximation.

Behavior in case of faults. In case some of the nodes holding the replicas of `<M,D>` fail, our protocol continues to operate correctly. Due to the GPSR protocol, any `get` with key `M` is routed to the node geographically closest to $(x, y) = h(M)$.

If the faulty node is not the home node, GPSR is able to find out the the home node and then it returns the datum stored on that node.

In the case the faulty node is the home node, the way in which the nodes are selected by the dispersal protocol guarantees that the nodes that store datum D are the Q closest nodes to point $(x, y) = h(M)$. In this way, when the GPSR routes the request to the (x, y), if is not able to find the old home node (due to its fault), it is still able to find the new home node (that was the second closest node), because it was part of the the Q closest nodes. In this way, our protocol guarantees that a datum is still reachable until the at least one of the Q nodes storing the datum is alive.

Enhanced GPSR. Q-NIGHT uses a slightly modified version of GPSR. Usually, when GPSR is in the *perimeter mode*, it always adopts clockwise turn to reach the destination coordinates. This behavior leads to pathological situations as the one shown in Figure 4.31.a.



(a) Standard perimeter mode

(b) Enhanced GPSR perimeter mode

Figure 4.31: GPSR routing perimeter mode

The right hand rule makes the packet traverse all the perimeter before reaching the destination node. This is not a problem for GHT as the datum is replicated on all the perimeter nodes, but may be very inefficient for Q-NiGHT , which replicates only on a ball surrounding the destination.

In our GPSR version, we turn clockwise or counterclockwise depending on which is the closest path to the destination, as shown in Figure 4.31.b. Let s_a be the position of the sender node, c the position of the center of the deployment area and s_d be the position of the destination. We turn clockwise if $0 < \widehat{s_a c s_d} < 180$[‡], and counterclockwise otherwise.

4.2.3 How much does Q-NiGHT cost?

In order to understand the cost that we must pay to achieve QoS and non-uniformity resistance in our protocol we have to compute the cost of both GHT and Q-NiGHT. The metric used to measure the cost of both GHT and Q-NiGHT is the number of messages used to perform `put`s and `get`s. We choose to use the number of messages metric because it is the same metric used by the original GHT paper (Ratnasamy et al., 2003) and thus, to have a direct comparison with the original paper.

In our experiments, we aimed to provide both the mean and the variance of number of used messages. We perform 2000 `put`s and 2000 `get`s with randomly chosen meta-data. Both `put`s and `get`s requester nodes are chosen uniformly on the whole sensors set.

The QoS for Q-NiGHT is again pure replication with 15 replicas for each datum. We always consider RNG networks because the results in the case of GG networks are almost the same and do not introduce considerations different from the ones that we make here in our study.

In all graphs, the x axis shows the sensor density in the network and the y axis the operation cost measured that is the average number of messages used by the two protocols.

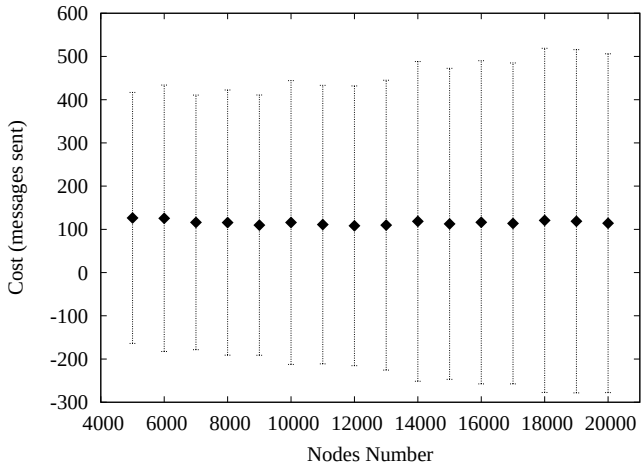
Figure 4.32 and Figure 4.33 show respectively the results of our experiments for the evaluation of the `put`s and of the `get`s in the case of networks distributed accordingly to the uniform distribution.

Figure 4.34 and Figure 4.35 show respectively the results of our experiments for the evaluation of the `put`s and of the `get`s in the case of networks distributed accordingly to the Gaussian distribution.

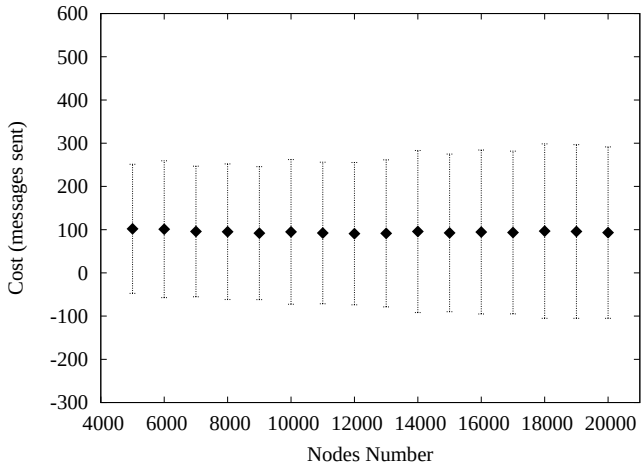
Figure 4.36 and Figure 4.37 show respectively the results of our experiments for the evaluation of the `put`s and of the `get`s in the case of networks distributed accordingly to the Hill distribution.

The first thing that we notice, comparing Figures 4.32–4.33 with Figures 4.34–4.37 is that the cost (or better, the convenience) of Q-NiGHT with respect to the

[‡] Computed in clockwise way



(a) GHT



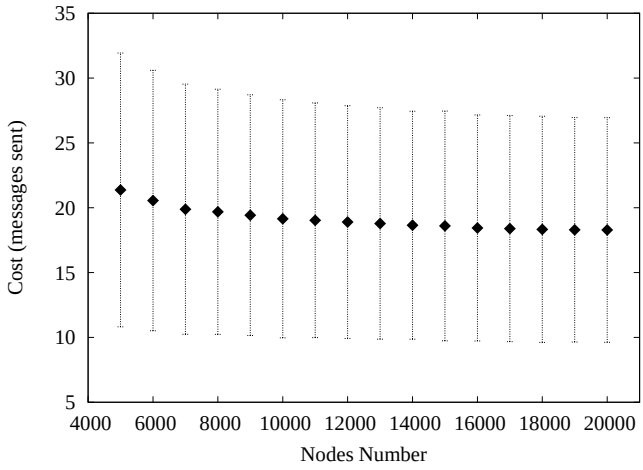
(b) Q-NiGHT

Figure 4.32: Mean and standard deviation of the costs (number of messages) of `put` *with uniform distribution and RNG planarization.*

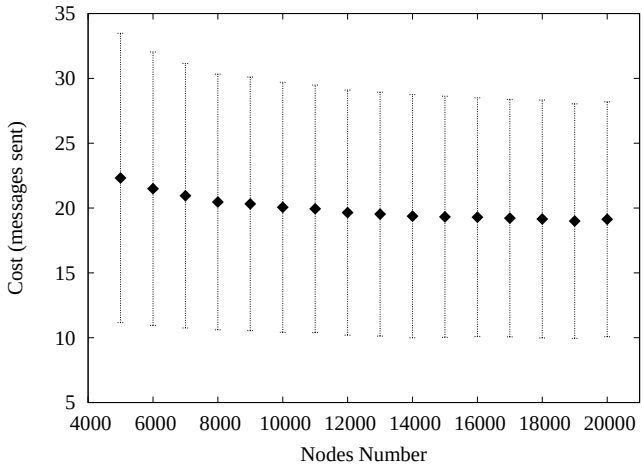
cost of GHT is different is we consider the uniform case and the non-uniform case. Now, we study this two situations separately.

Uniform case. In the uniform case, shown in Figure 4.32 and in Figure 4.33, shows that in both Q-NiGHT and GHT have comparable cost in the case of `get` operation, but in the case of `put` operation, the performance of Q-NiGHT is better than the performance of GHT.

The reason of this behavior is that in the case of the `put` operations, the cost of GHT is influenced by the use the external perimeters to store data. This is more



(a) GHT

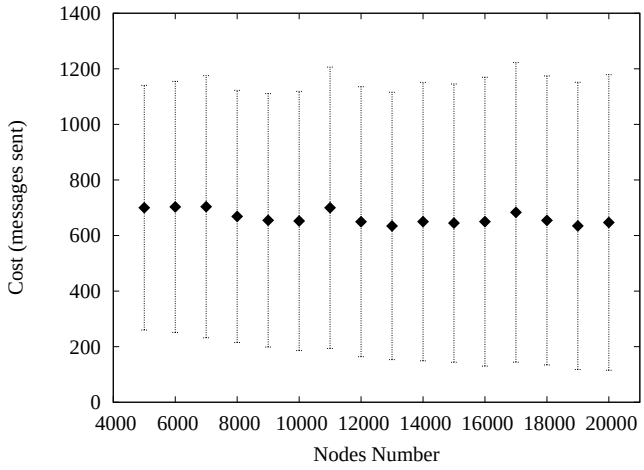


(b) Q-NiGHT

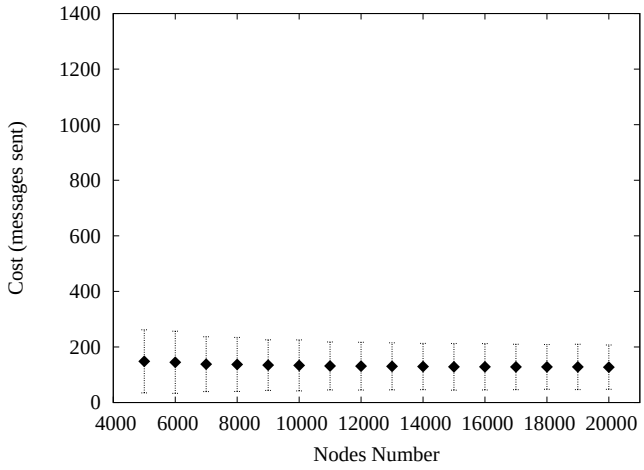
Figure 4.33: Mean and standard deviation of the costs (number of messages) of `get` *with uniform distribution and RNG planarization.*

evident if we look at the variance of the cost of the protocol. While the mean is quite similar in both cases, the variance in the case of GHT is very large with respect to the one of Q-NiGHT.

In the case of the `get` operations, the cost of Q-NiGHT is a little larger than the cost of GHT. In Q-NiGHT, once the home node is found, data is stored in the ball surrounding the home node. In GHT, once the home node is found, data is stored on all the nodes in the perimeter around the point (x, y). When Q-NiGHT needs to retrieve a datum, it needs to reach the perimeter around (x, y), and then it



(a) GHT



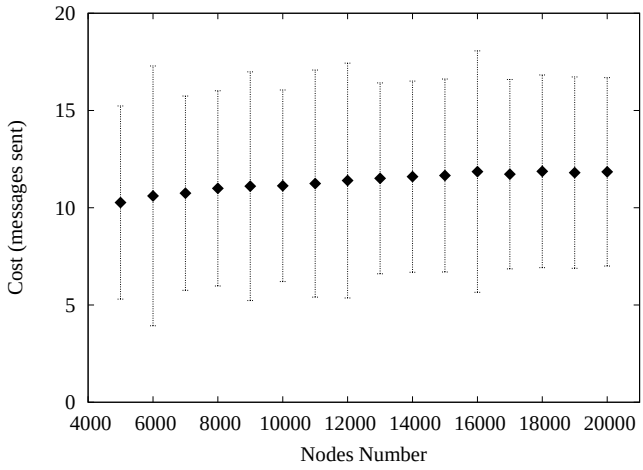
(b) Q-NiGHT

Figure 4.34: Mean and standard deviation of the costs (number of messages) of `put` *with Gaussian distribution and RNG planarization.*

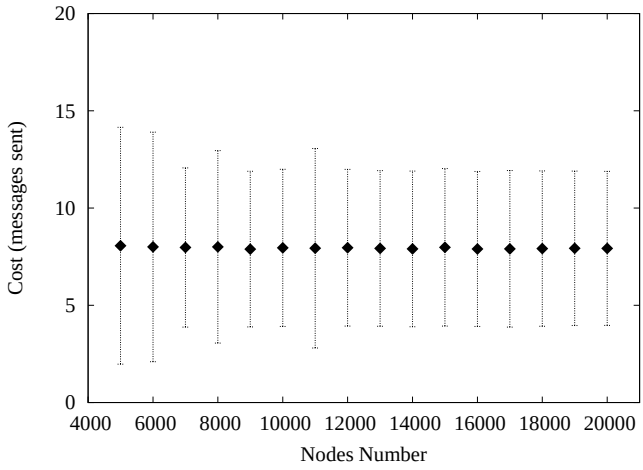
has to be routed around such perimeter until it finds a node in the ball that stores the datum. In GHT, the query simply needs to reach the perimeter to retrieve the datum.

Non-uniform case. In the non-uniform case, Q-NiGHT shows better performances than GHT for both `put`s and `get`s.

In Figure 4.34 and in Figure 4.36, we can see the cost of the `put` operation for both Gaussian and Hill distributions. In both cases, GHT has a performance that



(a) GHT

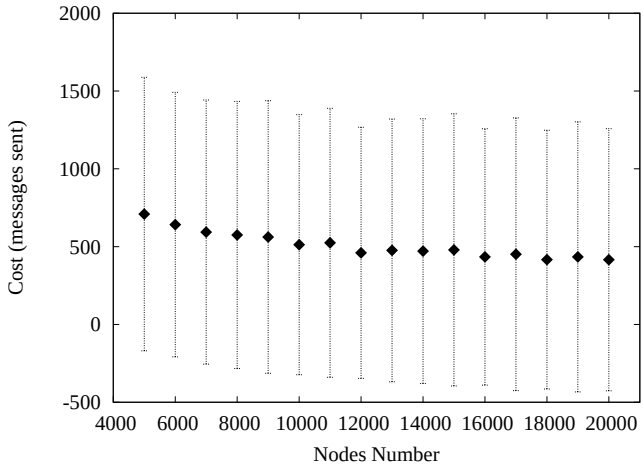


(b) Q-NiGHT

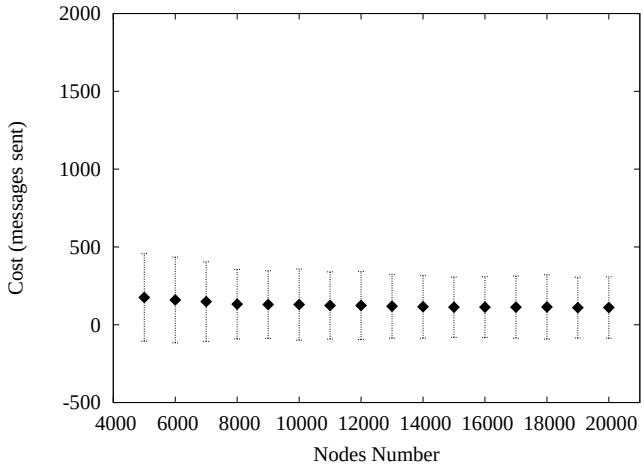
Figure 4.35: Mean and standard deviation of the costs (number of messages) of `get` *with Gaussian distribution and RNG planarization.*

is 4 to 6 times worst than the case of Q-NiGHT. This behavior is due to the bad interaction between the use of the uniform hashing and the use of the perimeters to store data: the uniform hashing distributes, with the same probability, data in denser and sparser regions, moreover, in sparser regions perimeters are longer than in denser regions. On the other hand, Q-NiGHT uses the dispersal protocol to store data and the dispersal protocol guarantees that we need only a small number of messages to be exchanged to store a datum, once the home node is found.

In Figure 4.35 and in Figure 4.37, we can see the cost of the `get` operation for



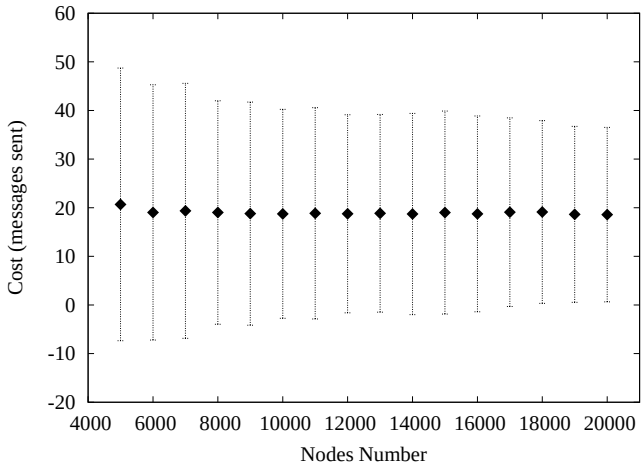
(a) GHT



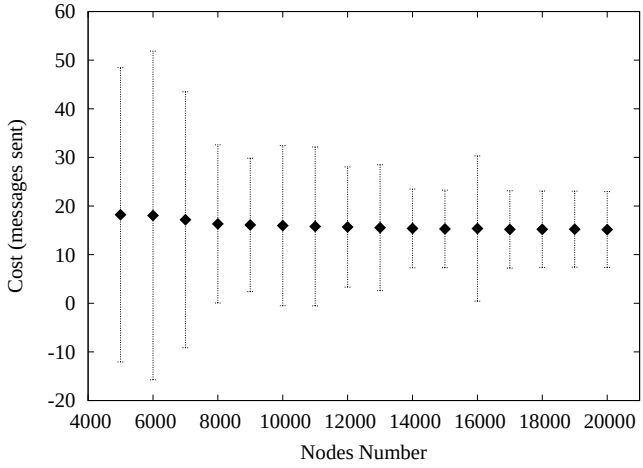
(b) Q-NiGHT

Figure 4.36: Mean and standard deviation of the costs (number of messages) of `put` *with Hill distribution and RNG planarization.*

both Gaussian and Hill distributions. In both cases, Q-NiGHT has a better performance with respect to GHT. This is due to the use of `RejectionHash`. Using `RejectionHash`, Q-NiGHT stores the largest part of data in denser regions. Storing the largest part of data in denser regions has two benefits: (*i*) the majority of requests for data arrives from denser regions too and in turn of it, the requesters tend to be closer to the nodes that store data and (*ii*) the greedy mode used to route messages goes though dense regions very fast because nodes are closer. On the other hand, GHT stores data uniformly on all areas (both dense and sparse). When



(a) GHT



(b) Q-NiGHT

Figure 4.37: Mean and standard deviation of the costs (number of messages) of `get` *with Hill distribution and RNG planarization.*

the nodes need to get a datum, that are more probably inside denser region, they need to perform longer paths. The need to perform longer paths is mitigated by the use of the perimeters that enable a large number of nodes to store a datum and thus the paths are not much longer than in Q-NiGHT.

This last set of experiments, as well as the others that helped us into the analysis of GHT and into the design of Q-NiGHT, show how the non-uniformity can influence a widely accepted system such as GHT. Moreover, these experiments show that it is possible to provide new solutions able to deal with non-uniformity and that these

solutions do not introduce a large overhead in uniform networks.

4.3 A Q-NiGHT based application for location management

In this section, we describe an application of Q-NiGHT. In this application, we deal with another kind of non-uniformity, namely the functional non-uniformity.

In the functional non-uniformity, a network is made up of different kind of nodes. Some of these nodes a simple sensing nodes, while others are nodes that are able to provide services to the sensing nodes (the *servers*).

In this scenario, the sensor nodes must be able to locate the servers to use the services that they provide. To enable the sensors nodes to locate the servers, we use Q-NiGHT. The servers use Q-NiGHT to store their position inside the network. The servers use the ID of the service that they provide as the meta-datum that is hashed to find the point (x, y) where they will store their own position.

When a sensor node needs to use a service, is hashes the service ID and query perform a `get` on the point $(x, y) = h(ID)$ to retrieve the position of the server. Then, the sensor contacts the server to use its service.

4.3.1 Heterogeneous wireless sensor networks and location management

Heterogeneous WSNs are made up of various kinds of nodes. Some nodes are used as the interface to the physical environment (we will call them *sensors*). Other nodes act instead as *servers*, providing services to the other nodes. For instance, in an outdoor intrusion detection application, the sensor are scattered randomly to provide tracking of possible intruders, while more powerful nodes provide the service or connection (e.g., through satellite links) to the end user.

In this section, we define an architecture for enabling efficient discovery of services for the sensors that need them. Servers are organized in two tiers. The first tier comprises the actual servers (such as the satellite up-link enabled nodes mentioned before). Then there is an intermediate server tier (front-end) where some common nodes (sensor) are chosen for storing the current position of the servers. A sensor that needs to find a specific service (and hence a server), sends a message to the front-end of the system and gets back the position of the service. At this point the sensor knows the server position and is able to start using the service needed.

The proposed architecture implements strategies to provide both load distribution and fault-tolerance, as described in the rest of the section. It also makes extensive use of a caching system to speed-up requests and to save energy. The localization/communication infrastructure used as the basis for our architecture is Q-NiGHT.

4.3.2 System architecture and operations

In this section we present our two-tier architecture for locating servers efficiently and in fault tolerant way.

We start by presenting the *actors* that play important roles in the architecture. These roles are both *structural* (given by the physical nature of the heterogeneous network) and *logical* (given the different usage of the same kind of nodes to perform different tasks).

Then, we present the operations of *service registration* and *look-up* that are provided by the system. The service registration procedure is performed by the servers to communicate their position to the nodes of the network. The look-up procedure is executed by the nodes that require the localization of a service.

4.3.2.1 Actors

There are three categories of nodes in our architecture. The *sensors*, the *servers*, and the *server locators*.

Sensors: the sensors are low power and low cost devices that are equipped with special equipment to control their surrounding environment. They also sport a CPU for performing simple computations, and an embedded radio to communicate with each other.

Servers: the servers are special nodes that are capable to provide some service to the sensor nodes. These services range from storage (to keep the sensed data) to perform as gateways between the WSN and the users. Each server $server_i$ provides one or more services that are identified by a name, for instance $service_j$.

Server locators: the server locator nodes are sensor nodes that know the servers, i.e., the services that the servers provide and the server locations in the network.

4.3.2.2 Two-tiers servers architecture

The sensors are the clients of the proposed architecture. The architecture *per se* is formed by two tiers. The *back-end* of the architecture is made up of the servers that are able to provide services to the clients. The servers are randomly deployed (as the clients) and need to be localized by the server locators. The server locators form the *front-end* of the architecture. The nodes requiring a service query the front-end to have the position of the the server, or servers, providing that service. Once obtained this information, the nodes communicate directly with the back end servers.

4.3.2.3 Services localization

Two operations implement service localization: *server registration* and *server discovery*. The first one is used by the servers to make the server locator aware of their location. The second one is used by the node that needs a service for finding the corresponding server. These two operations use Q-NiGHT `put` and `get` operations. `put` is used for storing the position of a server and `get` to retrieve it later.

4.3.2.4 Servers registration

During the network set-up phase the server i determines its position, $position_i$, and registers it with the server locator nodes. To perform such operation each server hashes the name of each of its services and determines the corresponding point (x, y). At this time, for each provided service, it performs a `put` of the pair $\langle service_j, position_i \rangle$ to the point (x, y) by using Q-NiGHT. The nodes that store the pair $\langle service_j, position_i \rangle$ become the server locator nodes for $service_j$.

4.3.2.5 Servers discovery

When a node needs $service_j$, it hashes the service name by using the Q-NiGHT hash function, finding the point (x, y). Then it performs a `get` operation of $service_j$ from point (x, y), i.e., it sends a request toward that point. One of the server locators replies with the position of the server (this operation is called a *look-up*). In addition to the basic Q-NiGHT `get` operation, at this time the node caches the position of the service/server for future use. It then sends the request for $service_j$ to the server. The use of caches improves the whole look-up process in many ways. First of all, faster replies are provided to those other nodes interested in locating the same server whose look-up queries pass through the caching node. Moreover, caching enables cheaper look-ups because fewer hops can be enough to provide a reply, and lower energy consumption for the server locator nodes is required since they have to deal with a lower number of queries. For instance, cached positions increase information retrieval performance in applications such as intrusion detection. In this case, messages for a particular server are generated by nodes that are close to each other and to where the intrusion happens. In this case, spatial locality of caches is taken advantage of.

4.3.2.6 Load distribution and fault-tolerance

By using Q-NiGHT with the strategy described above, our architecture is able to balance the query load to the locator servers and it is tolerant to location servers failures. The load balancing property is particularly useful for distributing multiple requests of the same service to multiple servers that provide it. Fault-tolerance helps in removing those servers that are no longer available (because of failures or

network disconnections) from the list of servers that provide a given service. The disappearance of a service/server can be signaled to a server locator by a sensor node who, trying to contact a server, realizes that it is no longer available. This feature of the architecture presents security issues, which we discuss in a separate paragraph at the end of this section.

Load balancing. Query load balancing is provided via multiple server registrations. All the servers that provide a service have the same server locators. This happens because the servers share a common service name, say $service_j$. The server locators store all the coordinates that were provided for each service name. When a request arrives to a server locator for $service_j$, the node chooses one of the possible servers according to a given strategy (for instance, randomly, or in a a round-robin way, by keeping a pointer to the last server returned and incrementing it modulo the number of servers). This method also provides an easy way to increase the number of servers. When a new server (that provides $service_j$) enters the network, the server registers itself with the server locators and these return the server location as one of the possible servers for that service.

Fault-tolerance. Fault-tolerance to service outage is obtained as follows. The servers keep providing their position to the server locators periodically (for instance each τ seconds).

In case of service failure, after τ seconds from the last update the server locators cancel the server position. In the worst case this system provides the cancellation of a server from server locators after 2τ seconds.

In order to make our protocol completely fault-tolerant we have to remove the cached server positions from the caches of the sensors that stored such information. To this aim, a server position is cached by a sensor for at most τ seconds, after which it is removed from the cache. In case the sensor needs the position of the server again after τ seconds, it will have to query the server locators again. Finally, if a node queries a server whose entry was in its cache (i.e., τ seconds from its last query to a server locator have not passed yet) and the server is no longer available, the server's (ex) neighbors report an error to the sensor requesting the service. Upon receiving the error message the node removes the cache entry and performs a `get` for a new server for $service_j$, at the same time communicating to the server locators that the server is unavailable.

Security issues. The capability of sensors to invalidate a service location at a server locator makes possible attacks in which the server locators server list are modified by malicious (or faulty) nodes.

To address this problem the network user (administrator) can choose between three solutions: (i) the sensors are not allowed to invalidate server locators and/or invalidation messages are dropped by server locators, (ii) the server locator verifies the invalidation querying the server itself to double check about the availability of

that server, or (*iii*) the user provides a cryptography based system to verify the identity of the message sender and possibly its *trust-level*. All these solutions are equally powerful to guarantee a minimum level of trust to our architecture. The choice of one of them (or any combination of them) depends on the characteristics of the network (e.g., computational power and energy) as well as the application and environmental characteristics (probably, for border monitoring or in the battlefield a level of trust much higher than wildlife monitoring is required).

4.3.3 Experimental results

We have performed experiments for measuring the effectiveness of our service localization architecture with respect to the energy cost of querying with and without caches, as well as the cost of the look-up operation.

In the simulation setting, we have considered 5000 WSNs where 5000 sensor nodes are scattered randomly and uniformly in a square area with a side of 1000m. Each node has communication range of $30m$. Power consumption for transmission is set to $24mW$ and that for reception is set to $14.4mW$ as in the EYES sensor prototypes (Havinga et al., 2003). The sensors that perform a look-up operation and then send a message to the server are uniformly chosen between the deployed sensors.

All the experiments are aimed at showing the effectiveness of the architecture in providing prompt and energy efficient response to sensor queries. In particular, we show here that caching is particularly useful in providing a more balanced energy consumption, and therefore better performance of the network. For this reason, all the presented experiments are provided with and without nodal cache enabled. All tests are performed starting from the same seeds to generate the same scenarios with different architectural parameters. The results we show achieve a statistical confidence of 95%, with a precision within 5%.

Figure 4.38, Figure 4.39 and Figure 4.40 depict the cost for a sensor to contact server locators and servers. In Figure 4.38 the cost is defined as the energy spent by a node to send a packet to the server locators, to get the server location back and then to perform one communication to the server. In other words, we compute the total energy to deliver/receive three packets. In Figure 4.39, the cost is defined only as the cost to send a message from a sensor to the server locators and to get the server location back (that is the energy to send two packets). This provides up with a more detailed idea of how much it costs to the sensor the use of the intermediate tier provided by the sensor locators. Figure 4.40 shows the cumulative cost of the look-up operation, to have an idea of difference in the growth of the energy cost. The cumulative cost is computed as follows: the cost of the qth look-up is given by its cost and summed to the cost of all the previous $q-1$ look-ups. As mentioned, each set of experiments is performed with and without the cache mechanism enabled. The network is observed for a time long τ to take into account the maximum usage

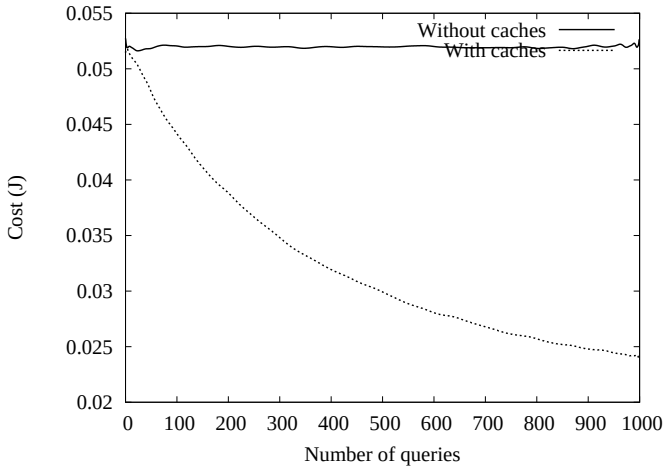


Figure 4.38: Cost for server look-up plus the cost for contacting the servers. The figure shows the effectiveness of the caches in our solution. With the growth of the number of the queries the cost of the protocol decreases because the nodes caching the wanted information grow up.

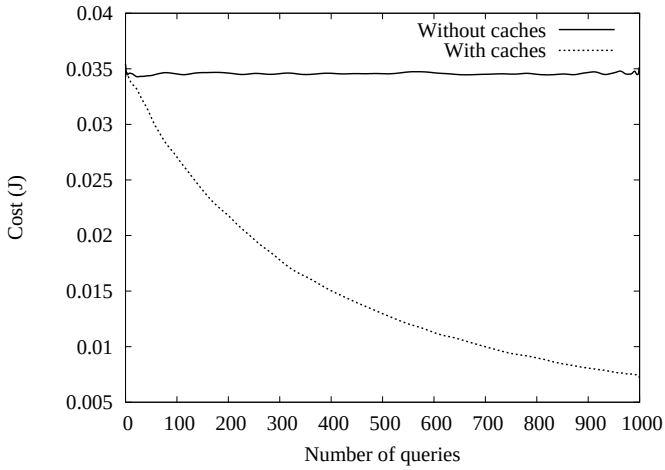


Figure 4.39: Cost for servers look-up operations only. The figure shows in detail the effectiveness of the caches in our solution (considering that the server contacting operations cannot be cached in our model).

of the caches before their refresh. In our experiments, $\tau = 30$ minutes i.e., the time that we need to perform 1000 queries.

Figure 4.38 depicts the cost for each single query in the network with and without the caching enabled. This cost, expressed in Jules, is defined as the sum of the energy spent at each node for propagating the query. This cost is computed considering both the cost for transmission and reception We observed that when caching is enabled the cost of the single query decreases with increasing number of queries because the

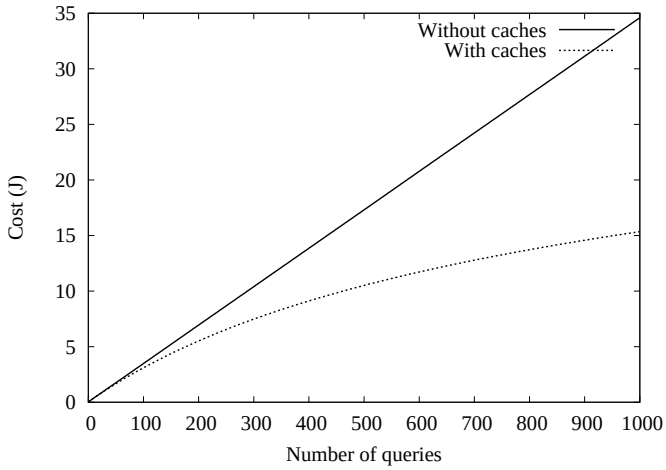


Figure 4.40: Cumulative cost for servers look-up operations. The cumulative cost for the qth look-up is given by its cost and summed to the cost of all the previous $q-1$ look-ups.

caching mechanism becomes more and more effective (more and more nodes have the location in cache).

Figure 4.39 depicts only the cost of the look-up operation for each single query in the network with and without the caching enabled on sensors. This cost, as the previous one, is defined as the sum of the energy spent at each node for propagating the query. This case, is analyzed to have a better evaluation of the look-up procedure, that is cached optimized, with respect to the server interaction, that in our scenario does not use caches to optimize the sensor-server. Some particular applications can use caching also between sensors and servers but we chose this situation (the worst case) in which the interaction with the server is not cached to have a more clear vision of the look-up costs and benefits.

Figure 4.40 depicts the cumulative cost of the look-ups only (sensor-server locators communication and back) without considering the cost for server interaction. In this case, we observe that the cost to reach the server locator nodes decreases when the number of the queries increases, as expected.

Figure 4.41 presents the (normalized) residual energy level of the server locators with and without using caches. The residual energy level of the server locator is computed reading the energy level of the server locators before the first refresh of the service location by the servers, that restore Q copies of the location, also in the case in which some server locators run out of energy. The figure shows a high influence (36%) of the caches in the energy spent by the server locators in their function.

As shown in the experimental results, the presented architecture provides the localization service in a load-balanced and fault-tolerant way and the caching system

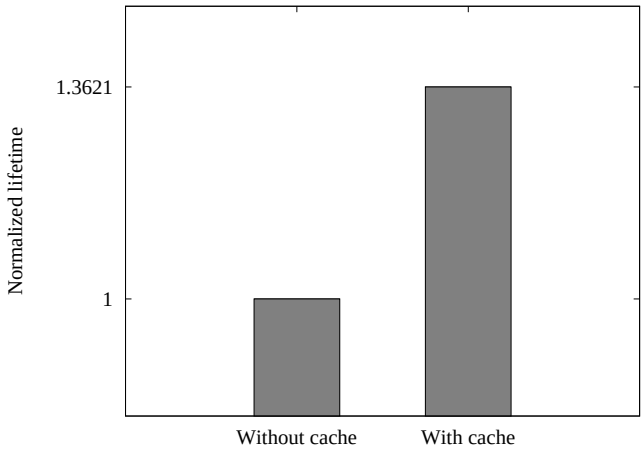


Figure 4.41: Server locators residual energy level before the server locators refresh step. The residual energy is normalized to the energy level of the system without using the caches.

enables the nodes that lay in the same region to assist in providing service location while relieving the service locators of providing the service. Furthermore, the obtained results are encouraging and open up possibilities for further studies and for the application of this method to problems such as data storage and replication, and similar problems.

4.4 Summary

In this chapter, we have presented our research work in the data management field in WSNs.

We presented our contribution, describing *Q-NiGHT* (Albano et al., 2006a,Albano et al., 2006b,Albano et al., 2007). Q-NiGHT is a data centric storage system based on GHT. As well as GHT, Q-NiGHT uses an hash function to provide compute the position of a datum starting from its mata-data (often called *key*). Moreover, Q-NiGHT uses a modified version of the GPSR protocol to route messages inside the network. The key differences between the Q-NiGHT and GHT is in the assumption made by the protocol to work. GHT assumes that the network is uniform and the hash function used to find the position of a datum is always the same, independently from the real distribution of the nodes. On the other hand, Q-NiGHT uses the RejectionHash algorithm distribute data in the network with a distribution that is an approximation of the network distribution.

Our work focused into the analysis of GHT, throughout simulations and analysis of the experimental results, spotting out its weaknesses in both-uniform and non-uniform environment. Then, we provided a solution that could overcome the

problems introduced by a bad initial design and the uniformity assumptions. Q-NiGHT showed up its flexibility and its ability to work properly in non-uniformly distributed networks. The extensive simulations performed on Q-NiGHT showed up that all the improvements aimed at providing fair load distribution and QoS features to GHT have a cost and that this cost is not very high if compared to the advantages in terms of load distribution and memory resources usage.

Finally, we described and evaluated the performances of a possible application scenario involving Q-NiGHT. We provide a fault tolerant and load-balanced localization system for services in WSNs. In our scenario, we enable a network made up of different types of nodes (some of them are normal sensors and some others are servers) to locate the services provided by the servers and thus to locate the servers in the network. The system provides a simple solution based on a two-tiers server system in which some of the sensors act as server-locators storing the position of the services and the position of the servers providing them. The server-locators are elected using Q-NiGHT: during the network initialization phase, the servers hash the name of the services that they provide and store their position in the point returned by the hash function. This operation defines a set of server-locators around that point. When a sensor needs a service it simply hashes the service name, sends a query to the server-locator position and as soon as a replay arrives it begins communicating with the server. This approach is totally generic and can be used to locate any kind of service provided in a simple and reliable way.

In all this chapter, we used Q-NiGHT pretending to know the correct network distribution functions, for instance inside the RejectionHash function. This assumption is not realistic but it made simpler the description of Q-NiGHT. Obviously, we have also addressed the problem of finding out the distribution of a WSN, which is the central problem for non-uniformly distributed networks. Our research lead to a system, that we called Stripes. This system is described into the next chapter.

Chapter 5

Stripes: Finding Out Distributions

I'm ten years old. My life is half over and I don't even know if I'm black with white stripes or white with black stripes.
- Marty the Zebra (Madagascar, 2005)

Abstract

In the last chapter, we described Q-NiGHT leaving outside our exposition the way in which our system was able to know the network distribution. In this chapter, we present Stripes. Stripes is a suite of protocols and algorithms that can be used to find out the network distribution from a fixed number of samplings. Moreover, Stripes optimizes energy usage of the network enabling cached information across the network. Stripes is divided in two equally important parts: two network density approximation algorithms, that starting from a limited number of samplings rebuild the density of the whole network and two protocols that are used to sample the density in the network and to spread the computed approximation of the density to the nodes of the network that need such density in normal operations.

In the last chapter, we revised Q-NiGHT. Our description of Q-NiGHT introduced the RejectionHash algorithm to provide data location during load and store operations. The RejectionHash algorithm needs to know the distribution of the nodes (or an approximation of it). We deliberately postponed the discussion on how Q-NiGHT is able to know the network distribution.

In this chapter, we move into the description of the protocols that Q-NiGHT uses to know the distribution of the nodes in the network. These protocols are called Stripes and they are proposed to provide an efficient way to reconstruct the sensors' distribution starting from a few samples. This protocols family is divided in two equally important parts.

The first part rebuilds an approximation of the density of the network starting from few samples using a two steps algorithm: the first step builds a first approximation of the density and the second step performs a refinement step that provides a much better approximation of the network density.

The second part focuses on the dissemination of the sampled densities and of the density approximation to all the nodes that need it. We proceed using the following strategy: we start from a very simple (and quite inefficient) broadcast algorithm, then we move to an *on-demand* strategy well suited for networks in which few nodes need to know the network density and, finally, we provide a very efficient protocol to distribute such information also in networks with a high number of requests.

A note about simulations. In this chapter, as well as in Chapter 4, we make a large use of simulations to undertand the non-uniformity influence upon networks and to measure the efficience of Stripes. Again, our simulation environment is a self-made simulator that is able to deal with large quantity of nodes and all the experiments are repeated until we achieve a statistical confidence of 95%, with a precision within 5%.

Chapter organization. This chapter is organized as follows. Section 5.1 presents the motivations behind the Stripes protocols family. Section 5.2 describes how Stripes protocols family works and the performance and cost of such protocols. Finally, section 5.3 draws the conclusions of the research work described in this chapter.

5.1 Why do we need Stripes?

Non-uniformity introduces new problems that must be addressed in WNSs research. A non-uniform distribution presents a natural unbalance, for instance in radio interferences, and solutions that well fit the needs of a sub area of the network can be useless in another. Moreover, as we have seen in Chapter 4, non-uniformity can be a great source of load unbalance for data storage inside a network.

On top of all the problems brought by the non-uniformity, there is the problem of *finding out* the distribution (or an approximation of the distribution) of the network itself.

This problem is central in the development of non-uniformity resistant solutions. For instance, without such knowledge Q-NiGHT is unable to perform storage load distribution in the network throughout the RejectionHash algorithm.

5.2 Stripes

In this section, we present Stripes, which finds out the nodes distribution of a WSNs starting from a fixed number of samples. Stripes, and it is made up two

orthogonal, and equally important parts.

These two parts are equally important because each of them plays a fundamental role into the finding out the network density. The algorithm that rebuilds the density from the samples is crucial to merge up all the samples and to produce the approximated density. As important as the algorithm used to rebuild the density are the protocols used to sample the network and to move the computed density around because they must perform such operation in very efficient way. The efficiency of such protocols is essential because STRIPES is a family of "support" protocols and they must be used as basements to build up non-uniformity resistant protocols.

Moreover, we designed the two parts to be orthogonal. The algorithm used to rebuild the density does not need to be used with the protocols presented here to get the samplings from the network: it is able to work with any other protocol able to provide density samplings from the network. In a similar way, the protocols used to sample the density of the network can be used with any other algorithm able to rebuild the density from samples and not only with the one presented here.

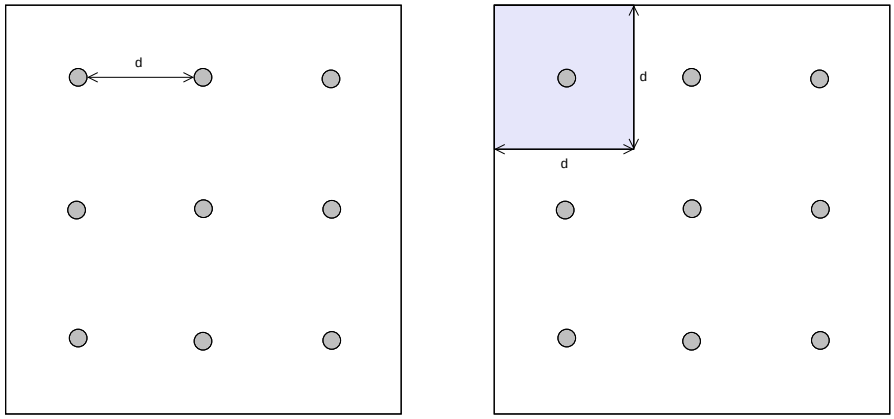
5.2.1 Building blocks

Let us suppose that one node needs to know the WSN density to perform some task. In this case, the node can do three things: (*i*) it can guess the density of the network, (*ii*) it can wait for some other node to communicate the density of the network or, (*iii*) it can query the network to find out the density of the nodes. The first solution is not really interesting but it is widely used in current protocols and moreover, the guessed distribution is the uniform one.

In the rest of this chapter, we focus on (*ii*) and (*iii*). We call (*ii*) a *passive* solution and (*iii*) an *active* solution.

Both solutions can appear in two different versions: the first one, in which the density is considered *static* and the other, in which the density is *dynamic*. In the case of static density, once the density is computed it is considered to remain the same for all the network lifetime. In the case of dynamic density, the density is computed various times during the lifetime of the network. In the dynamic case, the lifetime of the network can be divided in *epochs*. At the end of each epoch all the data used to estimate the density of the network is considered obsolete and our solutions must find out again the density using new data. The length of an epoch is system dependent and may vary according to the network usage, environmental conditions and so on. In the rest of the section we analyze the algorithm and the protocols in the static case because the dynamic one is only a repetition of the static one in different epochs.

Watch-points, sentinels and sampled density. Both the active and the passive solutions use the idea of *sentinel*. A sentinel is a node that is able to provide information about the density in the region it belongs to. Sentinels are related to another concept: the *watch-point*.



(a) Watch-points

(b) Area of interest of a watch-point

Figure 5.1: Watch-points and the areas approximates by the watch points. (a) shows the grid distribution of the watch points in the deployment area. (b) shows the watch-area belonging to the watch-point in the top left part of the deployment area.

A watch-point is a coordinate pair in the deployment area. Watch-points are distributed in a grid fashion in the network area and represent locations around which we want to sample the density of the network (Figure 5.1.a). The watch-points are distant d from each other. The value of d depends only on the number of watch-points that we want to use to approximate the network density and the size of the area in which the nodes are deployed. We use the density measured around a watch-point to approximate the density in a larger area (the *watch-areas* depicted in Figure 5.1.b). These sub-areas are squared and non overlapping, with side length equal to the distance between watch-points (d) and centered on watch-points.

As stated before, a sentinel is a node that is able to provide information about the density in the region it belongs to. Each watch-area has one sentinel and the sentinel is the closest node to a watch-area's watch-point.

We now discuss how each watch-area can elect its sentinel node. When the network turns on all the nodes check their position. If the distance of a node from a watch-point if less or equal than $r/2$ (remember r is the communication range), then the node can be a possible sentinel for that watch-area.

As depicted in Figure 5.2, where the watch-point is identified by the star symbol and the candidate sentinel is the dot, if a node is at most at distance $r/2$ from a watch-point it is able to communicate to all the other possible candidates and the sentinel election procedure is very simple: at this time each possible sentinel sends to its neighbors its own position and ID. The node closest to the watch-point is the sentinel. If two nodes are at the same distance from a watch-point, the one with higher ID is the sentinel.

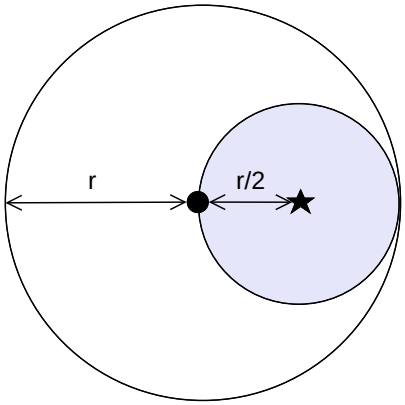


Figure 5.2: Sentinel and watch-point distance relation with respect to the communication range r.

Once the sentinel node is elected, the sentinel node counts the nodes inside its communication range. The number of the nodes inside the sentinel's communication range is the *reported density* for the corresponding watch-point.

In the case in which no node close is as close as $r/2$ to a watch-point, the watch-point is said to be *unguarded* and the reported density for its watch-area will be 0.

There is an important relation between d and r: it is must be always true that

$$d \geq 3r$$

this is because the sentinel counts how many nodes belongs to the circle or radius r centered on the sentinel itself. Since a sentinel can be as far as $r/2$ from its watch-point, having $d < 3r$ can bring nodes from different areas in the sentinel circle.

Thus, if we want a sentinel to report only the nodes inside its own watch-area, the watch-area must be greater than the maximum possible distance from a watch-point, at which a sentinel is able to find, and count, nodes. This implies that the watch-area side must be greater than $2(r + r/2) = 2(3r/2) = 3r$.

Figure 5.3 shows the relationship between d and r. From the picture it is clear that a smaller d implies that on sentinel can use, to estimate the density, nodes that belong to other sentinels and/or watch-points.

5.2.2 Moving density around: broadcast, Stripes and Fat-Stripes

In this section, we discuss how sampled densities from watch-points can be delivered to other nodes of the network.

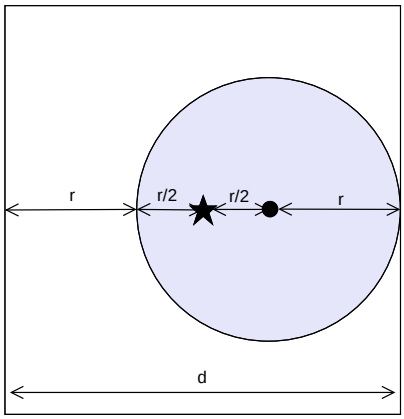


Figure 5.3: Relation between d and r. d must be greater or equal than 3r to guarantee that a sentinel reports only the nodes inside its own watch-area.

As stated before, there are two different solutions to address the problem of informing the nodes inside the network of its distribution: the passive solution and the active solution.

In passive solution, each sentinel broadcasts the network density that it found out in its watch-area to all the nodes inside the network. In the active solution, the nodes that need to know the network density query the sentinels to know the density inside each watch-area.

In this section we revise this two solutions. Moreover, for the active solution, we provide two different protocols: STRIPES and FAT-STRIPES. STRIPES is the first implementation of the active solution that uses caches to optimize the energy usage. FAT-STRIPES is an optimization of STRIPES. FAT-STRIPES uses the caches in a more aggressive way to save even more energy.

5.2.2.1 Passive solution

In the passive solution, a network finds out its own density during a bootstrap phase.

When the network turns on, all the nodes check their position and for each watch-area sentinel are elected. Once the sentinels election procedure is done and after that each sentinel has collected the number of nodes to be reported the sentinels *broadcast* to every node in the network the pair $\langle watch - point_{xy},\ number\ of\ nodes \rangle$.

In the case of unguarded watch-areas, no sentinel is elected and no message is broadcasted. Thus, the absence of messages from a watch-point is interpreted as *zero* in the corresponding watch-area. To ensure a correct termination of the network set up operation, a time out is used in order to prevent that nodes wait for a density value coming from a particular watch-point forever.

After the reception of the data from the sentinels and after the time out expired, all the nodes start to rebuild the density of the network according to the algorithm that will be described in Section 5.2.3.

The cost of the passive protocol (in terms of energy used to communicate the density) is given by the cost of k broadcasts operations, where k is the number of sentinels inside the network.

5.2.2.2 Active solution (Stripes)

On the other hand, the active solutions, each node that needs to know the network density queries the network to have such value. The query operation is performed sending a message to a watch-point. The sent message will reach the closest point to the watch-point, that is the sentinel. The sentinel, in turn of the reception of the query sends back the sampled density to the requester.

In the active protocol, there are two different cases to consider: (i) the node that queries the network to know the density is the first one attempting to do so and (ii) the node that do so it is not the first one. We consider the two cases.

A node that needs to know the distribution of the sensors in the network, sends a message, using GPSR (Karp and Kung, 2000), to each watch-point. The GPSR protocol finds the closest node to the watch-point (that must be a sentinel if it is closer than $r/2$ to the watch-point). In response to the query, the sentinel sends back to the querier the pair $\langle watch - point_{xy},\ number\ of\ nodes \rangle$.

If the closest node found by the GPSR perimeter mode, to the watch-point is much far than $r/2$, it sends back the pair $\langle watch - point_{xy},\ 0 \rangle$.

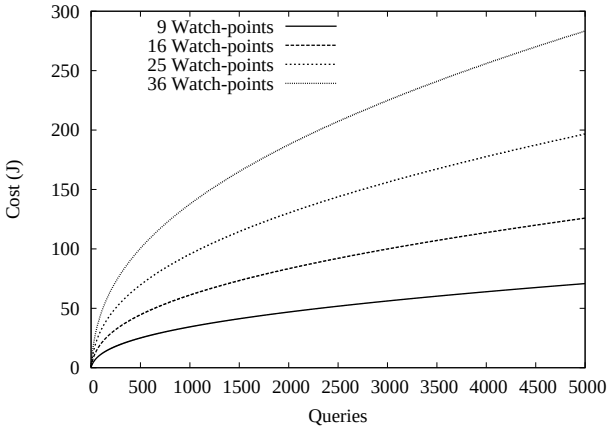
After the reception of the data from the sentinels the node starts to rebuild the density of the network as described in Section 5.2.3.

In order to improve subsequent queries from nodes traversing the same path, on the way back to the requester, the pair $\langle watch - point_{xy},\ number\ of\ nodes \rangle$ is cached on all the nodes belonging to the path, creating cache *stripes* across the network (a stripe for each watch-point).

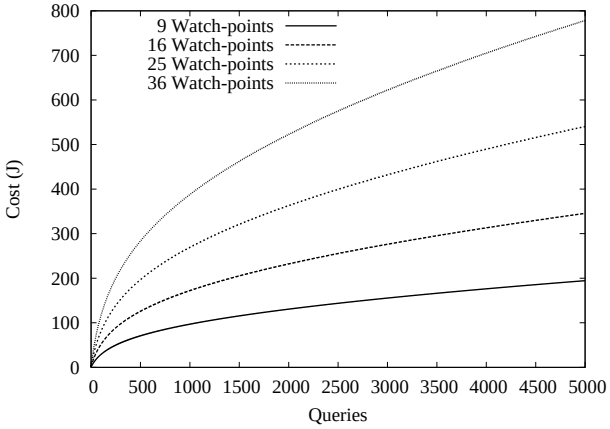
Consider now what happens when a node queries the watch-point after many others have done so. As soon as the query message reaches a node belonging to a cache stripe in a path belonging to a previous query, a reply is sent directly by the node belonging to the stripe without needing to reach the watch-point. If a node belonging to a stripe needs to know the density of the network, it simply queries the watch-points for which it has no cached data.

The idea behind the stripes is that, after some initial queries, a node is able to find all the data it needs at a few hops of distance from itself. Due to the presence of cache stripes we call this protocol Stripes.

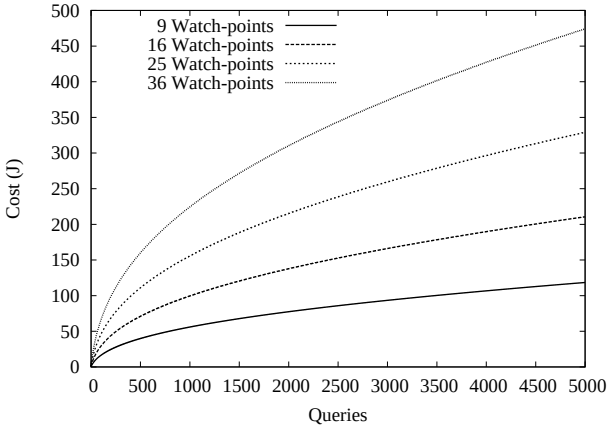
In order to understand the if our intuition is correct and to measure the real cost of Stripes we have performed detailed simulations of it. To measure the cost



(a) Uniform



(b) Gaussian



(c) Hill

Figure 5.4: Cumulative cost of the STRIPES *protocol.*

we compute the energy (in Joules) spent by the network to collect data from the watch-points.

The simulations have the following parameters. We simulate the deployment of 5000 nodes in a squared area of size $1000m \times 1000m$. Each node has a communication range of $30m$. We use 9, 16, 25 and 36 watch-points placed in a squared grid fashion. We distribute nodes using uniform, Gaussian (with $\sigma = 1$ and average in the center of the deployment area) and Hill distributions (stretched to fit the $1000m \times 1000m$ area). The transmission power is set to $24mW$ and the receive power to $14.4mW$ as in (Havinga et al., 2003). The idle power is set to 0 because we are interested only in the *raw* cost of our protocols, without considering the energy spent in idle state. For each test we generate 5000 requests to find the density: a request for each node of the network. The order in which the nodes are elected to send the requests is an uniform permutation of the whole nodes set.

Figure 5.4.a, Figure 5.4.b and Figure 5.4.c show the total energy spent by the network. These values are presented in a cumulative way: the cost of the qth query is given by its cost added to the cost of all the previous $q - 1$ queries.

As we can expect, the total cost of the protocol increases with the number of queries, but the curve is less step as the number of queries increases. This is due to cache effect on the length of the routes that must be traversed to find the value to be returned. The first time that a query is performed it must be routed as close as possible to the watch-point. The second time a request can stop before reaching the watch-point because the request has hit a cached density value on a node that is part of a stripe. This effect increases with the number of queries because more and more nodes are part of the cache. Moreover, the total cost increases with the number of watch-points because more queries must be performed in order to get all the densities from all the watch-areas.

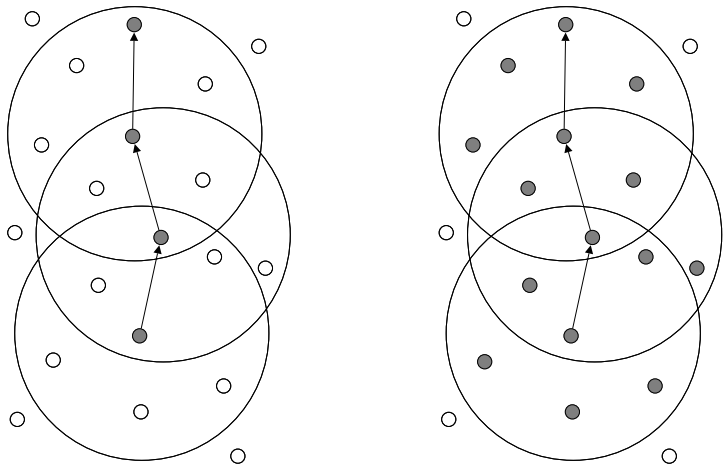
5.2.2.3 A better active solution (Fat-Stripes)

The STRIPES experimental results show that the use of caches is able to reduce the cost of the queries to get the densities from all the watch-areas.

However, we can observe that cache stripes produced by the STRIPES protocol are "thin" stripes. As depicted in Figure 5.5.a, the only nodes that belong to the stripes are the one used as relay nodes by the GPSR protocol on the way back to the node that required the density.

A major problem with this approach is that all the nodes that do not actively cooperate to move data, but still receive the packets spend their energy into the reception of the message without having any benefit.

This is the key observation for an improvement of STRIPES called FAT-STRIPES that works in the same way of STRIPES, with only one difference: the nodes that passively receive a $\langle watch - point_{xy},\ number\ of\ nodes \rangle$ packet and that are not involved actively into the communication cache the datum too.



(a) Stripes (b) Fat-Stripes

Figure 5.5: Nodes that cache the pair $\langle watch - point_{xy},\ number\ of\ nodes \rangle$ on the way back of a query.

As depicted in Figure 5.5.b, the cache stripes produced by the Fat-Stripes protocol are now larger, including all the passive receivers ant the corresponding cache stripes have become "fat". The aim of this little improvement is to increase the number of nodes that do *not* need to communicate to query the network.

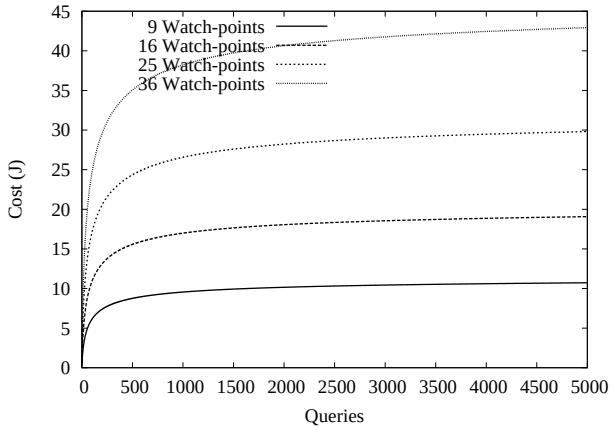
In order to measure the effectiveness of this improvement, we simulated again the same networks of Section 5.2.2.2.

As did for Stripes, or each test we generate 5000 requests to find the density: a request for each node of the network. The order in which the nodes are elected to send the requests is an uniform permutation of the whole nodes set.

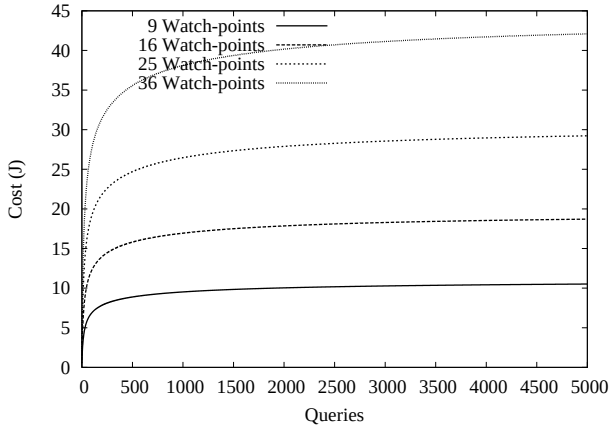
Figure 5.6.a, Figure 5.6.b and Figure 5.6.c show the total energy spent by the network. These values are presented in a cumulative way: the cost of the qth query is given by its cost and summed to the cost of all the previous $q - 1$ queries.

The total cost of the protocol increases in a slower way with respect the Stripes protocol. This is due to the presence of larger stripes. Larger stripes contain nodes that do not need to communicate to know the density of the network because this knowledge derives from listening packets that carry that information.

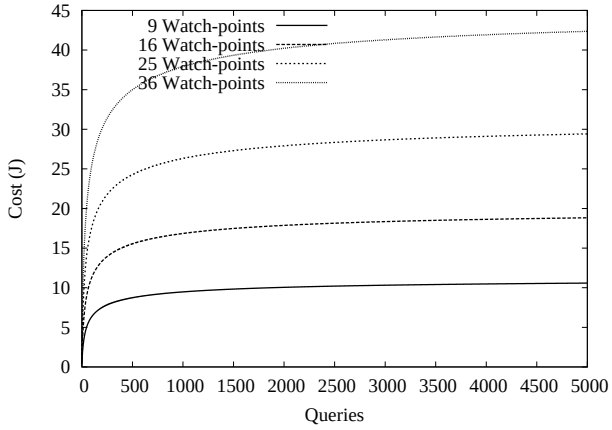
The performance improvement of Fat-Stripes over Stripes is very high: In the uniform distribution case Fat-Stripes uses less than the 13% of the energy used by Stripes, in the Gaussian distribution case Fat-Stripes uses less than the 6% of the energy used by Stripes and in the Hill distribution case Fat-Stripes uses less than the 10% of the energy used by Stripes. The advantage of Fat-Stripes over Stripes is more evident in non-uniform distributions because in non-uniform distributions we have areas in which the nodes are much closer to each other. The



(a) Uniform



(b) Gaussian



(c) Hill

Figure 5.6: Cumulative cost of the FAT-STRIPES *protocol.*

closer nodes increase the effect of the "fat" stripes because a lot of nodes receive and cache data without being active relays in the communication. As well as Q-NiGHT, Fat-Stripes is able to use the non-uniformity of the network as an advantage.

From these results it is evident that the little modification in nodes behavior introduced by the Fat-Stripes protocol with respect to Stripes, brings a very big improvement in the protocol's communications cost without increasing the complexity of the original Stripes protocol.

5.2.2.4 Stripes vs. Fat-Stripes

The performance advantage of Fat-Stripes with respect to Stripes is very evident and it was discussed in Section 5.2.2.3. Fat-Stripes overcomes Stripes in every cases, requiring a very small number of messages, due to the use of caches.

On the other hand, the use of the caches represents an increase in the memory usage of the nodes. Every node that receives a sampling message stores the sample, also if the node will never use it. It is even true that the memory usage to store the samplings is very limited and with a constan upper bound given by the number of the watch-points times the size used to represent the value of a sampling.

From the viewpoint of a network designer, the size used to used to represent the value of a sampling is upper bounded by the number of the nodes inside the network. For instance, if the maximum number of the nodes that a user can deploy is as large as N, the *bits* needed to store the value of a single sampling can be as large as $\log(N) + 1$: $\log(N)$ bits are effectively used to store the cache value and the extra bit is a flag used to tell if the cache line is a valid value (i.e., a cached sampling) or a missing value (i.e., a value that is unknown and that must be queried). The same flag bit is used to invalidate cache lines in the of dynamic densities at the end of each epoch.

Thus, if we have K watch-points points, the maximum size, in bits, used for the cached samples is $K \log(N) + K$. For instance, in the case of a network made up of 100000 nodes and with 25 watch-points, the size of the cache is 450 bits (i.e., 57 bytes).

5.2.3 Density rebuilding algorithm

In both passive and active protocols, once a node gets the number of nodes at each watch-point, it must rebuild the density of the network, creating a bucket density structure in which each bucket represents a watch-area.

Here, we propose a simple algorithm to rebuild density. The algorithm acts in two steps: the first step rebuilds the network density as the the number of nodes reported in each watch-area by the sentinels over the total number of nodes reported by all the sentinels and the second step adjusts such values performing a weighted mean of each watch-area with the neighbor watch-areas.

In the first step (the pseudo-code is presented in Algorithm 3), we reconstruct the density of each bucket as the fraction of the reported nodes in that bucket over the total number of nodes reported.

Algorithm 3 REBUILDDENSITY(W)

Require: The set W of messages from watch-points/sentinels.
Ensure: The set D of rebuilt densities.

```
tot ← 0
for all w_{i,j} ∈ W do
  tot ← tot + w_{i,j}.nodes
end for
for all d_{i,j} ∈ D ∧ w_{i,j} ∈ W do
  d_{i,j} ← w_{i,j}/tot
end for
```

Algorithm 3 acts as following. The number of nodes reported in each watch-area $w_{i,j}$ are summed up and stored into tot. Then, the approximated density of each watch-area $d_{i,j}$ is computed as the fraction of the nodes reported in such area $w_{i,j}$ with the total number of nodes tot. In practice, the algorithm assumes that the total number of the nodes inside the whole network is only the total number of the reported nodes and that the number of nodes in each watch-area is the only the one reported by each sentinel.

This approximation can bring to two pathological situations: (i) false zeroes and (ii) over reported nodes. The false zeroes are reported by areas that have very sparse nodes, or the nodes are not deployed near the watch-point (for instance in a corner), and the possible sentinels can be too far from their watch-point and then a zero is reported when nodes are present. Over reported nodes refers to the opposite problem. In the case of the over reported nodes, we have that the majority of the nodes in a watch-area are concentrated near a watch-point and an extra-large number of nodes is reported.

Figure 5.7 depicts the problem of false zeroes. If we consider the watch-point in the shaded watch-area, it reports only few (or zero) nodes due to the fact that the watch-point is located in a low density part of the watch-area. This value is not indicative of the real quantity of nodes in the watch-area because the majority of the nodes are located near the left border of the watch-area.

To overcome these two problems, we perform the second step. In the second step (the pseudo-code is presented in Algorithm 4), we apply a smoothing step to the density computed by the REBUILDDENSITY algorithm: the computed density for each watch area is substituted by the weighted average of watch-area density itself with the computed densities of its neighbors watch-areas.

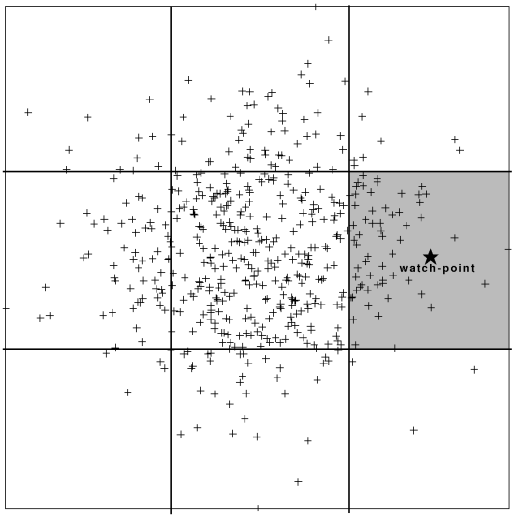


Figure 5.7: False zeroes problem.

Algorithm 4 SmoothDensity(D)

Require: The set D of densities.
Ensure: The set D' of smoothed densities.

for all $d_{i,j} \in D \wedge d'_{i,j} \in D'$ **do**
  $d'_{i,j} \leftarrow$ *weighted mean of* $d_{i,j}$ *and its neighbors watch* $-$ *areas*
end for

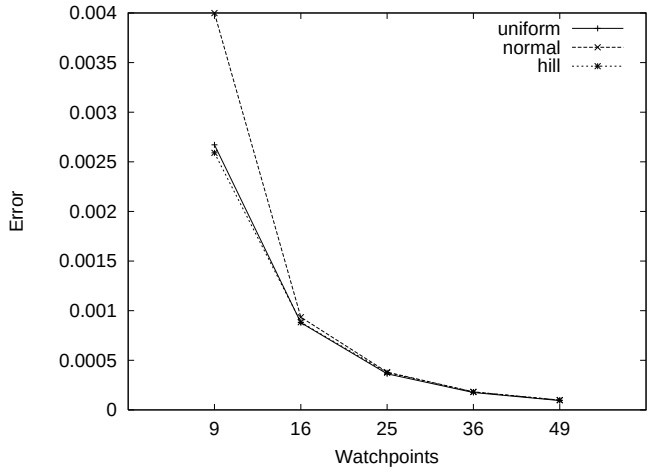
Algorithm 4 works as follows. For each computed density $d_{i,j}$, the algorithm computes the weighted mean of $d_{i,j}$ with the ones computed into the 4 neighbor watch-areas (i.e., $d_{i+1,j}$, $d_{i-1,j}$,$d_{i,j+1}$ and $d_{i,j-1}$). For the areas on the border the 3 neighbor watch-areas and for the ones in the corners the 2 neighbor watch-areas. In the weighted mean computation, the value $d_{i,j}$ is multiplied by the number of neighbors watch-areas used to compute the average.

We evaluate the RebuildDensity and of the SmoothDensity algorithms using simulations.

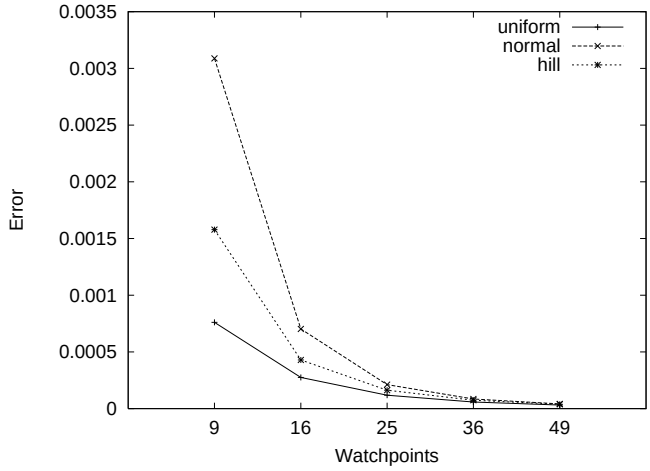
The simulations have the same parameters of the simulations presented in Section 5.2.2.2.

We measure the "fit" of the approximation computing the *error* in the computed density with respect to the real one. To compute the error, for each watch-area, we compute the square of the difference between the real percentage of nodes in the watch-area and the one computed by our protocol. The error is defined as the average of the squared of the differences computed for all the watch-areas composing the network.

The results, depicted in Figure 5.8, show, for each one of the analyzed distributions, the average of the errors computed for each simulation. We present the average error for both the Algorithm 3 alone and Algorithm 3 and Algorithm 4 together.



(a) Error after the execution of the RebuildDensity algorithm



(b) Error after the execution of the SmoothDensity algorithm

Figure 5.8: Average error in approximation algorithm. (a) is the error introduced by the in the RebuildDensity *algorithm and (b) is the error after the smoothing step performed by the* SmoothDensity *algorithm.*

The smoothing step, performed by Algorithm 4, is always effective and lowers the error in all the cases. However, its effect is more evident when the protocol uses few watch-points, because a lesser number of watch-points sample very few nodes with respect to the real number of nodes into the network and the probability to report false zeroes or over reported nodes is larger. In all cases, the error is very low. As the number of watch-points increase, the error decreases, and the smoothing step is,

in turn, less effective, as expected.

From an implementation point of view, the two proposed algorithms are very simple and can be easily implemented on a sensor. Both algorithms use only sum, multiplication and division operations that are usually provided by current processors hardware.

5.2.4 Comparative cost of active and passive protocols

In this section, we compare the costs of the broadcast, the STRIPES and the FAT-STRIPES protocols to complete our evaluation of the proposal.

Our comparative evaluation is divided into two parts. The first part provides an analytical study of the protocol cost and the second one is an experimental study of such costs.

In the analytical study of the cost, we model both the broadcast and a simplified version of the STRIPES protocol (without using caches) to have a comparative order of size between a generic passive protocol and a generic active protocol. The analytical study of the cost measures the cost of the protocols in terms of *sent messages*.

In the experimental study of the cost, we perform simulations of all the protocols and we compare them to study the behavior of the protocols with uniform, Gaussian and Hill distributions. The experimental study of the cost measures the cost of the protocols in terms of energy used by the network.

5.2.4.1 Analytical study of the cost

Before going into detailed simulations, we want to explore this issue theoretically. We assume a simplified setting. The first assumption is related to the fact that we intend to study networks that are deployed using a *fair* distribution. The characteristics of a fair distribution are drawn into the following definition:

Definition 5.2.1 (Fair distribution)**.** A node distribution f is *fair* if and only if

1. $\forall A_i$ s.t. $A_i \in$ domain of f and A_i is non null, $\int_{A_i} f(x,y)\mathrm{d}x\mathrm{d}y > 0$.
2. The diameter of the deployed network is close to $\sqrt{n}$, where n is the total number of deployed nodes.

□

The first part of the definition states that the nodes *could* be distributed in all the deployment area. The second part states that the distribution function of the network is not too step.

For instance, the uniform distribution is a good example of fair distribution, but the Gaussian distribution is not.

The study takes into account, for the passive solution, a "good" broadcasts algorithm, that is one that forward the data only once for each node (with a total number of exchanged messages comparable to n, the size of the network), and for the active solution does not take into account the effect of the cache stripes.

Let us suppose that in the passive solution all the k sentinels broadcast their data. The total cost of this operation is

$$C_p = kn. \tag{5.2.1}$$

Equation (5.2.1) shows that the cost of the passive protocol is proportional to the cost of the k broadcasts needed for each sentinel value.

In a *fair* distribution, the cost of the active solution, for each request, is

$$C_a^{single} = 2k\sqrt{n}. \tag{5.2.2}$$

Equation (5.2.2) shows that the cost of the active protocol for each request is given by the cost of sending a message to each sentinel that, in the worst case, is at the largest distance in the network (its diameter) times the number of sentinels. We must multiply this value by two because, for each query, we need a reply.

The total cost of the active solution depends on the number Q of queries that are performed

$$C_a = \sum_{i=1}^{Q} C_a^{single} = 2kQ\sqrt{n}. \tag{5.2.3}$$

Equation (5.2.3) shows the total amount of messages sent considering that the network needs to query Q times the sentinels to know the density.

In the static case, to measure the convenience of a protocol with respect to the other, we must find when

$$\begin{aligned} C_p &< C_a \\ kn &< 2kQ\sqrt{n}. \end{aligned}$$

The previous relation is satisfied when

$$Q > \frac{\sqrt{n}}{2} \tag{5.2.4}$$

Equation (5.2.4) shows that, in the static case, the passive solution is more convenient than the active one if the number of queries that the protocol will need is larger than $\sqrt{n}/2$.

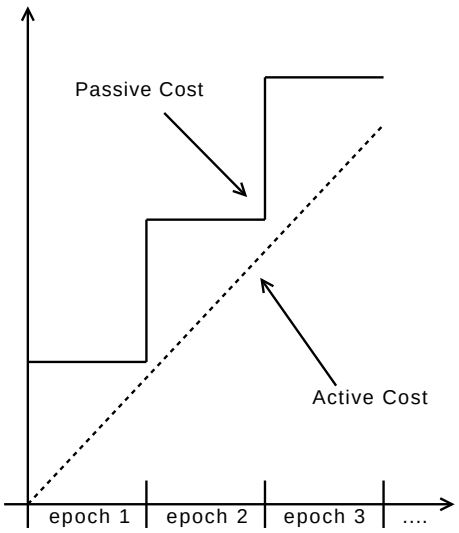


Figure 5.9: The convenience of the active protocol with respect to the passive protocol in the dynamic case.

In the dynamic case, this relation is not valid for all the network lifetime, but it does hold for each epoch. In this case, we can find a number of queries Q', for each epoch, such that the active protocol is always convenient with respect to the passive one.

As depicted in Figure 5.9, in the dynamic case the cost of the passive protocol increases at each epoch of the same quantity kn creating a steps-like graph. On the other hand, the active protocol increases constantly with time, independently from the epochs. Thus, in the dynamic case, if the number of queries performed by the protocol, Q', in each epoch is less than $\sqrt{n}/2$, the cost of the active protocol is always smaller than the cost of the passive protocol.

5.2.4.2 Experimental study of the cost

This last set of results compares the cost of the active protocols with the cost of the passive one. Our aim is to spot out the differences between the protocols presented and to understand the reasons of these differences. Moreover, we compare the experimental results with the analytical ones to understand better how the caches can influence the behavior of both Stripes and Fat-Stripes.

We expect a big difference between the experimental results and the analytical ones due to use of real distributions and caches. Real distributions can be not as *fair* as assumed in the theoretical model and caches play an important role in the cost of the protocols.

In order to compare the tree protocols, we performed simulations with the same parameters used in Section 5.2.2.2.

Also in this experiment, for each test we generate 5000 requests to find the density: a request for each node of the network. The order in which the nodes are elected to send the requests is an uniform permutation of the whole nodes set.

The results that we present in Figures 5.10–5.12 are structured as follows: for each distribution (uniform, Gaussian and Hill) and for each number of watch-points, we compare the costs, in terms of energy spent by the passive solution, by STRIPES and by FAT-STRIPES.

On the x axis of the charts, we have the number queries and on the y axis, we have the energy spent by the network. The graph of the energy is presented in a cumulative way: the cost of the qth query is given by its cost added to the cost of all the previous $q-1$ queries. The cost of the passive solution is independent from the number of the performed queries, thus it is represented by an horizontal line.

Figures 5.10.(a–d) show the comparative results in the case of uniform distribution for different numbers of watch-points. In the case of uniform distributions, we can observe that the total cost of the STRIPES protocol (after 5000 queries) is twice the cost of the passive one because we have that for each point we need to pay for the query and for the replay. The effect of caches is very evident here. Equation (5.2.4) suggests that the limit of convenience in the use of STRIPES protocol is $\sqrt{n}/2$, that for $n = 5000$, is approximatively 35. In our experiments, this value is raised to values near 1000. The cache effect influenced the protocol efficiency by a factor of 28.

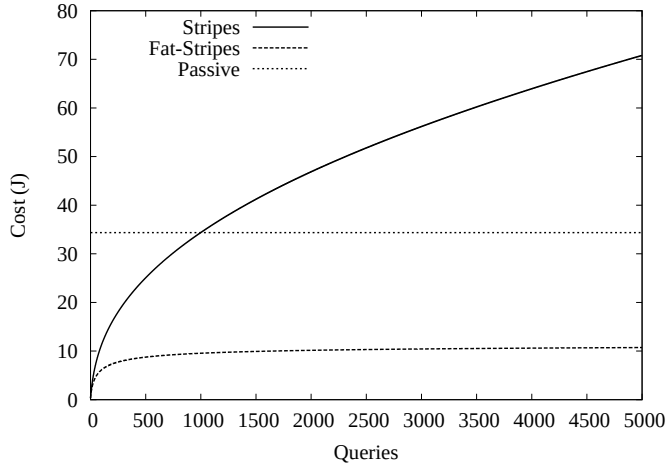
The effect of caches is even more evident in the FAT-STRIPES protocol. The protocol cost is lower than the cost of the other two protocols. This is due to the use of large cache stripes. In the passive protocol, each node sends the sampled density one time and all the nodes can receive it multiple times, each time paying the cost of receiving a message. The STRIPES protocol caches values only on relay nodes: all the nodes that are not in the relay path still spend their energy receiving a message without the benefit of caching useful information. The FAT-STRIPES protocol caches values in both relay nodes and nodes that passively received replay messages: these last nodes apply the principle that if they spent energy in receiving a message, it is convenient to cache information brought by the message, because such knowledge could be used in the future.

Figures 5.11.(a–d) show the comparative results in the case of Gaussian distribution for different numbers of watch-points. In the case of Gaussian distributions, we can observe that the total cost of the STRIPES protocol (after 5000 queries) is more than the double of the passive one. This is due to the fact that Gaussian distribution is not as *fair* as uniform distribution. Some watch-points are in sparser regions of the network (near the borders) and then they have no sentinels. In the case in which the watch-point is in a border region of the network, GPSR must route the query all along the external perimeter of the network. This fact increases the total cost of the protocol that grows faster than in the uniform case. Also in this case, in despite of the fact of the distribution is not *fair*, cache stripes do an excellent job. The STRIPES protocol is better than passive one still until 250–300 queries are

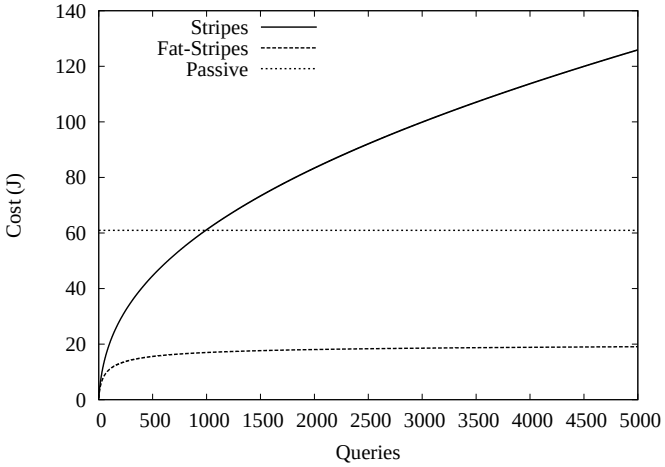
performed, that is a factor 7 with respect to the theoretical result. As in the uniform case, the FAT-STRIPES cost never grows to the level of the passive protocol nor to the level of the STRIPES protocol.

In the case of Gaussian distribution, the external perimeter affects negatively the cost of STRIPES. On the other hand in FAT-STRIPES, such long perimeters becomes an advantage because a lot of sensors, belonging to the external perimeter of the network, become caches for the replays to all the queries directed outside the network: less nodes will need to query the network for the density and a cache ring, around the external perimeter, is able to cache replays for more watch-points.

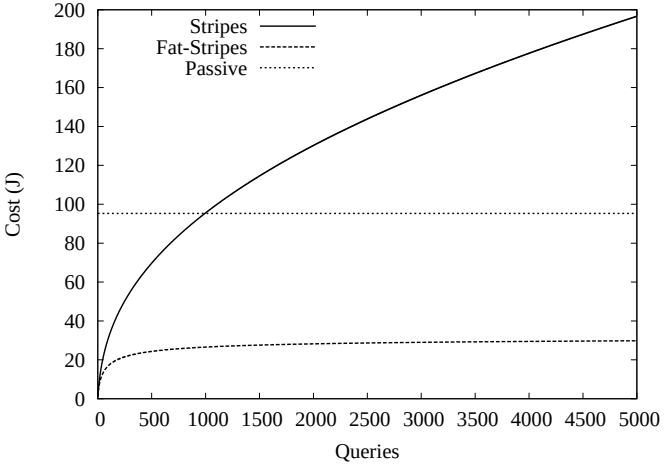
Figures 5.12.(a–d) show the comparative results in the case of Hill distribution for different numbers of watch-points. This case is in middle between the uniform and the Gaussian case. This is due to the fact that Hill distribution is *fairer* than the Gaussian one, but less fair than the uniform one. With the Hill distribution cache stripes work better than Gaussian but worse than uniform. The active protocol advantage still until 600–700 queries are performed, that is a factor 17 with respect to the theoretical result. As in the uniform and Gaussian cases, the FAT-STRIPES cost never grows to the level of the passive protocol nor to the level of the STRIPES protocol. In this case, the advantage of external perimeters is less evident in FAT-STRIPES protocol because the distribution is more *fair* than the Gaussian distribution.



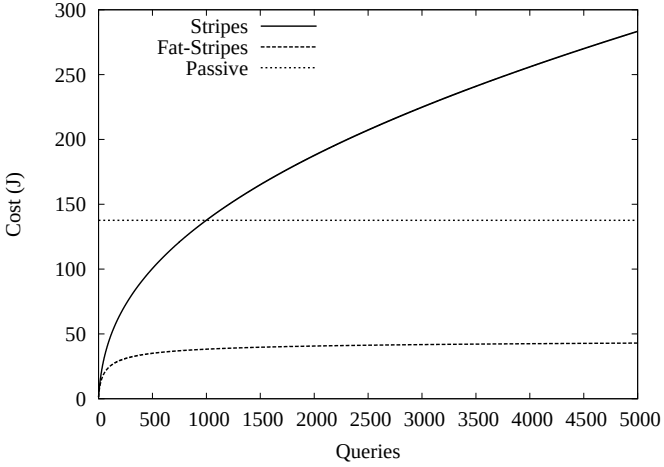
(a) 9 watch-points



(b) 16 watch-points



(c) 25 watch-points



(d) 36 watch-points

Figure 5.10: Comparative cost of the protocols (uniform case).

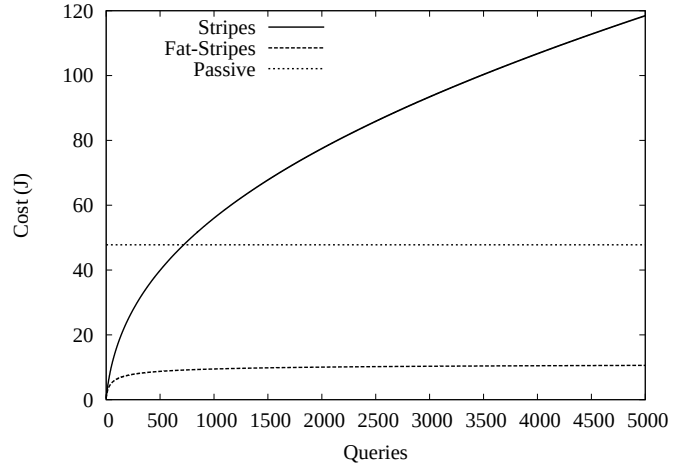
(a) 9 watch-points

(b) 16 watch-points

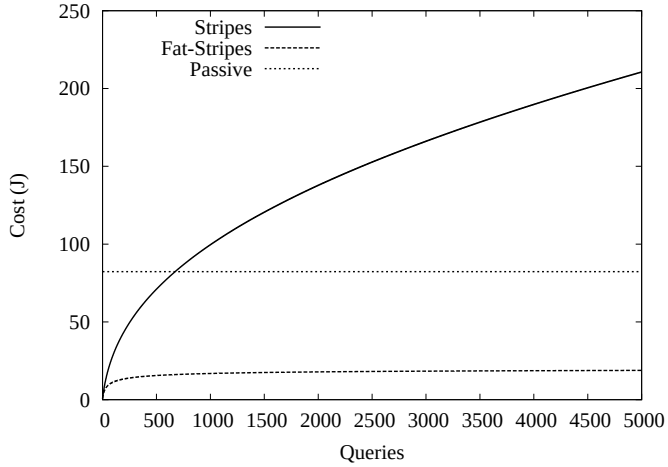
(c) 25 watch-points

(d) 36 watch-points

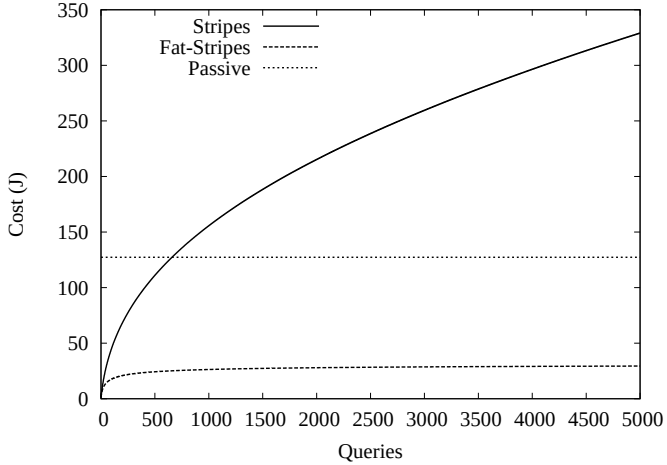
Figure 5.11: Comparative cost of the protocols (Gaussian case).



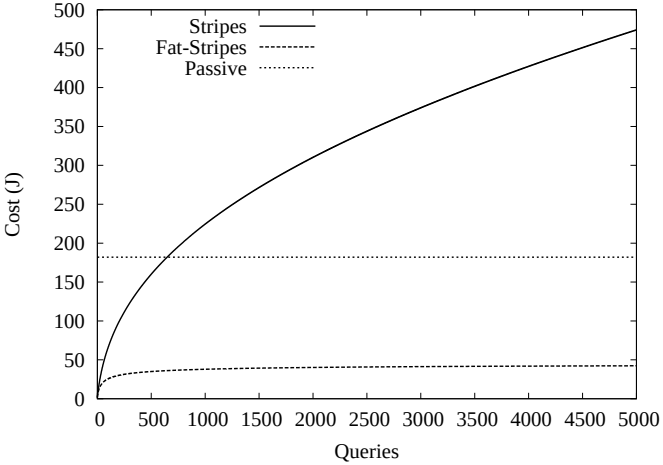
(a) 9 watch-points



(b) 16 watch-points



(c) 25 watch-points



(d) 36 watch-points

Figure 5.12: Comparative cost of the protocols (hill case).

5.3 Summary

In this chapter, we focused in finding out the distribution of a network of sensors.

Finding out the density of a network is the most important problem in non-uniform networks. When we drop the uniformity assumption and we want to deal with non-uniformly distributed networks, we must find new protocols and systems able to provide the intended services independently from the distribution of the network. Some of these systems, such as Q-NiGHT, need to compute the distribution of the network to provide fair load distribution a good resources usage.

In this chapter, we studied both the problem of density reconstruction starting from a predefined number of samples, providing a simple but effective algorithm (and its refinement step too), and the protocols to enable the sensors of the network to compute the distribution of the nodes.

For this second point, we started from a very simple timed-out broadcast protocol that spread the sampled density to all the nodes of the network. Then we moved to a more sophisticated density-on-demand protocol, namely Stripes, that is very efficient in the case of a low traffic network but it is not so good in high traffic networks.

To overcome the problem of Stripes with high traffic networks, we provided a new protocol (Fat-Stripes) that, using an opportunistic caching strategy, is able to provide density-on-demand in a very efficient way in all traffic conditions.

Conclusions

End? No, the journey doesn't end here.
- Gandalf (Lord of the Rings)

This is the final chapter of our thesis and now, we are ready to draw the conclusions of our research work.

In Section C.1, we describe the path throughout the thesis that moved us from the introduction to its conclusion. In Section C.2, we draw a conclusion motivating the choices in the development of this thesis. Finally in Section C.3, we present the possible future work.

C.1 How did we arrive here?

In the Introduction, we pointed out how the networks are the hidden building block of our world. We presented this concept describing various kinds of networks. We started from *natural networks*, showing that nature itself has a network structure (e.g. preys/predators structure). Then, we moved to *human networks* pointing out how our society and its unique features (e.g. nations) are indeed networks. It was a little step moving from human networks to *computer networks* because the communication networks are a product of the human society in response to the natural need for communications. In our review of networks, we ended pointing out that every kind of communication seems to move naturally to wireless. In other words, in natural, human and computer networks wireless communications are more flexible and, as a consequence of this, more practical than other forms of communications.

In Chapter 1 we provided an introduction to WSNs. We pointed out that WSNs are a recent technology designed for unattended, remote monitoring and control, which have been successfully employed in several applications. WSNs are designed to perform environmental data sampling and processing, and to guarantee access of the processed data to remote users. Moreover, we pointed out that in traditional WSN models, these tasks consist in transmitting sensed data to a powerful node (the sink) which performs data analysis and storage.

In Chapter 2, we pointed out how most of the current WSNs research focus on uniform networks and more specifically on networks in which the sensors are all

equals or on networks that are distributed following the uniform distribution. Then, we showed that in current WSNs research, uniformity is a dogma, a believed true assumption that no one wants to offend. At this point, we described our point of view and we showed that uniformity does not exist. Starting from the consideration that uniformity does not exist, we could only start to study non-uniformity.

In Chapter 3, we provided the description of the state of the art of the data management in WSNs because all the possible uses of WSNs deal with the idea of data acquisition and data retrieval. These two concepts are strictly related because data retrieval is the response to a user query. The user should be able to actively program the network, via control programs, to retrieve data that is considered useful. In traditional WSN models these tasks consist in transmitting sensed data to a powerful node (the sink) which performs data analysis and storage. However these models resulted unsuitable to keep the pace with technological advances which granted to WSNs significant (although still limited) processing and storage capabilities. For this reason recent paradigms for WSNs introduced data base approaches to define the tasks of data sampling and processing, and the concept of data-centric storage for efficient data access.

In Chapter 4, we presented our contribution to the non-uniformity in the data management research field in WSNs. We focused on the the data-centric storage model and our main contribution in this area is Q-NiGHT. Q-NiGHT originated by the analysis of the experimental results that we performed on the Geographic Hash Tables (GHT) approach in non-uniform networks. These results point out the inability of GHT to provide good results in non-uniformly distributed WSNs. We revised Q-NiGHT following a top-down approach, defining at the beginning the whole system, leaving open problems and solving them later in the chapter. At the end of the chapter we provide also an application scenario in which Q-NiGHT is used as a building block of a more complex system that enables the networks to locate special sensor nodes that are able to provide services to other nodes.

Finally, in Chapter 5, we presented Stripes. Stripes is a family of protocols aimed at finding out the network density starting from a constant number of samples. Finding out network density is a central problem in non-uniformity aware protocols and systems. For this reason, the Stripes family of protocols is designed to be not intrusive i.e., to consume as less energy as possible. Stripes is designed to provide the network density on demand: when a node needs to know the density of the network, it queries the sample points (watch-points) to get the local densities and to reconstruct an approximation of the whole network density. Stripes uses caches extensively to keep the energy used to sample the network low.

C.2 Drawing a conclusion

We choose to develop our study in the non-uniformity influence into WSNs essentially for one reason: only few people worked on that topic and it seemed exciting

to explore something new. And it was exciting.

After an initial study of the general problem, in which we defined non-uniformity in WSNs (with the classification provided in Chapter 2), we choose to focus on one particular problem and to analyze that one.

Our choice was the study of non-uniformity in data management in WSNs. Data management offered a great opportunity to point out two things: (*i*) the uniformity assumption/dogma can produce solutions that are not suitable for non-uniformly distributed networks and (*ii*) taking non-uniformity into account, we can find *general* solutions to solve problems and these solution can be effective both in non-uniform and in uniform networks.

We started with the analysis of the ill behavior of GHT in non-uniform networks and we pointed out its design problems. Then, we provided a more general solution, from the non-uniformity point of view, using a fully generic non-uniform hash function that was inspired by the non-uniform random number generation using the rejection method (Q-NiGHT). This solution opened another problem, the problem of finding out WSNs' distributions. For this problem, we found an efficient solution that reconstructs the density starting from few (a fixed number of) samples and we found a very efficient way to move this knowledge around the network (the Stripes family of protocols).

The most important thing that we intended to point out, is that GHT (as well as many other protocols) is not able to deal with non-uniformity because the most basic design choices are not able to deal with non-uniformity. The original GHT never looks around to figure out what is the real distribution of the nodes, it simply *assumes* that the distribution is uniform. Apart from the QoS support, Q-NiGHT works exactly as GHT using a different hash function that is more flexible and that can be used also in uniform networks.

To complete Q-NiGHT and to have a distribution to pass as a parameter to our hash function, we needed Q-NiGHT to look around to figure out how the nodes were distributed in the network. We choose to use a simple reconstruction algorithm based on the sampling of the density from a fixed number of nodes in the network. The most crucial aspect of the algorithm is how we can get the samplings for the reconstruction. We wanted this operation to be as efficient as possible, still knowing that the uniformity assumption costs nothing and that it is not possible to be as cheap as nothing. To get rid of this problem, the Stripes family of protocols uses an aggressive caching technique.

As a closing remark on the non-uniformity problem in WSNs, we have to point out two things: (*i*) systems are non-uniformity proof by their design (*ii*) non-uniformity proof choices can have a reasonable cost.

For the first one, we need to develop a kind of non-uniformity consciousness into the design procedure of WSNs protocols and we need to take care of a couple of things in the design process. Every time we need use concepts such as *neighbors number* and *nodes density* in some protocol design step, we need to remember that

these concepts can be relative to the position of the nodes in the network and that nodes position and nodes density will be available only at run time.

For the second one, we need to take into account the problem that finding out the network density has a cost and that we have to keep it as low as possible. Stripes is a general solution to find out the network density and it works on the whole network. Stripes uses caches and fixed sampling points to find out the network density. Both these two ideas help the protocol to be as efficient as possible.

C.3 Looking to the future

The cave you fear to enter holds the treasure you seek.
- Joseph Campbell

If we think to the possible future developments of the ideas that are presented in this thesis, we have two things in mind*: data management and non-uniformity.

In the data management side of the future, we can imagine a lot of development in the study of non-uniformity proof solutions for the data management problem. These solutions range from different QoS techniques able to use the non-uniformity to improve the reliability of the system, up to new hashing techniques that take into account other factors, such as an estimation of the residual energy of the nodes to choose the best candidates to store data.

On the other hand, in the non-uniformity side of the future, we can start to work on other aspects of the non-uniformity influence. We can study the problem of routing and/or topology control in WSNs taking into account non-uniformity. From the point of view of the topology control, we started our investigation of non-uniformity in topology control in (Nidito and Pizziniaco, 2006). In this paper, we study the problem of the network dimensioning before the deployment of a WSN, taking into account the problem of deploying a connected network (at least in a particular region). More studies are on the way. Into an ongoing work, we are studying the properties of network connectivity in non-uniformly distributed networks.

Network connectivity problem is an interesting theoretical topic in sensor networks. In a network whose nodes are randomly deployed in according to a non-uniform distribution we cannot design the network topology a priori. Many works study the problem of the *critical neighbors number* in uniform networks (Santi, 2005, Xue and Kumar, 2004, Blough et al., 2003). The problem can be found in disguised forms too: in (Panchapakesan and Manjunath, 2001) and (Gupta and Kumar, 1998) the problem problem is known as the *critical transmitting range*. The problem can be summarized as follows: *Let r be the nodes transmitting range, how many neighbors does a node need to be connected with high probability?* If the network is uniformly distributed (or distributed with a Poisson process), finding this quantity

* Or better: we have two *main* things in mind.

enables us to state if the *whole* network is connected with high probability or not. In our work, we will move away from them and we will focus on two main topics: (*i*) we will provide a solution that fits the needs of non-uniformly distributed networks (the old works fit only uniform networks) providing a new mathematical model and solution to the connectivity problem and (*ii*) we will provide a Monte Carlo method to easily compute an approximated solution of the presented mathematical model. In our work we will show how to find an equation that describes *where*, in a wireless network, connectivity is achieved with high probability, given the total number of deployed nodes n, and their distribution function f.

Bibliography

(Akkaya and Younis, 2005) Akkaya, K. and Younis, M. (2005). A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325–349.

(Akyildiz et al., 2002a) Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002a). A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114.

(Akyildiz et al., 2002b) Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002b). A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114.

(Akyildiz et al., 2002c) Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002c). Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422.

(Al-Karaki, 2004) Al-Karaki, K. (2004). Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6):6–28.

(Albano et al., 2006a) Albano, M., Chessa, S., Nidito, F., and Pelagatti, S. (2006a). Geographic hash tables with QoS in non uniform sensor networks. In *ACM Mobihoc 2006 Poster Proceedings*, Firenze, Italy.

(Albano et al., 2006b) Albano, M., Chessa, S., Nidito, F., and Pelagatti, S. (2006b). Q-NiGHT: Adding QoS to data centric storage in non-uniform sensor networks. Technical Report TR-06-16, Dipartimento di Informatica, Università di Pisa.

(Albano et al., 2007) Albano, M., Chessa, S., Nidito, F., and Pelagatti, S. (2007). Q-NiGHT: Adding QoS to data centric storage in non-uniform sensor networks. In *Proceedings of The 8th International Conference on Mobile Data Management (MDM'07)*, Mannheim, Germany.

(Aly et al., 2006) Aly, M., Pruhs, K., and Chrysanthis, P. K. (2006). KDDCS: a load-balanced in-network data-centric storage scheme for sensor networks. In *Proc. of the 15th ACM international conference on Information and knowledge management*, pages 317–326, New York, NY, USA. ACM Press.

(Amato et al., 2005a) Amato, G., Baronti, P., and Chessa, S. (2005a). MaD-WiSe: Programming and accessing data in a wireless sensor networks. In *Proc. of IEEE Eurocon*, pages 300–303, Belgrado, Serbia and Montenegro.

(Amato et al., 2006a) Amato, G., Baronti, P., and Chessa, S. (2006a). MaD-WiSe: a distributed query processor for wireless sensor networks. Technical Report 2006-TR-39, Istituto di Scienza e Tecnologie dell'Informazione del CNR, Pisa, Italy.

(Amato et al., 2006b) Amato, G., Baronti, P., Chessa, S., and Masi, V. (2006b). The stream system: a data collection and communication abstraction for sensor networks. In *Proc. of IEEE International Conference on Systems, Man, and Cybernetics,*, Taipei, Taiwan.

(Amato et al., 2005b) Amato, G., Chessa, S., Conforti, F., Macerata, A., and Marchesi, C. (2005b). Health care monitoring of mobile patients. *Ercim news*, 60:6.

(Araujo et al., 2005) Araujo, F., Rodrigues, L., Kaiser, J., Liu, C., and Mitidieri, C. (2005). CHR: a Distributed Hash Table for Wireless Ad Hoc Networks. In *Proc. of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05).*

(Arnbak, 1987) Arnbak, J. (1987). Capacity of slotted aloha in rayleigh fading channels. *IEEE Journal On Selected Areas in Communications.*

(Bash et al., 2004) Bash, B., Byers, J., and Considine, J. (2004). Approximately uniform random sampling in sensor networks. Technical Report BUCS-TR-2004-031, Boston University Department of Computer Science.

(Bejar et al., 2001) Bejar, R., Krishnamachari, B., Gomes, C., and Selman, B. (2001). Distributed constraint satisfaction in a wireless sensor tracking system. In *Workshop on Distributed Constraints, IJCAI.*

(Bentley, 1975) Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517.

(Bettstetter, 2001) Bettstetter, C. (2001). Smooth is better than sharp: a random mobility model for simulation of wireless networks. In *MSWIM '01: Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 19–27, New York, NY, USA. ACM Press.

(Bettstetter, 2004a) Bettstetter, C. (2004a). The cluster density of a distributed clustering algorithm in ad hoc networks. In *Proceeding of the IEEE International Conference on Communications, ICC 2004*, volume 7, pages 4336–4340, Paris, France.

(Bettstetter, 2004b) Bettstetter, C. (2004b). On the connectivity of ad hoc networks. *The Computer Journal*, 47(4):432–447. Oxford University Press.

(Blough et al., 2003) Blough, D., Leoncini, M., Resta, G., and Santi, P. (2003). The k-neigh protocol for symmetric topology control in ad hoc networks. In *Proc. of MobiHoc '03*, Annapolis, MD, USA.

(Bollobas, 1985) Bollobas, B. (1985). *Random Graphs*. Academic Press.

(Bonnet et al., 2000) Bonnet, P., Gehrke, J., and Seshadri, P. (2000). Querying the physical world. *IEEE Personal Communications*, 7(5):10–15.

(Bonnet et al., 2001) Bonnet, P., Gehrke, J., and Seshadri, P. (2001). Towards sensor database systems. In *Proc. of 2nd International Conference on Mobile Data Management (MDM 2001)*, pages 3–14, Hong Kong, China.

(Bose et al., 2001) Bose, P., Morin, P., Stoimenovìc, I., and Urrutia, J. (2001). Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. *Wireless Networks,*, 7(6):609–616. Also in Proc. of *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DialM'99)*, Seattle, Washington, August 1999, 48–55.

(Buck, 1988) Buck, J. (1988). Synchronous rhythmic flashing of fireflies II. *The Quarterly Review of Biology*, 63(3):265–289.

(Bulusu et al., 2000) Bulusu, N., Heidemann, J., and Estrin, D. (2000). "gps-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34.

(Cao and Abdelzaher, 2004) Cao, Q. and Abdelzaher, T. (2004). Scalable logical coordinates framework for routing in wireless sensor networks. In *Proc. of 25th IEEE International Real-Time Systems Symposium (RTSS 2004)*, pages 349–358, Lisbon, Portugal.

(Caruso et al., 2005) Caruso, A., Chessa, S., De, S., and Urpi, A. (2005). GPS free coordinate assignment and routing in wireless sensor networks. In *Proc. of 24th Joint Conference of the IEEE Computer and Communications Societies (Infocom 2005)*, pages 150–160, Miami, FL, USA.

(Cerpa et al., 2001) Cerpa, A., Elson, J., Estrin, D., Girod, L., Hamilton, M., and Zhao, J. (2001). Habitat monitoring: Application driver for wireless communications technology. In *Proc. of 1st ACM SIGCOMM Workshop on Data Communications in Latin America and the Carribean*, pages 20–41, San Jose, Costa Rica.

(Cerpa and Estrin, 2002) Cerpa, A. and Estrin, D. (2002). Ascent: Adaptive self-configuring sensor networks topologies. In *Proceedings of Infocom 2002*, New York, NY.

(Chen et al., 2002) Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R. (2002). Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, 8(5).

(Chessa et al., 2007) Chessa, S., Nidito, F., and Pelagatti, S. (2007). *New Research on Wireless Communications*, chapter Distributed Data Management in Sensor Networks. Nova Publishers Inc. To be published in 2007.

(Cormen et al., 1990) Cormen, T. H., Leiserson, C. E., and Rivest, R. L. (1990). *Introduction to Algorithms*.

(Desnoyers et al., 2005) Desnoyers, P., Ganesan, D., and Shenoy, P. (2005). TSAR: A two tier sensor storage architecture using interval skip graphs. In *Proc. of of SenSys'05, San Diego, CA*.

(Dorogovtsev and Mendes, 2003) Dorogovtsev, S. and Mendes, J. (2003). *Evolution of Networks: From Biological Nets to the Internet and WWW*. Oxford University Press.

(Dousse et al., 2002) Dousse, O., Thiran, P., , and Hasler, M. (2002). Connectivity in ad-hoc and hybrid networks. In *Proc. of Infocom '02*, volume 2, New York, NY, USA.

(Fullmer and Garcia-Luna-Aceves, 1997) Fullmer, C. and Garcia-Luna-Aceves, J. (1997). Solutions to hidden terminal problems in wireless networks. In *Proceedings of the ACM SIGCOMM Conference : Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM-97)*, pages 39–50, Cannes, France.

(Gao et al., 2005) Gao, T., Greenspan, D., and Welsh, M. (2005). Improving patient monitoring and tracking in emergency response. In *Proc. of International Conference on Information Communication Technologies in Health*.

(Gupta and Kumar, 1998) Gupta, P. and Kumar, P. R. (1998). Critical power for asymptotic connectivity in wireless networks. In *Stochastic Analysis, Control, Optimization and Applications*. Birkhauser, Boston.

(Havinga et al., 2003) Havinga, P. J. M., Etalle, S., Karl, H., Petrioli, C., M. Zorzi, H. K., and Lentsch, T. (2003). Eyes–energy efficient sensor networks. In *Proceedings of PWC 2003*, pages 198–201, Venice, Italy.

(Intanagonwiwat et al., 2000) Intanagonwiwat, C., Govindan, R., and Estrin, D. (2000). Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proc. of the 6th International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 56–67, Boston, MA, USA.

(Jaromczyk and Toussaint, 1992) Jaromczyk, J. and Toussaint, G. (1992). Relative neighborhood graphs and their relatives. *Proceedings of IEEE*, 80(9):1502–1517.

(Ji and Zha, 2004) Ji, X. and Zha, H. (2004). "sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling. In *Proc. of 23th Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom 2004)*, pages 2652–2661, Hong Kong.

(Johnson et al., 2001) Johnson, D., Maltz, D., and Broch, J. (2001). DSR: The dynamic source routing protocol for multihop wireless ad hoc networks. In Perkins, C. E., editor, *Ad Hoc Networking*, chapter 5, pages 139–172. Addison-Wesley.

(Johnson and Maltz, 1996) Johnson, D. B. and Maltz, D. A. (1996). Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, pages 153–181.

(Kaplan, 1996) Kaplan, E. D., editor (1996). *Understanding GPS: Principles and Applications*. Artech House, Boston, MA.

(Karp and Kung, 2000) Karp, B. and Kung, H. T. (2000). GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proc. of the 6th International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 243–254, Boston, MA, USA.

(Kleinrock, 1996) Kleinrock, L. (1996). Nomadicity: Anytime, anywhere in a disconnected world. *Mobile Networks and Applications (MONET)*, 1(4):351–357.

(Krishnamachari et al., 2002) Krishnamachari, B., Bejar, R., and Wicker, S. (2002). Distributed problem solving and the boundaries of self-configuration in multi-hop wireless networks. In *HICSS '02: Proc. of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9*, page 297.2, Washington, DC, USA. IEEE Computer Society.

(Lässig et al., 2001) Lässig, M., Bastolla, U., Manrubia, S., and Valleriani, A. (2001). The shape of ecological networks. *Physical Review Letters*, 86(19):4418–4421.

(Lieberman, 1998) Lieberman, P. (1998). *Eve Spoke: Human Language and Human Evolution*. W. W. Norton & Company.

(Lin et al., 2003) Lin, M., Kumar, A., Qing, X., Beard, S. J., Russell, S. S., Walker, J. L., and Delay, T. K. (2003). Monitoring the integrity of filament wound

structures using built-in sensor networks. In *Proc. of SPIE – Smart Structures and Materials 2003: Industrial and Commercial Applications of Smart Structures Technologies*, volume 5054, pages 222–229, San Diego, CA, USA.

(Liu et al., 2003) Liu, J., Zhao, F., and Petrovic, D. (2003). Information-directed routing in ad hoc sensor networks. In *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 88–97, New York, NY, USA. ACM Press.

(Liu et al., 2005) Liu, J., Zhao, F., and Petrovic, D. (2005). Information-directed routing in ad hoc sensor networks. *IEEE journal on selected areas in communications*, 23(4):851–861.

(Madden et al., 2002a) Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. (2002a). TAG: a Tiny AGgregation service for ad-hoc sensor networks. In *Proc. of 5th Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA, USA.

(Madden et al., 2003) Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. (2003). The design of an acquisitional query processor for sensor networks. In *Proc. of the 2003 SIGMOD Conference*, pages 491–502, San Diego, CA, USA.

(Madden et al., 2002b) Madden, S., Szewczyk, R., Franklin, M. J., and Culler, D. (2002b). Supporting aggregate queries over ad-hoc wireless sensor networks. In *Proc. of 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002)*, pages 49–58, Callicoon, NY, USA,.

(Malan et al., 2004) Malan, D. J., Welsh, M., and Smith, M. D. (2004). A public key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *Proc. of 1st IEEE communications Society Conference on Sensor and Ad-Hoc Communications and Networks*, pages 71–80.

(Marcucci et al., 2005) Marcucci, A., Nati, M., Petrioli, C., and Vitaletti, A. (2005). Directed diffusion light: low overhead data dissemination in wireless sensor networks. In *Proc. of Vehicular Technology Conference, VTC 2005-Spring. 2005 IEEE 61st*, volume 4, pages 2538–2545.

(Milgram, 1967) Milgram, S. (1967). The small world problem. *Psychology Today*, pages 60–67.

(Nagpal et al., 2003) Nagpal, R., Shrobe, H., and Bachrach, J. (2003). Organizing a global coordinate system from local information on an ad hoc sensor network. In *Proc. of 2nd International Symposium on Information Processing in Sensor Networks (IPSN 2003)*, pages 333–348, Paolo Alto, CA, USA.

(Nasipuri and Li, 2002) Nasipuri, A. and Li, K. (2002). A directionality based location discovery scheme for wireless sensor networks. In *Proc. of 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA 2002)*, pages 105–111, Atlanta, GA, USA.

(Neumann, 1951) Neumann, J. V. (1951). Various techniques used in connection with random digits. In Taub, A. H., editor, *John von Neumann, Collected Works*, volume 5, pages 768–770. Pergamon Press, Oxford, Oxford.

(Newsome and Song, 2003) Newsome, J. and Song, D. (2003). GEM: Graph EMbedding for Routing and Data-Centric Storage in Sensor Networks Without Geographic Information. In *Proc. of the First International Conference on Embedded Networked Sensor Systems*, pages 76–88, Los Angeles, California, USA.

(Niculescu and Nath, 2003a) Niculescu, D. S. and Nath, B. (2003a). Ad hoc positioning system (APS) using AOA. In *Proc. of 22nd Joint Conference of the IEEE Computer and Communications Societies (Infocom 2003)*, pages 1734–1743, San Francisco, CA, USA.

(Niculescu and Nath, 2003b) Niculescu, D. S. and Nath, B. (2003b). Dv based positioning in ad hoc networks. *Telecommunication Systems*, 22(1–4):267–280.

(Nidito et al., 2007) Nidito, F., Battelli, M., and Basagni, S. (2007). Fault-tolerant and load-balanced localization of services in wireless sensor networks. In *Proceedings of The 66th IEEE Vehicular Technology Conference (VTC2007-Fall)*, Baltimore (MD), USA.

(Nidito and Pizziniaco, 2006) Nidito, F. and Pizziniaco, L. (2006). On the dimensioning of ad hoc sensor networks. In *5th Fifth Annual Mediterranean Ad Hoc Networking Workshop 2006*, Lipari, Italy.

(Okabe et al., 1992) Okabe, A., Boots, B., and Sugihara, K. (1992). *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams*. Wiley.

(Orecchia et al., 2004a) Orecchia, L., Panconesi, A., Petrioli, C., and Vitaletti, A. (2004a). Localized techniques for broadcasting in wireless sensor networks. In *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*, pages 41–51, New York, NY, USA. ACM Press.

(Orecchia et al., 2004b) Orecchia, L., Panconesi, A., Petrioli, C., and Vitaletti, A. (2004b). Localized techniques for broadcasting in wireless sensor networks. In *Proc. of DIALM-POMC '04*, New York, NY, USA. ACM Press.

(Panchapakesan and Manjunath, 2001) Panchapakesan, P. and Manjunath, D. (2001). On the transmission range in dense ad hoc radio networks. In *Proc. SPCOM '01*.

(Perkins and Royer, 1999) Perkins, C. E. and Royer, E. M. (1999). Ad-hoc on-demand distance vector routing. In *Proc. of the 2nd IEEE Workshop on Mobile Computer Systems and Applications*, New Orleans, LA, USA.

(Rabin, 1989) Rabin, M. O. (1989). Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36(2):335–348.

(Rahnema, 1993) Rahnema, M. (1993). Overview of the GSM system and protocol architecture. *Communications Magazine, IEEE*, 31(4):92–100.

(Rao et al., 2003) Rao, A., Ratnasamy, S., Papadimitriou, C., Shenker, S., and Stoica, I. (2003). Geographic routing without location information. In *Proc. of 9th International Conference on Mobile Computing and Networking (MobiCom 2003)*, pages 96–108, San Diego, CA, USA.

(Ratnasamy et al., 2003) Ratnasamy, S., Karp, B., Shenker, S., Estrin, D., Govindan, R., Yin, L., and Yu, F. (2003). Data-centric storage in sensornets with GHT, a geographic hash table. *Mobile Networks and Applications (MONET)*, 8(4):427–442.

(Rizzo, 1997) Rizzo, L. (1997). Effective erasure codes for reliable computer communication protocols. *ACM Computer Communication Review*, 27(2):24–36.

(Santi, 2005) Santi, P. (2005). *Topology Control in Wireless Ad Hoc and Sensor Networks*. John Wiley and Sons, Ltd., Chichester, West Sussex, England.

(Santi and Blough, 2002) Santi, P. and Blough, D. (2002). An evaluation of connectivity in mobile ad hoc networks. In *Proc. of DSN '02*, Washington, DC, USA.

(Santi et al., 2001) Santi, P., Blough, D., and Vainstein, F. (2001). A probabilistic analysis for the range assignment problem in ad hoc networks. In *Proc. of MobiHoc '01*, Long Beach, CA, USA.

(Savvides et al., 2001) Savvides, A., Han, C., and Strivastava, M. B. (2001). Dynamic fine-grained localization in adhoc networks of sensors. In *Proc. of 7th International Conference on Mobile Computing and Networking (MobiCom 2001)*, pages 166–179, Rome, Italy.

(Seada and Helmy, 2006) Seada, K. and Helmy, A. (2006). Efficient and robust geocasting protocols for sensor networks. *Computer Communications*, 29(2):151–161.

(Seybold, 2005) Seybold, J. (2005). *Introduction to RF Propagation*. Wiley-Interscience.

(Shah et al., 2003) Shah, R., Roy, S., Jain, S., and Brunette, W. (2003). Data MULEs: Modeling a three-tier architecture for sparse sensor networks. In *Proceedings of IEEE SNPA Workshop*.

(Shang and Ruml, 2004) Shang, Y. and Ruml, W. (2004). Improved MDS-based localization. In *Proc. of 23th Joint Conference of the IEEE Computer and Communications Societies (Infocom 2004)*, pages 2640–2651, Hong Kong.

(Srivastava et al., 2001) Srivastava, M., Muntz, R., and Potkonjak, M. (2001). Smart kindergarten: Sensor-based wireless networks for smart developmental problem-solving environments. In *Proc. of 7th International Conference on Mobile Computing and Networking (MobiCom 2001)*, pages 132–138, Rome, Italy.

(Steere et al., 2000) Steere, D. C., Baptista, A., McNamee, D., Pu, C., and Walpole, J. (2000). Research challenges in environmental observation and forecasting systems. In *Proc. of 6th International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 292–299, Boston, MA, USA.

(Szewczyk et al., 2004) Szewczyk, R., Mainwaring, A., Polastre, J., Anderson, J., and Culler, D. (2004). An analysis of a large scale habitat monitoring application. In *Proc. of 2nd International Conference on Embedded Networked Sensor Systems (SenSys 2004)*, pages 214–226, Baltimore, MD, USA.

(Tilak et al., 2003) Tilak, S., Murphy, A., and Heinzelman, W. (2003). Non-uniform information dissemination for sensor networks. *Proceedings of the 11th International Conference on Network Protocols (ICNP03), Atlanta, GA, USA*.

(Wang et al., 2003) Wang, H., Elson, J., Girod, L., Estrin, D., and Yao, K. (2003). Target classification and localization in habitat monitoring. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2003)*, pages 844–847, Hong Kong.

(Ware et al., 2001) Ware, C., Chicharo, J., and Wysocki, T. (2001). Modelling of capture behaviour in ieee 802.11 radio modems. In *Proceedings of IEEE Vehicular Technology Conference*, Atlantic City, NJ.

(Watts, 1999) Watts, D. (1999). *Small worlds: the dynamics of networks between order and randomness*. Princeton.

(Watts, 2003) Watts, D. (2003). *Six degrees: The science of a connected age*. WW Norton.

(Xu et al., 2001) Xu, Y., Heidemann, J., and Estrin, D. (2001). Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 70–84, Rome, Italy. ACM.

(Xue and Kumar, 2004) Xue, F. and Kumar, P. R. (2004). The number of neighbors needed for connectivity of wireless networks. *Wireless Networks*, 10(2).

(Yao and Gehrke, 2002) Yao, Y. and Gehrke, J. (2002). The Cougar Approach to In-Network Query Processing in Sensor Networks. *SIGMOD Record*, 31(3):9–18.

(Ye et al., 2002) Ye, W., Heidemann, J., and Estrin, D. (2002). An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of Infocom 2002*, New York. IEEE.

(Ye et al., 2004) Ye, W., Heidemann, J., and Estrin, D. (2004). Medium access control with coordinated, adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506.

(Zhao and Guibas, 2004) Zhao, F. and Guibas, L. (2004). *Wireless Sensor Networks An Information Processing Approach*. Morgan Kaufman Publisher, S. Francisco.

(Zhou et al., 2005) Zhou, F., Chen, G., and Xu, Y. (2005). Construction of small worlds in the physical topology of wireless networks. *CoRR*, abs/cs/0503051.