Université des Sciences et Technologies de Lille – Lille 1

Département de formation doctorale en informatique **École doctorale SPI Lille**
UFR IEEA

Algorithms for Realistic Wireless Sensor Networks

THÈSE

présentée et soutenue publiquement le 15. December 2011.

pour l'obtention du

Doctorat de l'Université des Sciences et Technologies de Lille

(spécialité informatique)

par

Jovan Radak

Composition du jury

Président :	—
Rapporteurs :	Martin Heusse, Prof. Ensimag Grenoble Pedro Ruiz, Prof. Universidad de Murcia, Spain
Examinateurs :	Fabrice Théoleyre, CR CNRS, LSIIT Claude Chaudet, MdC Telecom Paris Tech Isabelle Simplot-Ryl, Prof, Univ. Lille 1, DCR Inria Rocquencourt
Directeur de thèse :	Nathalie Mitton, INRIA Lille - Nord Europe

Laboratoire d'Informatique Fondamentale de Lille — UMR USTL/CNRS 8022
INRIA Lille - Nord Europe

Numéro d'ordre: 40693

Abstract

Wireless sensor networks can be defined as networks of small spatially distributed devices, called sensor nodes, which are working cooperatively - exchanging messages wirelessly - on the same application. Today these kinds of networks are widely used in environmental monitoring, industrial and consumer applications and for military purposes.

In this thesis we are tackling different areas of research in wireless sensor networks: topology control, mobility, neighborhood discovery and large scale experimentation. We are using relative neighborhood graph reduction along with power supply data obtained from the sensor node to develop topology control algorithm. This algorithm maintains connectivity of the network in critical situations when some of the sensors drain their batteries. Neighborhood discovery parameters are used to deduce relative mobility of the sensor nodes. Then these parameters are adapted with transmission range to obtain energy efficient neighborhood discovery algorithm. Large scale experimentation sites are valuable tool for developing and testing of algorithms for wireless sensor networks but they also have various deficiencies, the biggest of them is cost. We present emulation of large scale networks as a solution. It uses small networks with the specific placement of the sensor nodes which allows replicating thus emulating behavior of the large scale networks.

Algorithms are tested and evaluated on the WSNet simulator and practically using the SensLab platform and WSN430 sensor nodes.

Resume

Les réseaux de capteurs sont des réseaux composés de petits objets répartis dans l'espace, appelés nœuds ou capteurs, qui travaillent en collaboration - échange de messages radio - sur la même application. Aujourd'hui, ces types de réseau sont largement utilisés dans le suivi environnemental, industriel et les applications grand public ainsi qu'à des fins militaires.

Dans ces travaux, nous nous attaquons à différentes primitives dans les réseaux de capteurs: le contrôle de topologie, la mobilité, la découverte de voisinage et l'expérimentation à grande échelle. Nous utilisons la réduction de graphe des plus proches voisins avec les données d'alimentation du nœud afin de développer notre algorithme de contrôle de topologie. Cet algorithme conserve une connexité du réseau dans les situations critiques où certains capteurs épuisent leur batterie. Les paramètres de découverte de voisinage sont utilisés pour déduire la mobilité relative des capteurs. Ensuite, ces paramètres sont adaptés, ainsi que la puissance d'émission, pour obtenir un algorithme efficace de découverte de voisinage. Les plates-formes d'expérimentation à grande échelle sont des outils utiles pour développer et tester des algorithmes pour ces réseaux de capteurs sans fil, mais ils ont aussi des déficiences diverses, le plus grand d'entre eux étant le coût. Nous présentons l'émulation de réseaux à grande échelle. Cela simule de petits réseaux avec un placement précis des capteurs qui permet la réplication de comportement pour ainsi émuler des réseaux à grande échelle.

Les algorithmes sont testés et évalués théoriquement et sur le simulateur WSNet et en pratique sur la plate-forme SensLab utilisant des capteurs WSN430.

Acknowledgments

First of all, I thank my Ph.D advisors Nathalie Mitton and Isabelle Simplot-Ryl for introducing me to wireless sensors networks, for being patient with me during the first year of my work and helping me find the right direction in my research work. Isabelle was there for me when I needed guidance and to find the direction of my research. Nathalie helped me express my ideas, encouraged my self-initiative and kept me going to the aim.

I would like to thank *rapporteurs* professors Martin Heusse and Pedro Ruiz for their valuable suggestions which have led to improved version of this dissertation.

Thanks to the jury members professors Fabrice Théoleyre and Claude Chaudet for their opinion on this dissertation and for interesting questions which gave me ideas how to further improve this work.

I would like to thank professor David Simplot-Ryl for the nice welcoming to the POPS research group and valuable discussions which gave me ideas how to proceed with my work.

Thanks to my colleagues Samuel Hym and Arnaud Fontaine for their advices and help finding the best software tools for resolving my problems.

I would like to thank to the former and current POPS research group members, namely Roudy Dagher, Tony Ducrocq, Milan Erdelj, Grégory Guche, Enrico Natalizio, Tahiry Razafindralambo, Loïc Schmidt, Julien Vandaële and others, for the disscusions, ideas, advices – both on the work and in private – and the great time spent in the Summer Schools. Also, I would like to thank Kate Price for helping me with my English language and my everlasting problem with articles.

Staying in France and dealing with (sometimes) complicated procedures would not be that easy if there were not Anne Rejl and Anne-Claire Binetruy, their help was invaluable.

I am hugely grateful to my family. My mother and father have been a great support and always stood by decisions that I made in my life. My wife Tamara and son Djordje whose love, patience and support were giving me strength to keep on the right track.

Contents

List of Figures

List of Tables

List of Tables 1

Essentially, all models are wrong,
but some are useful.

George E. P. Box

Chapter 1
Introduction

Wireless sensor networks can be defined as networks of small spatially distributed devices, called sensor nodes, which are working cooperatively - exchanging messages wirelessly - on the same application. Today these kinds of networks are widely used in environmental monitoring, industrial and consumer applications and for military purposes.

To say that wireless sensor networks, and in general wireless communication, have been under great scientific interest during the past few decades, is a trigger that immediately reminds us of numerous introductions from many scientific publications (even we have used it). Some of these publications were good, giving interesting results and milestones for future works, others were not so good. This introduction may be a stereotype but it does not have a great impact on the results that are presented herein. At this point we should point out that perhaps it is not nice or catchy to start scientific publications with this phrase but, at the same time it is the truth. It is definitely true that wireless communications are ubiquitous and that now we can hardly imagine the world without them. Being a necessity in the modern world, wireless communications give us a good starting point to explore and improve it even more. But the question is how do we do it?

1.1 Problem

Can we take the problem directly from the real world and try to solve it with all its little differences and nuances? Throughout the history (of science) we have witnessed two possibilities: either the solution was strongly profiled and customized for the small subset of instances (or even only one instance) which arise from the problem, or the bigger subset of problems was given a similar approach using a simple model leaving out more subtle differences and trying to fit in a wide range of problems in one single model (for example the approach one size fits all). Reasons for this kind of approach can be found in the complexity of systems which are found in the real world. Thus, to give access to the wide range of existing mathematical tools one needs to find good proportion between the entity found in the real world and the model which should replace it in calculations. This model can be very simple and even

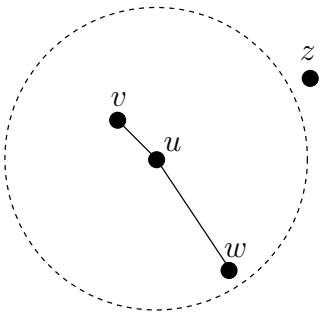


Figure 1.1: Unit disk graph model

inaccurate but it can still give good results when applied to more complex systems.

A good example for the aforementioned statement can be taken from the usage of unit disk graph, a model widely used in the area of wireless communications. A simple correlation can be made between real networks and unit disk graph. The first step would be to simplify our notion of network by saying that a network is represented with graph $G = (V, E)$ in which our devices are nodes of graph V and that their communication links are presented with the set of edges of graph V. Now, we have already made the first step, our network is modeled as a graph, one step further would be to add a simple property to our graph, saying that a link between two nodes of graph u and v exists if and only if the distance between them is smaller than the given unit distance R:

$$E = \{(u, v) \in V^2 \mid d(u, v) \leq R\}$$

where $d(u, v)$ is the Euclidean distance between nodes u and v. As simple as that, this model states that if we assume that node u (Fig. 1.1) wants to communicate with other nodes it can do it only if they are within the disk whose center is in node u. So, nodes v and w can communicate with u because they are within the unit disk while node z, even though its very close to this area, cannot communicate with u. What can we say about this model? For sure it's not the best approximation of the situations that we can find in the real world but still, many important results which apply to devices which run, in this same real world, are obtained using this simple model. And here, we come to the important question: if the model that we used for our calculations gives good results when applied to the real devices does this make it realistic? When it comes to this question it is hard to give a straight answer, it also depends on the other premises that we impose on our problem.

Then, if it is so hard to define what do **we** mean by the word realistic and why do we use this term in the title of this work? For us, the word realistic means something in connection with devices on which we are planning to use in

our solution. Not quite strong a definition but we can bring it under a different light and say that realistic also means that an important part of the problem is taken as it is and no further simplifications are applied to this certain aspect of our problem. For example, when we speak about battery level (in Chapter 3) we do use battery level from the real device and this information is used directly in calculations and in the application of the algorithm. In this way we can say that we are trying more to make a customized solution to the problems that we are addressing than to find one solution that fits a wide variety of cases. Nevertheless, the solutions that are given can be easily applied to any device which meets assumptions that we made.

It seems that we insist on the realistic approach and that we want a solution which is strongly connected to the real world, either through the usage of real devices or through the mechanisms (techniques) that they are using in the real world. If we insist so much on real world applications then logical question would be: did we manage, in our approach, to avoid usage of some general models used in this area? Well, certainly we did not avoid usage of some simple models but that does not makes our solution, or the approach, any less realistic. It just means that there are some general spots, in the problems that we are addressing, that can be easily, efficiently and at the same time accurately solved by using well known models and solutions (energy model used in Chapter 4).

1.2 Contributions

In the *sea of possibilities* where did we find our place? Looking at the names of the chapters can give the idea that we are addressing different problems and different areas of wireless sensor networks. It may be true, but our vision was always a practical problem and if not directly applied to hardware, then how it would fit in, and possible problems and enhancements to the current solution. We are trying to fit our way of thinking into three big areas of research in wireless sensor networks.

Topology control It is one of the classical problems in the area of wireless sensor networks, how to make our network better, energy efficient, simple and yet connected, through the control of the links between the sensor devices. We base our work on the sensor devices WSN430 [WSN430], we find possible ways to improve devices response to the critical problem of the loss of power supply. Using the information obtained from the device, and one of the graph reduction algorithms we create a simple topology control algorithm that follows the power supply of the sensor device and reacts in the proper

way (reconfiguring the network) in the case when some of the devices reach a critical power supply level.

Mobility Another well known property of wireless sensor networks. Inherent to some networks, imposed on other networks or used by some networks, the mobility of devices raises lots of questions and endangers the functioning of some algorithms and mechanisms created for static networks. We try to address the problem of neighborhood discovery in mobile networks. Being a very important mechanism, neighborhood discovery (Chapter 2.3.2) needs to retain its proper functioning in mobile networks also. Furthermore, since this protocol includes periodical sending of the messages it also needs to be efficient both in finding all its neighbors, keeping accurate track of it in the neighborhood table, and efficient in energy meaning that it might lower the power of transmission and omit reaching the nodes which are soon going to be out of range. We give our solution to this problem by addressing some well known models and results and we prove its efficiency and accuracy.

Scalability in experiments Experimental work with wireless sensor network was for many years the weakest spot. It requires large number of devices, yet these devices are very limited in their power supply as well as in their facilities. Meaning that these sensor devices need to be recharged and to be loaded with proper programs on them. This might not be the problem with a small number of devices, up to 20 - 30, but to keep experiments more realistic we usually need more devices which also gives us the problem of the cost beside the problem of proper functioning. During the last few years there have been many attempts to build various testbeds which would allow researchers to test their algorithms in real environments. Even though they are great help for development, these testbeds are often constrained in the placement of sensors, the type of sensors used and sometimes in the gathering and the interpretation of obtained data. We present emulation, a process which uses small wireless sensor networks to emulate (estimate in real hardware) the behavior of the large scale wireless sensor network. This small network is used with some tools (graph generation tool) to estimate different kinds of routing algorithms on real devices and in large scale networks.

1.3 Structure

This manuscript is divided into 6 chapters. The introductory chapter covers some basic notions and giving the preview and introduction to the following chapters. Chapter 2 presents preliminaries for the models that we used and

gives a brief review of the current advances in real devices and experimentation possibilities with them and a brief review of theoretical advances in the areas that we are covering. Next, Chapter 3 describes our approach to topology control and gives the explanation of our algorithm and its functioning on the real sensor hardware. Results show that our proposition performs very well and allows sensors to dynamically prefer high battery level nodes. Chapter 4 presents our approach to the accurate and energy efficient neighborhood discovery in mobile wireless sensor networks. Results show that we still maintain a good neighborhood table accuracy while minimizing the energy spent. Scalability of the experiments, and our idea for the small testbed which can be used to obtain the results for large scale networks are given in Chapter 5. Finally, we give our conclusion in the last chapter, and we also give starting points for possible advances and future works.

Chapter 2

State of the art

Contents

2.1 Introduction

This work involves several different areas of research in wireless sensor networks connected to a similar idea incarnated to the word *realistic* in the title of this thesis. Usage of the realistic wireless sensor networks assumes the usage of the real sensor hardware to obtain the results, to validate the solution already proven in theory or the results collected by the simulation. Thus, we first present sensor nodes, their general characteristics and tools provided for the development. Large scale testbeds are one of the tools that we are using to retrieve results in our work but also to try to present alternatives (emulation) to these large testbeds, one which is less costly and still producing accurate results. We present large testbeds in general emphasizing three testbed sites in Europe.

The second part of this chapter involves more details about the background of our work and places our work in an appropriate context. First, we present our general notations and define the notions which will be used in other sections. In this preliminary part, we explain detailed notions of neighbors and the neighborhood discovery and a simple model for calculating energy cost when transmitting packets from the sensor nodes. Then we move to the core of this chapter, which is divided into three smaller sections. We have the same motivation when investigating realistic wireless sensor networks but we chose

to make the distinction between smaller areas that we are confronting in this work. First we give more details about topology control, and one type of it which is based on the graph reduction algorithms. A few different types of graph reduction algorithms are presented emphasizing the one that we are using in our work: *relative neighborhood graph*. Neighborhood discovery in mobile wireless sensor networks is presented in the next section. We highlight the problem of accurate and energy efficient neighborhood discovery and present intuition behind the solution that we are proposing. The last section deals with the problem of scalable experimentation in wireless sensor networks. A few different approaches to emulation are presented finishing with our vision of the emulation.

2.2 Real sensors used in wireless sensor networks

2.2.1 Types of sensors and their capabilities

The research area of wireless sensor networks, along with ad-hoc networks, began to develop much before the actual hardware and the devices for it became widely accessible. Nowadays, there are lots of different types of sensor nodes, but there are some parts which are characteristic to all of them. First of all it has to contain a processing unit, in almost all cases this is a microcontroller, it is usually one of the low power solutions of the leading manufacturers (Texas Instruments MSP430 series or Atmel ATmega). Rarely can we see devices as it is case with IMote 2.0 (Table 2.1) that have ARM processor or some more powerful processing unit. Radio chip is the part which enables these devices to communicate wirelessly thus making them wireless sensor nodes, more often we can find radio chip CC2420 which operates at 2.4GHz and is compliant with IEEE 802.15.4 standard and less often CC1100 series of radio chip which operates at lower frequencies, 315, 433, 868 and 915 MHz. The third important point involves sensors, being a wireless sensor node means that it is supposed to have sensor, usually it is some kind of sensor that follows environment data like temperature, humidity, noise, etc. The sensor part of the node is usually extensible via daughterboard cards which contain additional sensors or communication interfaces. All these parts are usually supplied with Lithium Ion batteries included in the package of a sensor. Figure 2.1(a) shows sensor node WSN430 which is based on the TI MSP430 microcontroller, and comes in two varieties with CC2420 or CC1100 radio chip, onboard it contains temperature and light sensors (photodiodes). It is extensible with additional daughterboard cards which may contain a digital compass, accelerometer,

Sensor node	Microcontroller	Radio Chip
IMote 2.0	ARM11 XScale	CC2420
MicaZ	ATmega128	CC2420
TelosB	TI MSP430	CC2420
WSN430	TI MSP430	CC2420 and CC1100

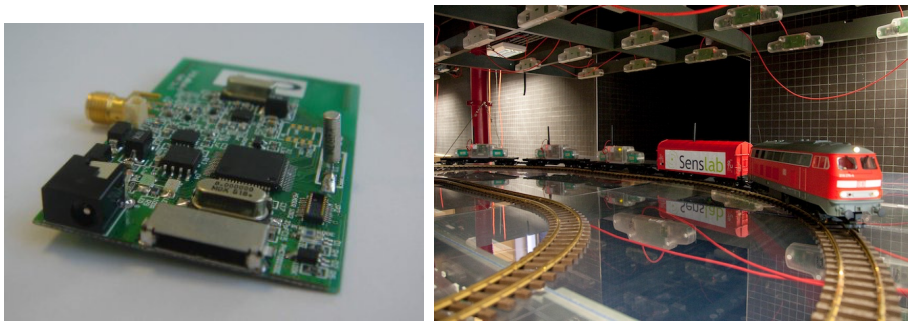
Table 2.1: Sensor node hardware

GPS device...

2.2.2 Program support for WSN

Different platforms exist upon which we can develop and implement programs for wireless sensors. We are going to briefly present three operating systems for wireless sensor devices.

TinyOS Presents a real-time operating system dedicated to wireless sensors. It is presented in 2000 and supports a wide range of existing sensor nodes, but there are also unofficial ports for the devices not officially supported. Applications for TinyOS are written in NesC (Network Embedded System C). It uses event-based programming of the devices meaning that the system waits for events to happen and then responds according to the handling of the event. There are two types of entities that are accessible in TinyOS: events and tasks. Scheduling of the tasks is done preemptively. Communication between the layers of the program are possible via interfaces.



(a) WSN430 (b) SensLAB platform

Figure 2.1: WSN430 and SensLAB platform. INRIA / Photo N. Fagot

Contiki It is another open source multitask operating system that also supports event-based programming. Apart from the operating system core this operating system also contains processes which can become services or applications. It supports multithreading and preemptive scheduling. Contiki programs are written in C programming language. Contiki also provides IP stack both for IPv4 and IPv6 with complete support for 6lowPAN.

FreeRTOS This is also real-time operating system [FreeRTOS] for the embedded devices. It is a highly configurable operating system made to be small and simple. The kernel of this operating system consists of only four C files. Scheduler can be configured in both preemptive and cooperative modes. Being small and simple makes it a good choice for beginners but nevertheless it can be used to write serious applications.

Apart from the presented operating systems there exists also a full protocol stack for embedded wireless devices (6lowpan), and the systems which can handle database on a memory constrained device [Tsiftes 2011]. These solutions are out of the scope of this thesis so we will not give any further details about them.

2.2.3 Large scale experimentation

Large testbeds are also one of the possibilities to conduct the experiment and to observe behavior of the designed algorithm in a real environment. Being very interesting subject for the future development, wireless sensor networks are gaining lots of resources for the experiments. The idea is to have large sites with lots of sensor nodes, possibly heterogeneous, and to observe how different algorithms and services perform in the presence of other wireless sensors and with possible interferences from the real environment but also from other wireless devices which may be using the same part of spectrum for their functioning. In here we will mention only three large experimentation sites deployed in Europe just to have an idea of the structure of these kinds of testbeds and their advantages and shortcomings.

Wisebed

Wisebed is a joint project of 9 different academic institutions all over Europe. Large scale experimentation sites consist of 750 sensor nodes. This experimentation site is heterogeneous having different types of sensors like Isense, TelosB, Tnode. It is organized in federation architecture providing multilevel structure of interconnected testbeds. Large numbers of sensors on these experimentation sites are static but they also contain 40 mobile sensors.

Different kinds of measurement are supported including the power measurement on some sensor nodes. It also supports virtualization of the topology and possibility of co-simulation with part of the testbed.

SmartSantander

SmartSantander is a city-scale facility for research and experimentation of Internet of things technologies and services. It contains 4 large experimentation sites: *(i)* Santander (Spain) with sensor nodes placed across the city, mainly outdoor deployment *(ii)* Guilford (UK) with sensors placed in a university campus, mix of outdoor and indoor deployment *(iii)* Lubeck (Germany) mix of university campus and city deployment *(iv)* Belgrade (Serbia) mostly city deployment also containing mobile sensors placed on buses facilitating vehicular network experiments. Up till now 2000 sensor nodes have been deployed with a plan to deploy a total number of 20000 by august 2013. This experimentation testbed is mostly developed for the experimentations with Internet of things services but it can also be used for experiments with protocols for wireless sensor networks.

Senslab

SensLAB is a large scale experimentation testbed 2.1(b) deployed in 4 cities in France (Grenoble, Lille, Rennes and Strasbourg). It contains total number of 1024 sensor nodes based on the SensLAB node which consists of 2 WSN430 [WSN430] nodes, with 2 possibilities for the radio chip CC2420 or CC1100, and the gateway board connecting them. This kind of implementation of sensor node, though not too small, allows users to access the information in the sensor node which is already running the program that it was developed for. Users can follow energy consumption and also to communicate with sensors. Sensor nodes are deployed both outdoor and indoor and the testbeds also contain the mobile nodes. Containing only one type of sensors makes this testbed specially suitable for the experiments with protocols and applications for wireless sensor networks. Besides the testbed itself, SensLAB also offers a wide range of tools developed for it, WSnet and WSim – simulators and predeveloped ports for different kinds of real time operating systems (FreeRTOS, TinyOS, Contiki)

2.3 Notations and preliminaries

In this work we are addressing the problems of topology control, neighborhood discovery in mobile networks and scalable experimentation - we are using the

term emulation – testbed built upon small wireless sensor networks which can be used to emulate large scale networks. In this section we will give the preliminaries and definitions of some basic models that are being used in our work and which are common for all chapters. More specific details are given in the following chapters explaining in detail the main points that we are addressing and our contribution.

2.3.1 General remarks

Graph G is defined as $G = (V, E)$, a pair of set of nodes V and set of edges E. For a node u we say that it is in graph G when $u \in V$. In text we usually address to the nodes of the graph G using u, v and w, if not otherwise emphasized these nodes are considered to be the part of the graph G ($u, v, w \in V$).

When talking about transmission of the single package we refer to the *source* node as the sender of the package and *destination* node is the receiver of the package. Other preliminary definitions are given in following 2 separate sections.

2.3.2 Neighborhood discovery

Neighborhood discovery is the process in which each node finds its neighboring nodes. In order to define neighborhood, we first have to define notion of neighbors. For two nodes u and v in graph G we say that they are neighbors if they can receive the message from each other. More formal, this is given as:

$$E = \{(u, v) \in V^2 \mid v \text{ receives messages from } u\}$$

In similar way, 1-hop neighborhood of the node u, $N(u)$ is defined as:

$$N(u) = \{v \in V \mid (u, v) \in E\}$$

usually this is applied vice versa, meaning that the link is symmetrical and that if node u can communicate with node v then also node v can communicate with node u. Certain algorithms require knowledge of 2-hop neighbors of the nodes, this means that each node needs to know neighbors of its neighbors. 2-hop neighborhood $N_2(u)$ of node u is defined as:

$$N_2(u) = \{v \in V \mid N(N(u)) \setminus N(u) \cup \{u\}\}$$

In our work (Chapter 4) we are defining the notion of *bilateral* and *unilateral* neighbors in order to have more precise view of the nodes that can be

ID	source address	neighborhood table (ID, distance)

Table 2.2: Format of HELLO message

discovered by the single node. Bilateral neighbors present the pair of neighbors u and v which can communicate between themselves – meaning u can receive messages from v and vice versa, formally given:

$$E_{bil} = \{(u, v) \in V^2 \mid uv \in E \wedge vu \in E\}$$

For node u we say that it is unilateral neighbor of v if it can receive message from v and at the same time v cannot receive message from u, more formally:

$$E_{uni} = \{(u, v) \in V^2 \mid uv \in E \wedge vu \notin E\}$$

The process of the neighborhood discovery consists of sending of HELLO messages – simple messages which allow the node which has received them to find source of the message and some additional information. In the example (Table 2.2) HELLO message contains also data about the neighbors of the node which is sending this HELLO message. These data can be saved by the node which receives message and then further on used (for routing or topology control).

The second part of the neighborhood discovery consists in gathering data from all received HELLO messages – this is being done by all nodes independently – and forming the neighborhood table. HELLO message can be short and containing as little as ID of the message, so that the node which receives message knows that its a HELLO message, and address' of the source of HELLO message, which can be some unique identifier (in practical solutions it is usually MAC address). In some more complex algorithms, which would require more data, HELLO message can contain more data. Table 2.3 shows neighborhood table of N entries in which each entry contains data of the source node, distance to it and state of the battery of the source node, this organization is somewhat similar to the neighborhood table that we are using in our work in Chapter 3.

Knowing the process of neighborhood discovery, now we can easily define process for retrieving 2-hop neighborhood. Knowledge of 2-hop neighborhood is practically obtained by putting the neighborhood table, or just list of 1-hop neighbors, in the HELLO message. In this way each time when one of the nodes receive HELLO message it can also know which are the neighboring nodes of the node which sent HELLO message and create the table for 2-hop neighborhood.

$source_1$	$distance_1$	$battery_1$
$source_2$	$distance_2$	$battery_2$
$\vdots$	$\vdots$	$\vdots$
$source_N$	$distance_N$	$battery_N$

Table 2.3: Neighborhood table

2.3.3 Energy consumption

Energy consumption of the single node is modeled as:

$$E = r^{\alpha} + C \tag{2.1}$$

where E represents energy consumption of the single node when it is used for sending single message, with the transmission range r and C which is the constant used to model global losses, additional computation before the sending of the message, handling the message on the MAC layer, idle state, etc. In the literature [Fleury 2009] as the most common values are mentioned $\alpha = 4$ and $C = 10^8$.

2.4 Topology control

> Topology control can be defined informally as the art of coordinating nodes' decisions regarding their transmitting ranges, in order to generate a network with the desired properties while reducing node energy consumption and/or increasing network capacity.
>
> Paolo Santi [Santi 2005].

This definition, although being informal, explains main aims of this mechanism in wireless sensor networks. Considering that these kinds of networks consist of big number of constrained devices which are communicating between each other it is obvious that there is a need for some kind of organization which will enhance desired properties (connectivity, throughput) and at the same time diminish undesired behavior (energy consumption, collisions, interferences). Following the definition we can say that the main aim of the topology control is to virtually simplify the network (by removing some unwanted parts) and in this way optimize communication in the network. Simplification of the network also helps the nodes to use less memory requiring from them to know only the subset of obtained information thus saving the memory for other critical operations.

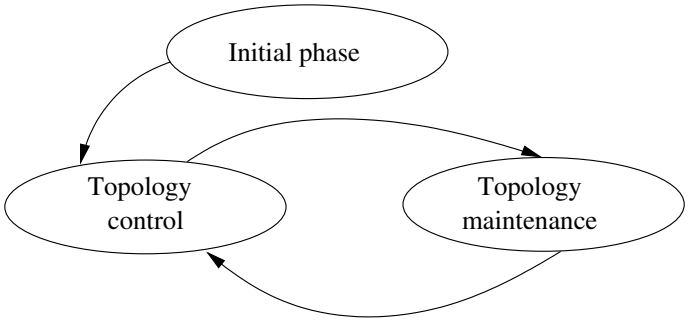


Figure 2.2: Process of topology control

In his book [Santi 2005] author places topology control in the wireless sensor network protocol stack, between MAC layer and routing layer, with the routing layer as upper and MAC layer as lower layer to topology control. Placed in the protocol stack in this way, topology control is facilitating more efficient routing with updates of the list of neighboring nodes which are available for routing and at the same time it can set appropriate power level for transmission communicating with the MAC layer. Recently, there have been more debates on the placement of the topology control in the protocol stack, some authors agree with the placement previously explained while the others are placing topology control as the lower network layer part which should be called according to the needs of the upper layers [Fleury 2009].

In [Wightman 2008] authors make the clear distinction between the topology construction and topology maintenance as the parts of topology control. Further on [Labrador 2009] they give more detailed description dividing topology control in three subprocesses: *(i)* initial phase – in which the nodes are discovering their neighborhood and build initial topology; *(ii)* topology construction – second phase in which the nodes are building new topology according to some algorithm, and when the new algorithm is established they move to *(iii)* topology maintenance – the phase in which the new algorithm monitors the status of the reduced network and in if needed triggers the topology construction (Figure 2.2).

There are many considerations which should be taken into account when designing topology control algorithm, it should have some properties specific to the wireless sensor networks: **localized** – allowing it to run independently of other nodes and using only information from its immediate neighborhood thus saving the memory and lowering energy requirements; **simple** – allowing it to be run multiple times without impact (or with small impact) on the performance of the network, having in mind that wireless sensor network mostly consist of constrained devices this also means that we are not wasting processing resources; **energy-efficient** – it is usually run on constrained devices

with limited power supply (battery) so it needs to help the nodes to reduce overall energy consumption; **preservation of connectivity** – it is critical for the reduced network that it remains connected in order to preserve end to end communication in the network. Out of many possible techniques for the topology control we have chosen graph reduction based techniques which can be easily implemented, can be constructed localized, preserves connectivity and augments energy efficiency.

2.4.1 Topology control based on graph reduction

This kind of topology control algorithm requires knowledge of neighborhood for each node thus the first step in this kind of algorithms is neighborhood discovery. Each node finds his neighbors exchanging the HELLO messages and by creating the neighborhood table as explained in Section 2.3.2. For these kinds of algorithms, to run, either 1-hop neighborhood and nodes positions or 2-hop neighborhood is needed, in case when we cannot accurately know the position information.

Alternatively, there is a possibility of using some of the link quality parameters which can be either calculated or extracted from the physical layer of the protocol stack which can be used as a notion of distance between two sensors. Examples for these link quality parameters are expected transmission count (ETX) and received signal strength indicator (RSSI). ETX of the link l is calculated as [Javaid 2009]:

$$ETX_l = \frac{1}{p_{lsf} \cdot p_{lsr}} = \frac{1}{reliability(l)} \tag{2.2}$$

where p_{lsf} and p_{lsr} are the probabilities that the packet is going to be successfully delivered in forward (f in the p_{lsf}) and in reverse direction (r in p_{lsr}). Probabilities can be easily calculated empirically counting the number of messages that are successfully received per time unit. RSSI is the link quality factor which can be directly extracted from the hardware that sensor uses for communication. It is usually part of the message that is being received. RSSI gives valuable information of quality of communication [Srinivasan 2006] but it cannot be used directly as a notion of distance to get accurate information of the node location, it should be rather used with appropriate statistical method to obtain accurate parameters for localization [Figuera 2009].

Graph $G = (V, E)$ with its nodes V, equipped with neighborhood tables, and links between them E is a starting point to perform one of the algorithms based on a graph reduction. In general, these algorithms aim to remove communication links which do not meet the imposed constraints, either they have low quality or they are redundant by some criteria. Applying this kind of

algorithm on the initial graph G gives us reduced graph $G_r = (V, E_r)$ which has the same set of nodes V as the starting graph but contains only subset of the edges (communication links) $E_r \subseteq E$. In this class of algorithms we are differentiating two types of algorithms: *(i)* centralized – which need to collect the global data of whole network in order to run it properly; *(ii)* localized – algorithm which can be implemented and run independently on each node using only its neighborhood table.

Minimum spanning tree

Minimum spanning tree (MST) – is the subgraph of the starting graph, which is a tree with a property that its weight is less than or equal to the weight of every other spanning tree. In [Prim 1957] is presented one of the oldest algorithms that addresses this problem. The main problem with this algorithm is that for this kind of structure there is no localized algorithm. However, using the structure called localized minimum spanning tree we can use localized algorithm [Li 2003]. Each node is computing its own localized MST over the informations gathered locally and exchange its results with its neighbors. A link is kept in reduced graph iff it belongs to both trees. This algorithm has nice properties of preserving connectivity of starting graph and bounding the node degree to 6 but it contains unidirectional links which can be eliminated without affecting the graph connectivity.

Gabriel graph

Gabriel graph [Gabriel 1969] of the graph $G = (V, E)$ is reduced graph $GG(G) = (V, E_{GG})$ in which two nodes u and v are neighbors if and only if there is no node in circle with diameter uv more formally:

$$E_{GG} = \{(u, v) \in E \; if \; \nexists w \in E \mid w \in D(u, v)\}$$

This graph has a nice property: if the G is unit disk graph then $GG(G)$ is planar subgraph of G. This property of the Gabriel graph is used to construct connected planar subgraph as the first phase in face and greedy-face-greedy (GFG) routing algorithms [Bose 1999]. Though this is good property it limits us only to geometric graphs.

Relative neighborhood graph (RNG)

Relative neighborhood graph is the first time mentioned in the work of Toussaint [Toussaint 1980] connecting it to the MST and Delaunay triangulation, and showing that the RNG over the set of nodes is superset for MST and subset of the graph obtained by Delaunay triangulation. In his work [Supowit 1983]

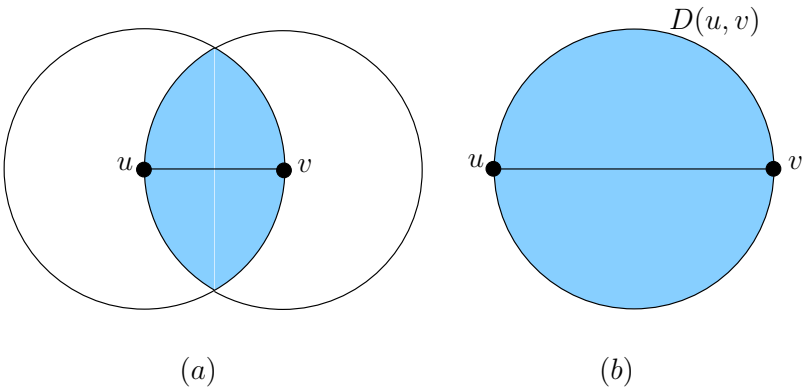


Figure 2.3: Example of graph reductions – (a) Relative neighborhood graph RNG (b) Gabriel graph

Supowit presented the algorithm which can compute RNG in $O(n \log n)$ time. $RNG(G) = (V, E_{RNG})$ is subgraph of the graph with the property:

$$E_{RNG} = \{(u, v) \in E \mid \forall w \ \mathbf{max}(d(u, w), d(v, w)) > d(u, v)\}$$

Defined in this way, RNG, MST, LMST and GG of the unit disk graph $G = (V, E)$, which uses Euclidean metric as a weight function, have a following property: $MST(G) \subseteq LMST(G) \subseteq RNG(G) \subseteq GG(G)$ [Fleury 2009]. One of the nice properties of RNG is that it can be computed locally. Each node u equipped either with data of the position its neighbors or with knowledge of its 2-hop neighborhood can compute its set of RNG links $N_{RNG}(u)$ locally:

$$N_{RNG}(u) = \{v, w \in N(u) \wedge v \in N(w) \mid d(u, v) < d(v, w) \wedge d(u, w) < d(v, w)\}$$

In this way node u can calculate set of its RNG neighbors $N_{RNG}(u)$ visiting every triangle which it is part of and eliminating the links with to the nodes which does not satisfy given condition. Less formally defined, in every triangle which contains node u the link with the highest weight is eliminated.

Another nice and useful property of RNG is the possibility to choose the metric that we want to use *i.e.* weight function can be defined using the value or the property which is of our interest, for example it can be the strength of signal, quality of link, probability of receiving the message, the only constraint is that weight function needs to be symmetric: $d(u, v) = d(v, u)$.

Because of its good properties RNG can be found in many different applications. XTC algorithm [Wattenhofer 2004], X topology control - where X means exotic, exceptional, extraterrestrial, is also using RNG algorithm (as proven in [Santi 2005]). Another example of modification of RNG algorithm is found in [Khadar 2009] where the authors use RNG algorithm with received

signal strength indicator RSSI as a weight function. Since RSSI can be different on in each of the nodes they are choosing one value (smaller) to serve as the weight function. The algorithm is slightly modified in the sense that the authors before the application of RNG reduction apply primary filter removing all the links which have weight function (RSSI) smaller than given filter value. In this way they are relaxing computation of RNG for each node but with the risk of loosing the connectivity in case of using strict filter – if the filter value is too high the graph will be disconnected before the application of RNG.

2.4.2 Energy concerns in topology control

Energy efficiency is one of the major concerns in topology control, and in wireless sensor networks in general, taking into account limited power supply [Ephremides 2002]. There are many papers claiming energy efficiency in the algorithms that they are presenting. However our goal is not energy efficiency itself but to focus on the problem of the limited power supply (battery) and when we reach the critical state, near the end of functioning of the battery, to react properly. To the best of our knowledge there are not many scientific papers that treat this problem in the same way.

In [L.Sichitiu 2005] authors are investigating the behavior of wireless sensor networks when using batteries with different levels and how can they improve overall performance of the network assigning different battery levels to different nodes. Similar problem is examined in [Long 2009] where the authors are presenting algorithm for battery allocation for optimization of the lifetime of the network with given cost constraints. Result is assignment of the different set of battery packs (with different capacities) to the different sets of nodes with significantly extended network lifetime.

Our idea differs from the ideas in these papers in the sense that we are trying to work with the sensor nodes the way they are, without trying reallocate batteries or to assign them different levels. Using the same idea as [Ravindranath 2010], in which authors are proposing usage of the sensors, like accelerometer, magnetic compasses, and gyroscopes, on the wireless devices - smart phones and tablets - to retrieve so called *sensor hints* which help them to improve network protocols. Our idea is to detect possibilities of the sensor nodes which can then help us formulate our algorithm and apply appropriate. Using the battery level which can be detected on the sensor nodes, that we are using, we are able to define RNG based topology control algorithm.

2.5 Neighborhood discovery and network mobility

Mobility is one of the important properties of wireless sensor networks. Sensor nodes in wireless sensor networks can have the mobility as one of the inherent properties, as it is case with sensor nodes attached to the animals with the purpose to track animals' habits and their natural habitat [WASP]. In this case mobility pattern is very often hard to estimate and protocols built upon this problem must take into account possible losses of connectivity or delays in the transmission of messages. Other possibility would include mobile agents (robots) which may have given mobile pattern to follow or even used (controlled) to improve some of the parameters in the WSN [Loscrì 2010]. In this case overall energy efficiency of transmission can be significantly augmented using controlled mobility and smart placement of nodes related to the routing path.

Due to the specific nature of mobile networks some of the characteristic mechanisms used in static wireless sensor networks need to be redefined and adapted to the specific types of the mobility of the nodes. We are specially interested in the functioning of the neighborhood discovery under the assumption that nodes are mobile.

Neighborhood discovery or *HELLO* protocol as explained in [Moy 1994] functions in similar way as given in Section 2.3.2. Neighborhood discovery, as described in [Moy 1994], assumes fixed rate for the frequency of the HELLO messages. While this can be true for the static networks in case of the mobile networks this assumption is not adequate since it is more natural to assume that the nodes which are moving faster are also changing their neighbors faster thus they need to update their neighborhood table more often and since they are gaining and loosing their neighbors more often it is logical to assume that they need to send HELLO messages at a higher rate. In this case when we speak about mobility, we think about relative mobility *i.e.* mobility of the sensor node referring to the other sensor nodes. For example if we have a fleet of the nodes which are moving together at some fixed speed then they are in relative sense static because each node always "sees" the same nodes in its neighborhood.

In [Troel 2004] assuming that there is a knowledge of relative speed between nodes V and threshold distance in communication area aR such that $a < 1$ is given the optimal value for the frequency of HELLO messages:

$$f_{opt} = \frac{2V}{aR}$$

Connection between mobility and neighborhood discovery is made through

the adaptation of the rate of sending of HELLO messages [Li 2011, Ingelrest 2007]. In [Ingelrest 2007] authors are proposing turnover based adaptive HELLO protocol (TAP). The main problem is to find appropriate frequency for HELLO messages in the mobile network. Starting from the same intuition explained in previous paragraph, the authors define *turnover* which presents difference in the neighborhood table in two successive transmits of HELLO messages. Using result of the existence of optimal frequency of HELLO messages [Troel 2004] they numerically calculate optimal turnover. This optimal value for the turnover is then used to adapt the current frequency of HELLO messages. In this way each node responds to its mobility, visioned through the change of neighborhood table (turnover), with appropriate change of frequency of HELLO messages. Using the same premises authors are defining auto regressive hello protocol (ARH) [Li 2011] in which each node predicts its own position using auto regression based mobility model. Following the network dynamics ARH is evolving and adapts itself to the optimal state using information gathered locally. ARH is compared to the TAP and authors show that it has the same performance in neighborhood discovery at a reduced rate of HELLO messages but with the cost of additional awareness of the location.

For a good performance of TAP algorithm accuracy of neighborhood table is important factor meaning that each node needs to remove entries in neighborhood table which are not valid. Neighborhood lifetime algorithm (NLA) [Ahmad Kassem 2010] performs alongside with TAP and it adapts refreshing rate of neighborhood table entries thus improving the accuracy of neighborhood table and the performance of TAP algorithm.

In [McGlynn 2001] authors are considering static network and aim at minimization of overall energy consumption. In their analysis they are using birthday paradox (probability that two persons with the same birthday are in the same place) to determine specific time slots in which they are going to put node in one of the three different states: transmit, listen or idle with chosen probability. After this step probabilistic round-robin is applied maximizing the number of discovered links.

In [Cohen 2011] the distinction is made between initial and continuous neighborhood discovery. Initial neighborhood discovery is applied when the sensor is unaware of its immediate environment, while the continuous neighborhood discovery is performed when the node is already aware of the neighborhood. Initial neighborhood discovery needs to be done by each sensor separately while continuous discovery can be applied as joint task of the segment of the neighborhood, and not each sensor. This joint task allows single node to spend more time in sleep mode and relies to the part of its neighborhood which is currently in active state. In this way they can lower energy consumption with high probability that new node is going to be properly dis-

covered. The algorithm guarantees that new node is going to be discovered in given time slot with requested probability,

2.6 Experiments with scalable networks

Large scale testbeds, previously mentioned (Section 2.6), are very good tool for practical implementation of the protocols for the wireless sensor network. However, these testbeds can not be used instantly and the user usually depends on the technology used to build this kind of testbed (sensor nodes) and topology of the testbed, which is usually fixed not allowing to many parameters to change. Another approach is specific type of the experiment called emulation which tries to combine the best of the simulation and experimentation with the real hardware.

Emulation of large scale networks has been studied in a variety of contexts [Canonico 2007], [Maier 2007], [Grau 2009]. In the area of emulation of large scale wireless sensor networks few different types of approaches are proposed. Main aim in all approaches is to overcome deficiencies of simple simulation either with environment emulation - in which the characteristics of the real nodes are built-in and executed in simulator - or using network emulation - in which each node communicates with real node in order to obtain more accurate results.

Object-oriented representation of sensors, communication channels and physical media (mobility model, power model, etc) has been used in J-Sim [Sobeih 2006]. In this approach, application specific models are developed in object-oriented fashion using subclasses within simulation framework. Usage of FPGA based DSP engine with almost any wireless device (RF node), is presented in [Judd 2005]. It provides good results on the higher level performance in real networks but it lacks precision on the hardware (lower) level. It is proven that using this kind of emulator good results can be obtained in development and evaluation of wireless protocols.

In their work [Ke 2000] authors are using combination of the simulation and the real sensors. They have added three modules (real time scheduler, network objects and tap agents) to ns-2 simulator to achieve better cohesion with the hardware and to better simulate hardware on the ns-2 simulator. Two real machines are attached to the endpoints of the network producing real network traffic and running the algorithms. The traffic from the real machines are then transfered to the ns-2 server using added modules to the nodes which are being emulated in ns-2. Using this system good results are obtained for the emulation of 10 - 120 mobile nodes with the problem that it cannot follow the experiments in the real time.

When using word emulation we have in mind completely different concept. The idea that we propose is the usage of small wireless sensor network, of up to 50 sensor nodes, which is used iteratively thus emulating behavior of the real sensor network. Other big difference in our setup is that we choose the source node and its neighborhood while the destination is virtual and routing algorithm, for example, is executed in iterative steps each time recalculating new position of the destination (more about it in Chapter 5).

CHAPTER 3

Topology control using sensor hardware

Contents

The work presented in this chapter shares the same idea as the work presented in [Ravindranath 2011, Ravindranath 2010] but has the novelty to apply it to topology control in wireless sensor networks. In these papers the authors present the usage of the sensors (gyroscopes, magnetic sensors, accelerometers) on the wireless devices, smart phones and tablet devices, in order to get some additional data, called external sensor hints, which can then further help them to improve wireless network protocols that run on these devices. In our work we have fixed our platform to WSN430 sensor nodes and, having the same idea, used sensor specific data to improve topology control algorithm for wireless sensor networks.

3.1 Introduction

As defined in Section 2.4, topology control is a mechanism that allows logically removing 'bad' links to favor some other ones regarding some parameters. We chose to use hardware characteristics to provide a topology control in which nodes with low battery level do not hold an important role in the network connectivity.

Hardware which used for building wireless sensor networks (WSN) usually consists of few different sensors, depending on the usage of WSN – as an example this can be temperature or humidity sensors which can detect different types of units connected with environmental parameters, microphones – for

the detection of level of noise, different types of proximity sensors or some types of seismic sensors in special of wireless sensor networks for structural health monitoring or when gathering characteristics of the ground. Second important part is processing unit, which is usually microcontroller with good performance/energy ratio such as Texas Instruments MSP430 series or Atmel's Crossbow series. Third part is communication unit which allows sensors to communicate between themselves, most usual radio circuits used are CC1100 series which are low power radio transceivers working under the frequency of 1GHz and CC2420 which are 2.4GHz radio transceivers compliant with IEEE 802.15.4 (ZigBee ready).

These kinds of systems are designed with the intent to be deployed on places which are hardly accessible, so the main idea is to deploy sensors once and then to use them for a long period of time, and to leave them to reconfigure themselves and gather the data. Such devices, in the absence of the uninterruptible power supply, are usually powered with batteries. This makes them even more challenging to work with, because all the algorithms designed for this kind of hardware also must take into account energy as one of the primary goals in optimizations of designed algorithms in order to allow longer functioning of these devices and networks built on them.

Usage of batteries can input also another types of errors which cannot be noticed in wired networks with uninterruptible power supply. Peaks in the usage of microcontrollers or sensor units may cause malfunctioning of the whole unit which can then cause different types of unwanted behaviors such as sudden restarts of the units or problems with the access to specific part of the units (radio communication circuitry or some of the sensors). Our experiments with WSN430 sensor nodes, which are the building part of the SensLAB platform [Senslab] and which are extensively used for practical experiments with wireless sensor networks, have shown that discharge of the battery supply of these units can cause two types of errors. The first one, less problematic, is in case when battery is close to the discharge and sensor stops communication with the others still allowing the usage of processing unit (microcontroller) until battery completely depletes. The second type of error is more malignant, on certain levels of battery it was noticed that sensor starts to jam the rest of the network using radio communication unit more extensively and sending data packages even when it is not supposed to thus causing jams in the network.

These issues led us to the direction in which we wanted to **create topology of the network which will be less prone to this kind of errors**. It means not just specific topology of the sensors but also certain level of the smartness which each sensor needed to have in order to apply certain type of topology control which will then in conjunction with the smartness of sensors

eliminate or at least diminish this kind of problem. The problem was two layered *(i)* make the sensors smarter and allow them to know the state of their power supply and *(ii)* use these smarter sensors with specially designed topology control to have effects of battery depletion minimized *i.e.* to remove from the network the sensors which can cause malfunctioning and at the same time to maintain connectivity in the network or to localize the part of the network which needs to be reconfigured due to the loss of connectivity.

3.2 Making the sensors aware of their power supply

In our work we use WSN430 platform [WSN430] which allows us to use hardware specific parts in order to make the sensors smarter.

3.2.1 Using analog-digital converter input of microcontroller

Circuitry of the WSN430 is conceded in such a way that positive connector of the Lithium Ion (Li Ion) battery used as power supply is connected through the diode to the ADC2 input of the MSP430 microcontroller used (as shown on Figure 3.1). This kind of architecture allows us to easily use analog input of the microcontroller and digitalizing it get the value of the current state of the power supply.

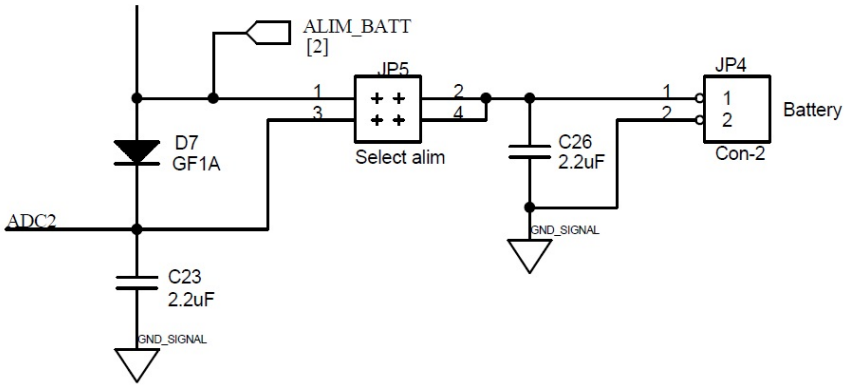
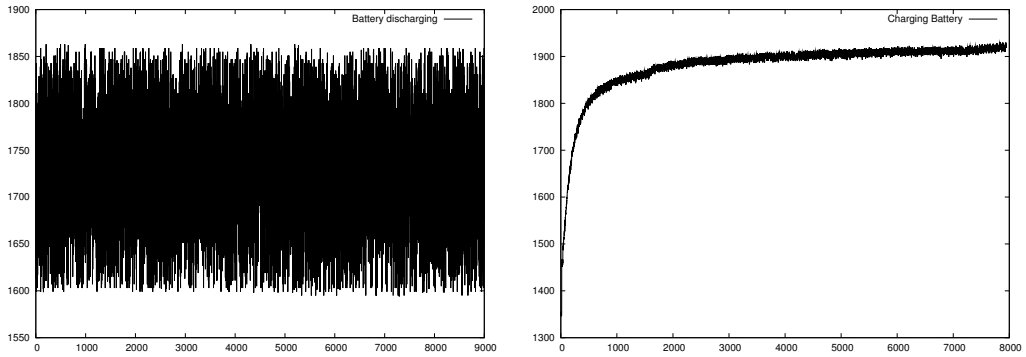


Figure 3.1: Part of the schematics of the WSN430 sensor node, showing connection between battery input and A/D converter input of the microcontroller

As an idea this approach was promising but the real issue was to get precise and accurate results. This was due to the usage of the battery –

the same one which we are trying to evaluate – to get reference voltages for A/D conversion. In this way we were getting highly inaccurate and imprecise results which we could not use further to build sensor smartness based on this kind of values. Figure 3.2 shows fluctuations of the voltages obtained from the A/D converter pin of the MSP430 microcontroller. On the x-axis is the number of the samples which were taken periodically each 500 ms, and on the y-axis is the value captured which presents battery voltage diminished for the drop of the voltage on the diode D7 (Figure 3.2). From these two Figures (3.2(a) and 3.2(b)) we can see the difference in the reading of ADC2 pin (Figure 3.1) in two cases, with no external power supply attached when battery is discharging (Figure 3.2(a)) and with power supply attached when battery is charging (Figure 3.2(b)).



(a) Discharging of the battery (b) Charging of the battery

Figure 3.2: Samples captured at the input 2 of A/D converter of the microcontroller

3.2.2 Supply voltage supervisor

Microcontrollers in the MSP430x series are equipped with circuitry called *Supply Voltage Supervisor* – SVS. This part of microcontroller follows voltage level of the power supply of the microcontroller and gives user information when power supply drops under fixed voltage level. SVS can detect 16 discreet voltage levels ranging from 3.7V to 1.9V (See Table 3.1).

In this way we can set certain voltage level to be tracked in our program. When battery drops bellow this voltage SVS will raise the bit in its register and we will have the information that battery has dropped bellow the voltage level that we have set up. There few ways in which SVS can react, it can trigger power-on reset or just set pin SVS to one. The second solution is more appropriate to the problem that we are trying to address, so we just set up

VLDx bits	Voltage level
0000	SVS is off
0001	1.9V
0010	2.1V
0011	2.2V
0100	2.3V
0101	2.4V
0110	2.5V
0111	2.65V
1000	2.8V
1001	2.9V
1010	3.05V
1011	3.2V
1100	3.35V
1101	3.5V
1110	3.7V
1111	Voltage compared to 1.2V

Table 3.1: Values of the SVS register and voltage levels

simple function which periodically, with period of few seconds, track SVS pin and in case that it is raised sends this information further on to be used in topology control algorithm.

3.3 Topology control Algorithm based on battery level

The main idea is that nodes that are running out of energy should be given less importance and appear as leaves in the resulting graph. In this way, when their battery is exhausted, the network will not be disconnected and we avoid costly graph re-computing.

Nevertheless, the battery level can not be used directly by the algorithm since battery level is related to a node and RNG graph uses metrics related to links. Value assigned to link uv between nodes u and v should be the same from the point of view of u and from the point of view of v to avoid links to be removed improperly and network disconnections. Therefore, we first assign a value to edges based on node battery levels.

3.3.1 Models and preliminaries

The network is modeled as a graph $G = (V, E)$ where V is the set of sensors and E is the set of edges. $uv \in E$ if and only if there exists a radio link between sensors u and v, *i.e.* they are in communication range of each other. We denote by $N(u)$ the set of physical neighbors of node u, *i.e.* the set of nodes v such that $uv \in E$. Let $\delta(u) = |N(u)|$ be the cardinality of $N(u)$, also called the degree of node u. We also define $N_{RNG}(u) \subset N(u)$ the set of RNG neighbors of node u. Every node is aware of its battery level, denoted as $BL(u)$ for node u. We consider that every node u has a unique identifier. We denote the identifier of node u as $ID(u)$.

We assume that a node u is aware of every edge within its neighborhood, *i.e.* node u knows every edge vw such that $v \neq w$ and such that $v \in \{u \cup N(u)\} \wedge \{w \in N(u)\}$. This can be achieved by two ways: either nodes are aware of their positions and of the one of their neighbors (nodes broadcast their position in HELLO messages) or nodes are aware of their 2-neighborhood (nodes broadcast their neighbor list in HELLO messages). If nodes are aware of their positions, they can easily compute the edge length. If they are not, we assume that node u can estimate the distance between itself and its neighbor v by the using some of the link quality estimators such as RSSI. RSSI is not always inversely proportional to the distance – which should be its main advantage – but it gives an indication on the link quality. Thus if the link is short and it has a low RSSI, that means that the quality is poor and equivalent to a very distant node.

3.3.2 Making Connection from Voltage Level to RNG Weight Function

In order to make a connection between battery level of two sensor nodes on each link, we introduce a value called *power factor*. Power factor of link uv, noted $PF(u, v)$ is determined using the voltage level of the two nodes on the link and is such that $PF \in \{0, 1, 2\}$. We consider a battery level threshold τ. If the battery level of a sensor node is lower than this threshold, the node is considered as a critical node. Power Factor value is assigned to every edge by using Algorithm 1 that takes as an input value battery level threshold τ and battery level for all nodes. According to these values, Algorithm 1 divides sensor nodes into two sets: *(i)* normal battery state nodes, with battery level higher than τ, and *(ii)* critical battery state nodes, with battery level lower than τ. Then depending on the battery state of each sensor node on the link, the power factor is assigned to each link as follows:

- power factor 0 – if both nodes are in normal battery state,

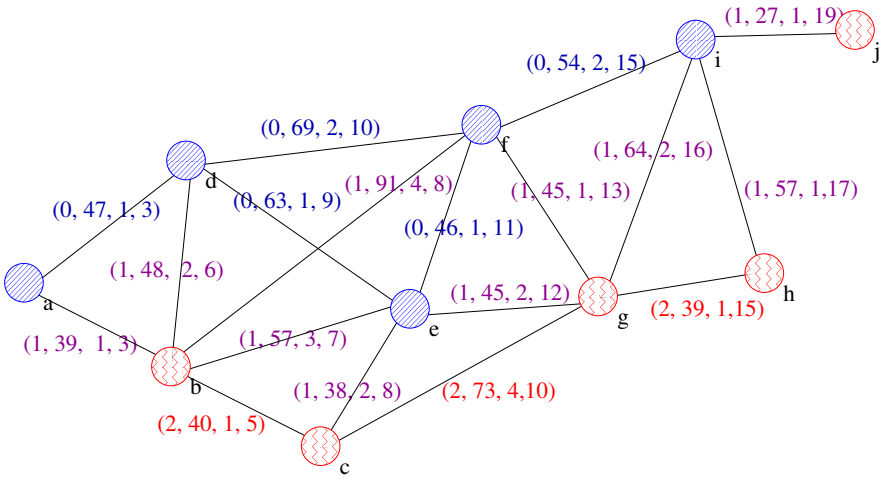


Figure 3.3: Example for PF computation. Blue nodes have a high battery level. Red nodes have a low battery level.

- power factor 1 – if exactly one of the nodes is in critical battery state,
- power factor 2 – if both nodes are in critical battery state.

Algorithm 1 is run on node u. It assumes that node u is aware of the battery level of each of its neighbors and knows whether there exist a link between the two of them. This information can be achieved through the use of Hello messages in which every node piggybacks its battery level and its position if available otherwise its neighborhood table. Based on this, node u is able to identify any triangle within its neighborhood and then to assign power factor value on every of these links.

To illustrate the Power Factor computation, let us consider Figure 3.3. Nodes that appear in blue are nodes which battery level is higher than τ while red nodes are the ones which battery level is lower than τ. On this figure, link ad gets a Power Factor equal to 0 since both nodes a and b have a high battery level. Link gh gets a Power Factor of 2 since it connects two nodes with a low battery level. Link be connects two nodes with different battery levels and thus gets the Power Factor value 1.

3.3.3 Algorithm for topology control using power factor

Once Power Factor is computed on every link, RNG computation can be performed. Nevertheless, since Power Factor only returns 3 different values, in a triangle, several edges could hold the same Power Factor value. Thus we need to use a second metric that allows nodes to choose between the

Algorithm 1 Calculate **Power Factor**

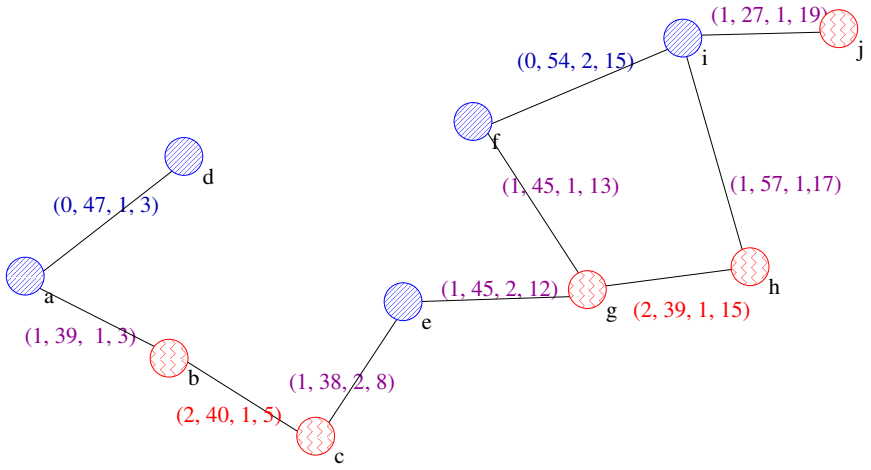
1: **for all** $v \in N(u) \mid u \neq v$ **do**
2: **if** $(BL(u) > \tau) \wedge (BL(v) > \tau)$ **then**
3: $PF(u, v) \leftarrow 0$
4: **else if** $(BL(u) > \tau) \wedge (BL(v) \leqslant \tau) \vee (BL(u) \leqslant \tau) \wedge (BL(v) > \tau)$ **then**
5: $PF(u, v) \leftarrow 1$
6: **else**
7: $PF(u, v) \leftarrow 2$
8: **for all** $w \in N(u) \cap N(v) \mid u \neq v \neq w$ **do**
9: **if** $(BL(w) > \tau) \wedge (BL(v) > \tau)$ **then**
10: $PF(w, v) \leftarrow 0$
11: **else if** $(BL(w) > \tau) \wedge (BL(v) \leqslant \tau) \vee (BL(w) \leqslant \tau) \wedge (BL(v) > \tau)$ **then**
12: $PF(w, v) \leftarrow 1$
13: **else**
14: $PF(w, v) \leftarrow 2$

nodes in such a way that every node of the triangle take the same decision regarding edge removal to avoid network disconnections. To do so, we apply the traditional RNG metrics, *i.e.* Euclidean distance between nodes and then, to break any potential additional ties, the node identifiers. To ensure that every node computes the same value for a link, we first compute the difference between the identifiers of the two end nodes. Nevertheless, if node with ID 2 is connected to nodes with ID 1 and 3, the two differences of ID remains the same. So, in such a case, we consider afterwards the addition of IDs. Using both difference and addition of the nodes IDs, we provide that every node computes the same values for a link and that two links in a triangle can not have the same values since nodes IDs are unique and we can not have at the same result for both addition and difference for two different links in a same triangle. Figure 3.3 shows the different values considered for every link. Every link holds a set of values $(PF, d, ID-, ID+)$ where PF is the Power Factor, d is the Euclidean length of edge, $ID-$ is the difference between ID of end nodes and $ID+$ is the addition of ID of end nodes. As ID we have considered the rank of the letter in the alphabet. For instance, $ID(a) = 1$, $ID(b) = 2$, etc....

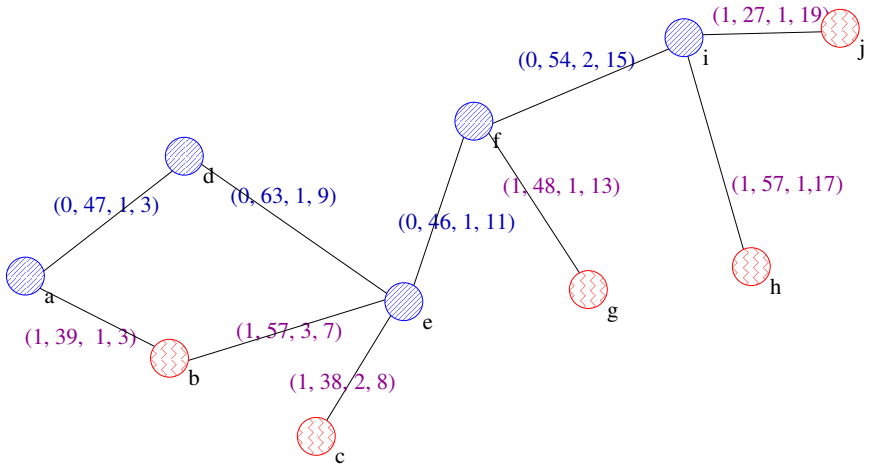
We define $\prec$ as a binary total order such that $uv \prec uw$ if and only if

- $PF(uv) < PF(uw)$ or
- $PF(uv) = PF(uw) \wedge d(u, v) < d(u, w)$ or
- $PF(uv) = PF(uw) \wedge d(u, v) = d(u, w) \wedge |ID(u) - ID(v)| < |ID(u) - ID(w)|$.

- $PF(uv) = PF(uw) \wedge d(u,v) = d(u,w) \wedge |ID(u) - ID(v)| = |ID(u) - ID(w)| \wedge |ID(u) + ID(v)| < |ID(u) + ID(w)|$.



(a) Distance based RNG



(b) Battery level based RNG

Figure 3.4: Example for RNG computation. Blue nodes have a high battery level. Red nodes have a low battery level.

Yet, the edge removal runs as described by Algorithm 2. We assume that node u is aware of the length of every edge within its neighborhood, either because nodes are aware of their positions and broadcasts it in Hello messages to their neighbors, or because they are able to estimate it based on RSSI for instance. Every node considers every triangle within its neighborhood (like triangles fig, igh or efg on Fig. 3.4(b)) and determines what edge to logically remove in the RNG. To do so, it first compares the PF values of nodes. If

one of edges has a lower PF value than other ones, it is removed (*e.g.* link gh is removed in triangle igh on Figure 3.4(b)).

If two edges have the same lower PF value or than every edge has the same PF value, the longer one is removed. For instance, in triangle fig on Figure 3.4(b), the longer edge is fi and should be removed in a traditional RNG algorithm. But f has a PF value of 0 and thus it is kept. Instead, we remove edge ig which has the same PF value than edge fg but is longer.

If edges hold the same PF value and are of the same length, ties are broken by considering the difference between identifiers of two nodes on the end of each edge. The edge with the largest ID difference is removed. This is for instance the case in triangle efg on Figure 3.4(b) where edges eg and fg hold the same PF value and have the same length. Their ID is used to differentiate them. Since $|(ID(f) - ID(g)| < |(ID(e) - ID(g)|$, link eg is removed.

Algorithm 2 Calculate RNG

```
1: N_RNG(u) ← N(u)
2: CalculatePowerFactor(u)
3: {Node u computes PF factor of every link within its neighborhood.}
4: for all v, w ∈ N(u) do
5:    if uv ≺ vw ≺ uw or vw ≺ uv ≺ uw then
6:       N_RNG ← N_RNG(u) \ {w} {Link uw is removed from the RNG.}
7:    else
8:       if uw ≺ vw ≺ uv or vw ≺ uw ≺ uv then
9:          N_RNG ← N_RNG(u) \ {v}
10:         {Link uv is removed from the RNG.}
11:      else
12:         {Link vw will be removed from the RNG.}
13: Return N_RNG(u)
```

Execution of this algorithm is distributed in the sense that each node is calculating its own set of RNG neighbors, according to given condition and knowing its neighborhood. As a result, we obtain a connected graph in which weaker edges have been removed and where disappearance of weak nodes has a minimal impact of the graph connectivity. Figure 3.4 compares the graph obtained after topology control when applying distance-based RNG (Fig. 3.4(a)) and our battery level based RNG (Fig. 3.4(b)). As we can see, in our approach, critical nodes appear either as leaves in the graph (nodes c, g, h, j) or in a redundant path (node c). Yet, if one of these nodes fail, the network is not impacted. At contrary, in the traditional RNG, there is no battery level concern and critical nodes belong to principal paths. If node b, c or g fails, the network is disconnected.

3.4 Experimental results

3.4.1 Experimentation Set Up

Experiments were run on the Lille SensLAB platform. We selected a 5×6-node grid via the SensLAB interface spread as depicted by Figure 3.5. In the SensLAB Lille platform nodes are placed in grid on the distance of $d = 60cm$ between each other. We chose a subset of nodes on the grid leaving out some of the nodes such that we can see what happens with physically longer links and how the algorithm is applied to them.

We use critical voltage level, $\tau = 3.7V$, which corresponds to value 1110 in the SVS register, *i.e.* we consider that a node has reached a low battery level as soon as the microcontroller sends an information that the critical voltage is reached. Experiments are run 12 hour long, during this time all sensors are loaded with same program which runs RNG algorithm 2, calculating RNG neighbors in distributed way, and recording statistics – neighbor candidates, RNG neighbors and the parameters for each link (power factor, RSSI value, IDs). For the implementation of our algorithm we use CSMA/CA MAC layer implementation provided by the SensTools project[1] and FreeRTOS port for MSP430 microcontrolers.

Our solution is compared to [Khadar 2009] from the literature. As detailed in Section 2.4, in [Khadar 2009], a RNG is built based on the RSSI on links after a first filter on neighborhood. Our algorithm also uses the RSSI as a metric to estimate at the same time distance and link quality as claimed in Section 2.4 but only as a secondary weight. We use the Power Factor metric as primary one. We do not apply the filter used in [Khadar 2009] since we assume that these bad links will be automatically removed in the RNG computation except if the removal of these links disconnects the network.

Figure 3.6 shows the edges connecting node 46 to its neighbors after . For the sake of visibility, we have represented only these links.

Figure 3.7 shows communication links after a topology control performed with our solution (Fig. 3.7(a)) and with RSSI-based RNG with filters (Fig. 3.7(b)) like in [Khadar 2009]. At these pictures black edges are RNG edges of the node 46, and yellow edges are RNG edges of the rest of the nodes, ensuring the connectivity between node 46 and the rest of the nodes in its neighborhood. At this step, all nodes have a high battery level and thus both graphs are equivalent. We are running test program on the nodes making them exhaust the battery, this program is just made for the nodes to exchange messages and faster discharge batteries thus to speed up the experiment.

Figure 3.8 shows the final topology control after some time, when some of

[1] http://senstools.gforge.inria.fr/

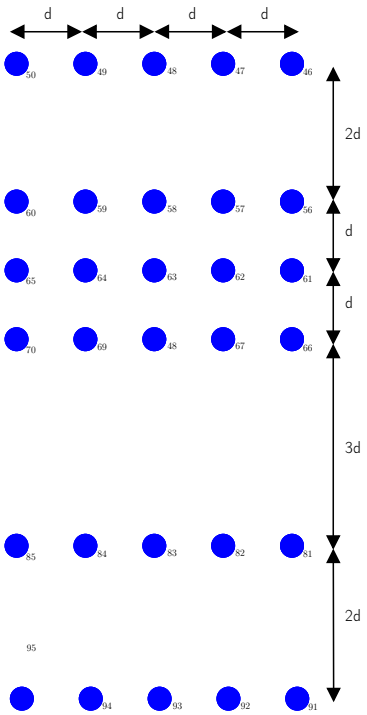


Figure 3.5: Placement of the nodes on which the experiment is run. In SensLAB, $d = 60cm$.

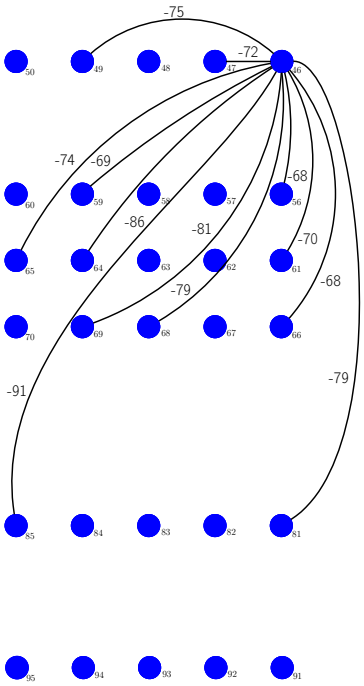
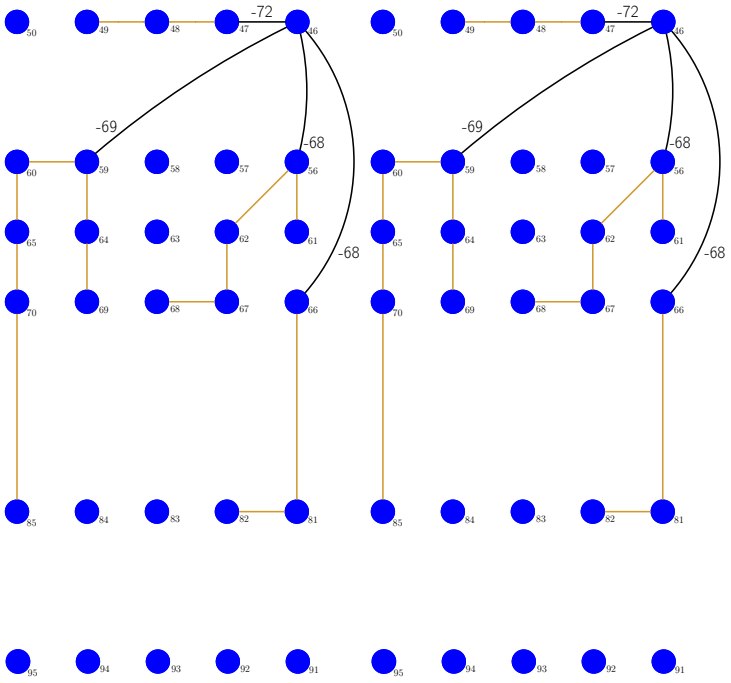


Figure 3.6: Initial topology of graph.



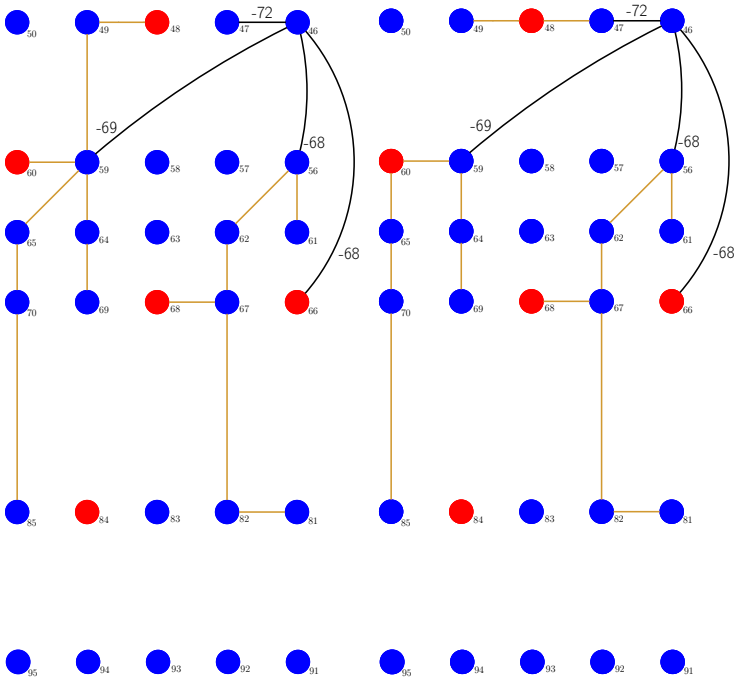
(a) Battery-level based RNG (b) RSSI-based RNG + filters

Figure 3.7: Topology after topology control.

the nodes exhausted their batteries and brought them to the critical state. We can see that with our solution (Fig. 3.8(a)), the network is still connected and that weak nodes appear as leaves in the reduced graph. We can also see that the network dynamically reorganized itself: link $46 - 47 - 48$, which allows node 46 to reach node 48, has been changed for link $46 - 59 - 49 - 48$ when the battery of node 48 has dropped under critical level. Furthermore, if we consider complete discharge of the battery of the node 48 in this case if we are using RSSI-based RNG then we will lose of connectivity between nodes 46 and 49 while battery-level based RNG is preserving connectivity between those two nodes.

3.4.2 Node degree and connectivity preservation using 3 different RNG algorithms

In this section we compare results obtained by our battery driven RNG algorithm with two RNG algorithms presented in [Khadar 2009]. On the figures results marked with BatRNG presents results using battery driven RNG algorithm, RSSI-RNG presents algorithm which is based solely on RSSI as the weight function and F-RSSI-RNG present algorithm which as a first step uses primary filter which filters "bad nodes", *i.e.* nodes which have received mes-



(a) Battery-level based RNG (b) RSSI-based RNG + filters

Figure 3.8: Topology after topology control after nodes have exhausted their battery.

sage with signal strength lower than some fixed value, in this way number of possible RNG neighbors for each node is additionally limited. In our experiments two limits for RSSI value, 70dBm and 80 dBM are used.

Figure 3.9 shows average node degree using different types of RNG algorithms in three distinct time moments, *i* in the beginning of experiment when all nodes approximately same battery level, *ii* at the moment when some of the nodes (7 - 12 nodes) have exhausted their battery and reached $\tau = 3.7V$ and *iii* at the moment when the most of the nodes have exhausted their battery (more than 20 nodes). The lowest node degree is obtained by the algorithm F-RSSI-RNG, which can be explained with the usage of additional filter before applying of the RNG algorithm. It has to be noted that the average value includes nodes which have 0 neighbor after application of the primary filter and which are basically disconnected from the rest of the network, producing disconnected network. If the value for the primary filter is too high number of disconnected grows, leaving the graph disconnected before application of RNG algorithm. BatRNG and RSSI-RNG at the time moments 1 and 3 have almost the same values for node degree which can be explained with almost same battery level for the majority of the nodes in these time moments. In this case the algorithm which we propose, BatRNG, uses as the second metric RSSI value which in this case dominates leading to the similar results to RSSI-RNG algorithm. Time moment 2 is specific since the change can be noticed only in the BatRNG algorithm, in this moment the graph reconfigures itself according to our weight function and RNG algorithm leaving the nodes with higher battery with more neighbors and nodes with lower battery level with only one or just two neighbors (only in the case when node with low battery level can not reach the one with high battery level – in this case it chooses the neighbor with low battery level and the best RSSI value).

Figure 3.10 shows average number of connected nodes during our experiments. The first thing that we can notice is that number of connected nodes is always lower than number used in experiments this is due to the hardware problems of the SensLAB platform that we used. Some of the nodes which have been used we unaccessible and we could not use them in the experiment thus we have reduced number of connected nodes. Second thing to notice is significantly lower number of connected nodes when using F-RSSI-RNG algorithm, this can be explained with the average value which we obtained in the experiments with both strict RSSI limit (70 dBm) and less strict RSSI limit (80 dBm). In this figure we have 10 distinct time moments, first five of them are with all nodes alive and we can see that there is practically no change in the number of connected nodes. From the time moments 6 to 10 we were looking what is happening when nodes are starting to lose their battery power and to switch of from the network. We can see that the fastest drop in

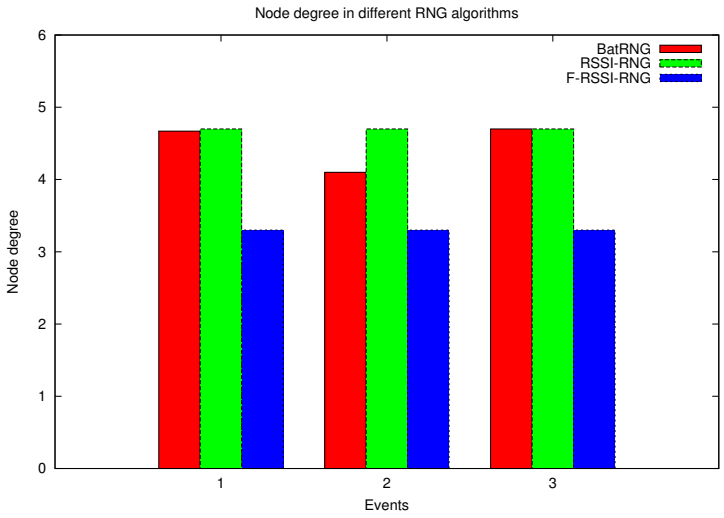


Figure 3.9: Node degrees for different RNG algorithms

the number of connected nodes have F-RSSI-RNG algorithm which again can be explained with primary filter which removes possible neighbors more strict leaving more nodes without neighbors. We can also notice that BatRNG evaluates better than RSSI-RNG algorithm which can be explained with previous reconfiguration of the graph due to the critical change in battery level.

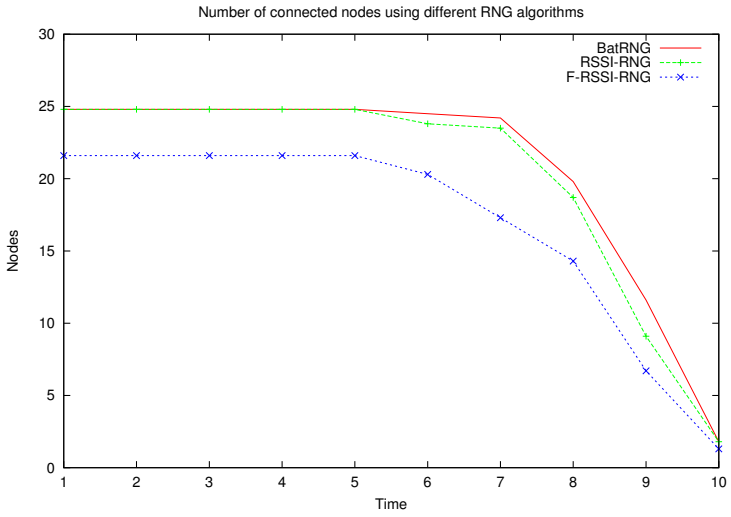


Figure 3.10: Number of connected nodes during execution of different RNG algorithms

3.5 Conclusion

In this chapter we proposed a topology control algorithm of wireless sensor network based on RNG reduction aware of dynamic battery level of nodes. As weight function we have introduced level of the battery through the power factor which is calculated for each link of the starting graph. Our algorithm favors the links in which at least one of the nodes have high battery level. In this way the nodes which have low battery level appear in graph more often as leafs or with more links but always with the nodes with higher battery level. Reconfiguring the network in such a way helps in pertaining connectivity in critical cases when node depletes its battery. For future works, it will be interesting to study the behavior of algorithms in presence of traffic.

Possible advances and future work could take into account multiple battery levels reconfiguring the network on more levels. Also secondary weight function (distance) used in this case is retrieved using the RSSI which used together with power factor does not insure planarity of obtained structure, so the future work could include additional conditions which would, along with used weight function, guarantee planarity of the graph.

CHAPTER 4

Neighborhood discovery in mobile wireless sensor networks

Contents

4.1 Introduction

Mobility is one of the important properties of wireless sensor networks. It is not immanent to all wireless sensor networks but often it is seen as the issue which has to be taken into account when designing specific solutions for wireless sensor networks. There are many different approaches to the mobility issue, some of them are trying to adapt networks to various mobility patterns, some of them treat mobility as an inherent property to which everything has to be adjusted and the others are trying to use mobility to improve other properties of network (connectivity, routing) [Wang 2009, Loscrì 2010].

Nodes can loose some of their neighbors after a period of time for numerous reasons: mobility, interferences, loss of power supply, duty cycling. As shown in Section 2.3.2, neighborhood discovery depends on the mobility of the nodes *i.e.* if nodes are mobile there is a probability [Ingelrest 2007] that after a given period of time a node will discover certain neighbors and at the same time it will loose some of the previous ones. To have accurate and up to date neighborhood table, a node must check validity of the elements periodically[Ahmad Kassem 2010] and adapt its HELLO Messages frequency in order to give accurate information to its potential neighbors [Ingelrest 2007]. The efficiency of table-driven protocols (*e.g.* routing, clustering, activity scheduling) obviously relies on the accuracy of these tables.

In a mobile environment, neighborhood tables are subject to changes, and in this case, the accuracy depends on the frequency of HELLO messages. If this frequency is too low, nodes may not be detected by their neighbors, leading deprecated neighborhood tables, and protocol failures are likely to occur. On the other hand, if the frequency is too high, neighborhood tables are up to date, but then energy and bandwidth are wasted to the detriment of data traffic.

Periodic sending of HELLO messages comes with a certain cost in energy. Although there exists an optimal frequency of HELLO messages for a given speed of the nodes [Troel 2004] and good results [Ingelrest 2007] in adaptation of frequency of HELLO messages in accordance with the number of new neighbors that each node discovers during given period of time, there is no result which would take into account all three values: speed, number of new neighboring nodes and energy consumed. In this chapter we present algorithms that are based both on the adaptation of frequency of HELLO messages and on the minimization of the energy cost by range adjustment. When we talk about the energy, we are not taking into account energy needed for the nodes to move, we rather assume that mobility is inherent to the nodes. We minimize overall energy consumption balancing the appropriate level of transmission power and value of frequency of HELLO messages.

In this chapter, we introduce two algorithms that adapt both the HELLO frequency and the range of nodes to provide them accurate neighborhood tables while minimizing the cost of the neighbor discovery. First algorithm is based on the TAP algorithm from the literature [Ingelrest 2007]. It thus computes a turnover based on the observation of changes in its neighborhood and then proposes two variants to adapt the frequency and the range of nodes. First variant adapts the HELLO frequency in a TAP-fashion. Second variant adapts the HELLO frequency through the computing of minimum cost. Both variants then set the range accordingly based on optimal HELLO frequency [Troel 2004]. Second algorithm tries to set at the same time HELLO frequency and range based on theoretical analysis.

To summarize, main aim of this chapter is to present energy efficient neighborhood discovery using mutual adaptation of transmission range and frequency of HELLO messages. This goal is retrieved using the results of optimal HELLO frequency, turnover based approach and theoretical analysis on the turnover and energy cost of sending of HELLO messages.

This chapter is organized as follows. First section (Section 4.2) presents a theoretical analysis which allows to estimate the optimum turnover and the optimal cost. This analysis is further used by algorithms. Then Section 4.3 details the different algorithms. Finally, simulation results are displayed in Section 4.4.

4.2 Theoretical analysis

4.2.1 Probable number of new neighbors to compute the turnover

This analysis is used to find probable number of new neighbors in a given period of time Δt with certain assumptions imposed on the type of deployment and the mobility of sensors. Even though it is constrained to a specific case, with just one type of probability of placement of the sensors and the type of mobility it serves us to roughly evaluate possible values as well as to get probable outcome of algorithm and possible points for its further improvement.It is inspired from [Ingelrest 2007] but extended to the case where nodes do not have the same transmission range.

Preliminaries and notations

We suppose that nodes are randomly deployed using a Poisson Point Process [Mališić 1989], with node positions which are independent and $\lambda > 0$, where λ represents the mean number of nodes per surface unit. Each node u has transmission range R_u such that $0 < R_u \leqslant R_{max}$ where R_{max} is the maximal transmission range.

We differentiate two types of neighbors:

- *bilateral neighbors* – u and v are such that $|uv| < min(R_u, R_v)$, in short if node u is neighbor of v then v is also neighbor of u,
- *unilateral neighbors* – u is unilateral neighbor of v iff $R_v < |uv| < R_u$, node v is neighbor of u but node u is not neighbor of v.

Every node moves at a constant speed V in a random direction. Position of node u at moment t_0 is given as u_0 (respectively node v at moment t_0 has position v_0) and at moment t_1 position is u_1 (respectively v_1). Distance covered by a node during the time Δt is given as $\Delta d = V \times \Delta t$.

In our analysis, we are interested in the mean number of new neighbors that node u meets during time interval Δt. We focus on a typical node u. Let $N_{bi}(u)_{\Delta t}$ be the number of new bilateral neighbors of node u and $N_{uni}(u)_{\Delta t}$ the number of new unilateral neighbors of node u detected during the period Δt. Let v be a node at the distance d ($d < R_{max}$) from node u at time $t_1 = t_0 + \Delta t$. We note:

- P^{bi} – the probability that node v is a new bilateral neighbor of node u
- P^{uni} – the probability that node v is a new unilateral neighbor of node u

From this we can determine the average values of $N_{bi}(u)_{\Delta t}$ and $N_{uni}(u)_{\Delta t}$:

$$\mathrm{E}[N_{bi}] = \int_{R_u=0}^{R_{max}} \int_{R_v=0}^{R_{max}} \int_{d=0}^{R_u} P(R_u)P(R_v)\lambda\pi d \times P^{bi}(d, R_u, R_v) ddd R_u d R_v \quad (4.1)$$

$$\mathrm{E}[N_{uni}] = \int_{R_u=0}^{R_{max}} \int_{R_v=0}^{R_{max}} \int_{d=0}^{R_u} P(R_u)P(R_v)\lambda\pi d \times P^{uni}(d, R_u, R_v) ddd R_u d R_v \quad (4.2)$$

where $P(R_u)$ is probability that u has radius R_u and $P(R_v)$ is probability that v has radius R_v

Figure 4.1 illustrates our model in case when $R_u < R_v$. Circle C_{u,R_u} is centered at the position of the node u with radius R_u. In this case, node v, with radius R_v is a new neighbor if and only if v_0 does not lie in the area delimited by C_{u_0,R_u} and if v_1 lies in the area delimited by C_{u_1,R_u}. The blue dashed circle $C_{u_1,\Delta d}$ and the red dotted circle $C_{v_1,\Delta d}$ represent the possible positions of the u_0 and v_0. Angles α and β are given as $\angle\overrightarrow{u_1,v_1},\overrightarrow{u_1,u_0}$ and $\angle\overrightarrow{u_1,v_1},\overrightarrow{v_1,v_0}$ respectively, and they represent the directions from which nodes u and v come. In the worst case, that we consider, node direction is random and thus α and β are uniformly distributed in $[-\pi, \pi]$.

Computing the number of new bilateral neighbors $\mathbf{E[N_{bi}]}$

We are interested in the probability that at time t_0, u and v were either not neighbors ($|u_0v_0| > max(R_u, R_v)$) or only unilateral neighbors ($max(R_u, R_v) > |u_0v_0| > min(R_u, R_v)$). This means that we are interested in probability P^{bi} that given R_u, R_v and d, $|u_0v_0| > min(R_u, R_v)$ knowing that $|u_1v_1| \leq min(R_u, R_v)$.
We make the distinction between two cases:

Case 1: $\mathbf{R_u \leq R_v}$ We note $P^{bi}_{R_u \leq R_v}$ the probability that v is a new bilateral neighbor of u if $R_u \leq R_v$; for this case $P(R_v) = \frac{1}{R_{max}}$ and $P(R_u \mid R_u < R_v) = \frac{R_v}{R_{max}}$,

Case 2: $\mathbf{R_u > R_v}$ We note $P^{bi}_{R_u > R_v}$ the probability that v is a new bilateral neighbor of u if $R_u > R_v$; for this case $P(R_v) = \frac{1}{R_{max}}$ and $P(R_u \mid R_u \leq R_v) = \frac{R_{max}-R_v}{R_{max}}$.

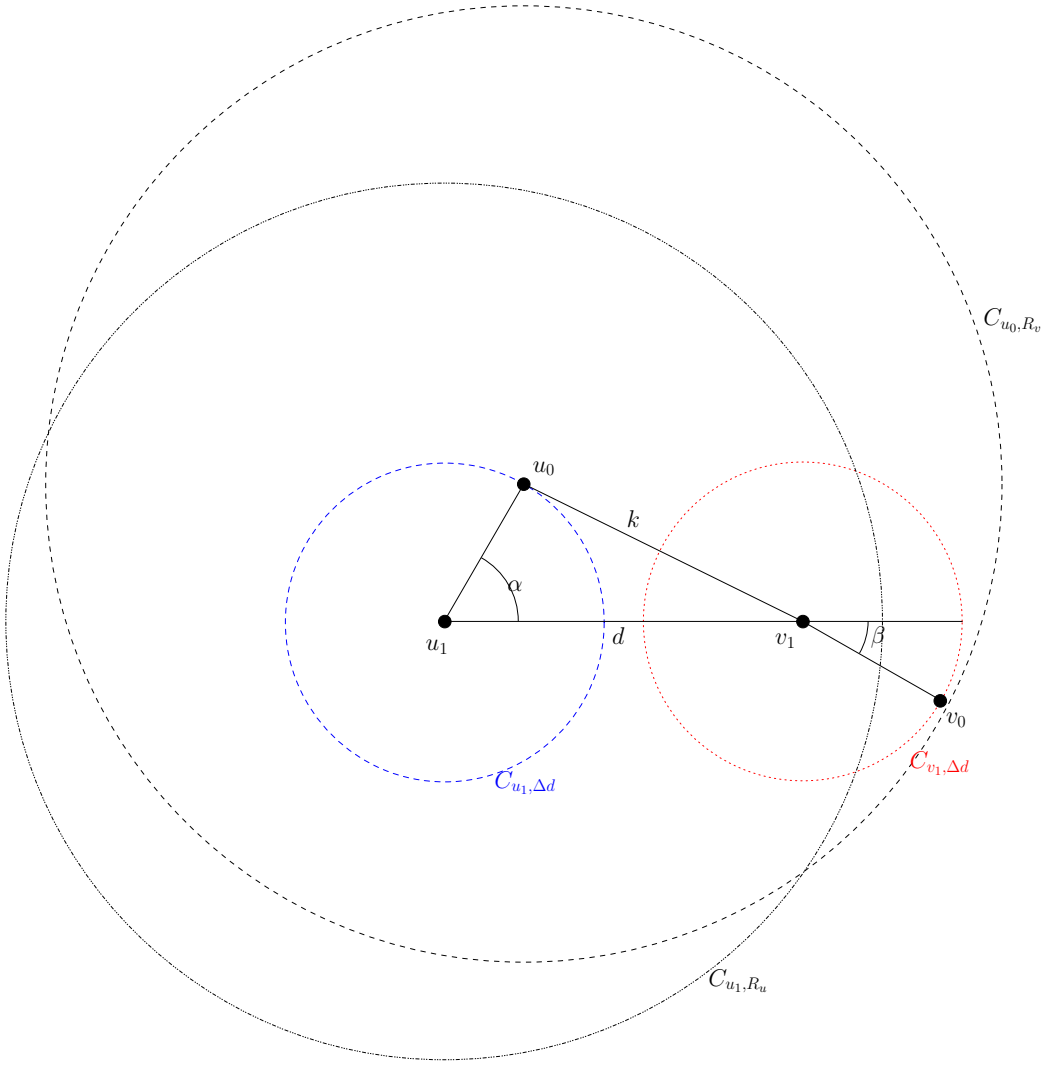


Figure 4.1: Global view. Circle C_{u,R_u} is centered at the position of the node u with radius R_u. In this case, node v, with radius R_v is a new neighbor if and only if v_0 does not lie in the area delimited by C_{u_0,R_u} and if v_1 lies in the area delimited by C_{u_1,R_u}. The blue dashed circle $C_{u_1,\Delta d}$ and the red dotted circle $C_{v_1,\Delta d}$ represent the possible positions of the u_0 and v_0.

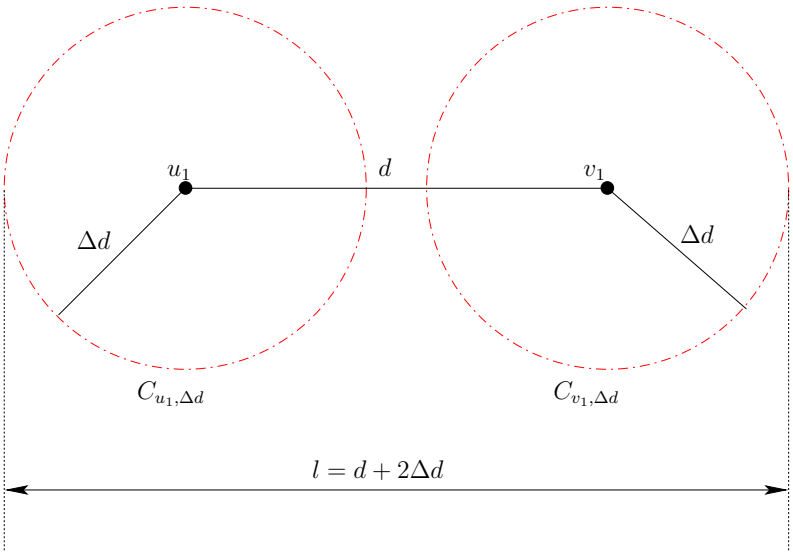


Figure 4.2: Calculating d_{min}

From these two cases we have:

$$\begin{aligned}
\mathrm{E}[N_{bi}] &= \int_{R_u=0}^{R_{max}} \int_{R_v=0}^{R_{max}} \int_{d=0}^{R_u} P(R_u)P(R_v)\lambda\pi d \times P^{bi}(d, R_u, R_v) dd dR_u dR_v \\
&= \int_{R_v=0}^{R_{max}} \int_{R_u=0}^{R_v} \int_{d=0}^{R_u} \frac{R_v}{R_{max}} \times \frac{1}{R_{max}} \times \lambda\pi d \times P^{bi}_{R_u \le R_v}(d, R_u, R_v) dd(dR_u) dR_v \\
&+ \int_{R_v=0}^{R_{max}} \int_{R_u=R_v}^{R_{max}} \int_{d=0}^{R_u} \frac{R_{max} - R_v}{R_{max}} \times \frac{1}{R_{max}} \times \lambda\pi d \times P^{bi}_{R_u > R_v}(d, R_u, R_v) dd(dR_u) dR_v
\end{aligned} \tag{4.3}$$

Case 1: $\mathbf{R_u \le R_v}$ First, we note that if $d \ge R_u$, $P^{bi}_{R_u \le R_v}(d) = 0$ since v is not a neighbor of u at a time t_1. Next, we find that there exists a value $d_{min} < R$ such that, if $d < d_{min}$, nodes u and v were already neighbors at time t_0 regardless of α and β *i.e.* v cannot be a new neighbor of u. As illustrated by Fig. 4.2, the longest distance l between u_0 and v_0 at time t_0 is when both node directions are opposite (for example when $\alpha = 0$ and $\beta = \pi$). We have $l = 2\Delta d + d$, where $\Delta d = \Delta t \times V$. For v to be a new neighbor of node u, we need $l > min(R_u, R_v)$, which leads to:

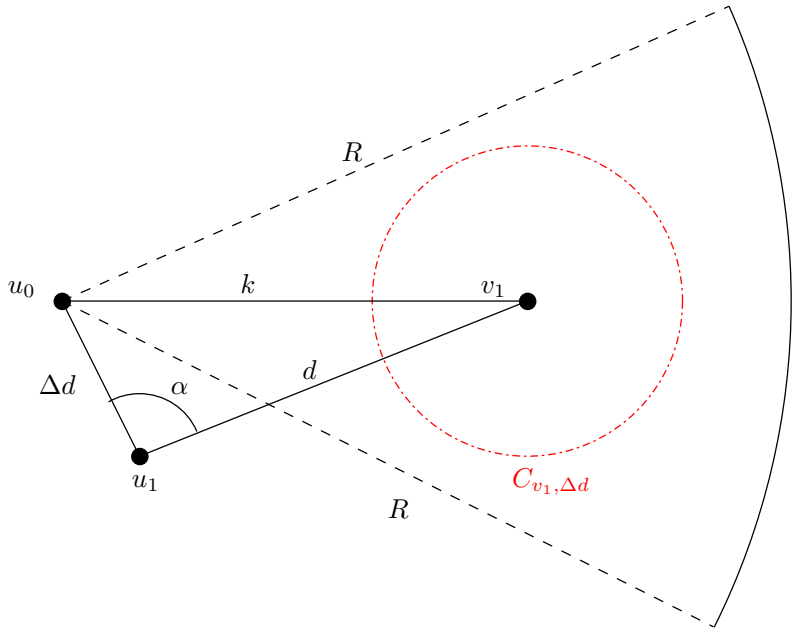


Figure 4.3: Calculating α_{min}

$$\begin{aligned} & 2\Delta d + d > min(R_u, R_v) \\ \Leftrightarrow\ & d_{min} = 2\Delta d - min(R_u, R_v) \end{aligned} \tag{4.4}$$

Since, depending on the node speed, we may have $2\Delta d < min(R_u, R_v)$, we finally get:

$$d_{min} = max(0, 2\Delta d - min(R_u, R_v)) \tag{4.5}$$

Since we have $R_u \leq R_v$,

$$d_{min} = max(0, 2\Delta d - R_u)$$

Using Eq.4.5 we have:

$$P_{bi}(d, R_u, R_v) = \begin{cases} \int_{-\pi}^{\pi} P^{bi}_{R_u \leq R_v}(d, R_u, \alpha) d\alpha & \text{if } d_{min} < d < R_u \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

where $P^{bi}_{R_u \leq R_v}(d, R_u, \alpha)$ is the probability that node v with radius $R_v \geq R_u$ at distance d from u is a *new* bilateral neighbor of node u with radius R_u coming from direction α, assuming that $d_{min} < d < R_u$.
Now, we can compute $P^{bi}_{R_u \leq R_v}(d, R_u, \alpha)$. For this purpose, we notice that, for a given value of d such that $d_{min} < d < R_u$, there exists a value α_{min} such

that, for $\alpha < \alpha_{min}$, node v was a bilateral neighbor of the node u at time t_0 regardless of its direction β, thus it cannot be a *new* neighbor. This is illustrated on Figure 4.3. Indeed, for $\alpha < \alpha_{min}$, the whole circle $C_{v_1,\Delta d}$ lies inside circle C_{u_0,R_u} thus, for any direction that node v may had it was already in the bilateral neighborhood of node u. As a result we have:

$$P^{bi}_{R_u \leq R_v}(d, R_u, \alpha) = 0 \quad \text{if} \quad \alpha < \alpha_{min}$$

α_{min} is computed using Figure 4.3 knowing that with given $d > d_{min}$ and $\alpha < \alpha_{min}$ then for any β (no matter the value which it takes) v was neighbor of u at time t_0.

For it, we introduce k such that $k = |u_0 v_1|$, like illustrated on Figure 4.3. According to Pythagore's theorem, we can compute the value of k:

$$\begin{aligned} \Delta d^2 - (\Delta d \cos(\pi - \alpha))^2 &= k^2 - (d + \Delta d \cos(\pi - \alpha)) \\ \Leftrightarrow \; k^2 &= \Delta d^2 + d^2 - 2d\Delta d \cos\alpha \\ \Leftrightarrow \; k &= \sqrt{\Delta d^2 + d^2 - 2d\Delta d \cos\alpha} \end{aligned} \tag{4.7}$$

For v to be a new bilateral neighbor of node u we need that $|u_0 v_0| > min(R_u, R_v)$ for any β *i.e.* that the disks delimited by $C_{v_1,\Delta d}$ and $C_{u_0,min(R_u,R_v)}$ overlap:

$$\begin{aligned} k - \Delta d &> min(R_u, R_v) \\ \Leftrightarrow \; k^2 &> (min(R_u, R_v))^2 + \Delta d^2 + 2\Delta d \cdot min(R_u, R_v) \\ \Leftrightarrow \; d^2 - 2d\Delta d \cos\alpha &> (min(R_u, R_v))^2 + 2\Delta d \cdot min(R_u, R_v) \\ \Leftrightarrow \; \cos\alpha &< \frac{d^2 - (min(R_u, R_v))^2 - 2\Delta d \cdot min(R_u, R_v)}{2d\Delta d} \\ \Leftrightarrow \; \alpha &> \arccos\left(\frac{d^2 - (min(R_u, R_v))^2 - 2\Delta d \cdot min(R_u, R_v)}{2d\Delta d}\right) \\ \Leftrightarrow \; \alpha_{min} &= \arccos\left(\frac{d^2 - (min(R_u, R_v))^2 - 2\Delta d \cdot min(R_u, R_v)}{2d\Delta d}\right) \end{aligned} \tag{4.8}$$

For any α, such that $\alpha > \alpha_{min}$, computing the probability $P^{bi}_{R_u \leq R_v}(d, R_u, \alpha)$ amounts to computing the probability that node v is coming from the dotted blue angular sector on Figure 4.4. Node v is a new neighbor of node u if and only if β is such that $|u_0 v_1| > R_u$ *i.e.* such that v_0 is outside of the circle C_{u_0,R_u}. In this case node v is a neighbor of u if and only if $\beta^- < \beta < \beta^+$, where β^- and β^+ are the angles of the intersection points between C_{u_0,R_u} and $C_{v_1,\Delta d}$, as illustrated on Figure 4.4. As a result we have:

$$P^{bi}_{R_u \leq R_v}(d, R_u, \alpha) = \int_{\beta^-}^{\beta^+} \frac{d\beta}{2\pi} = \frac{\beta^+ - \beta^-}{2\pi} \tag{4.9}$$

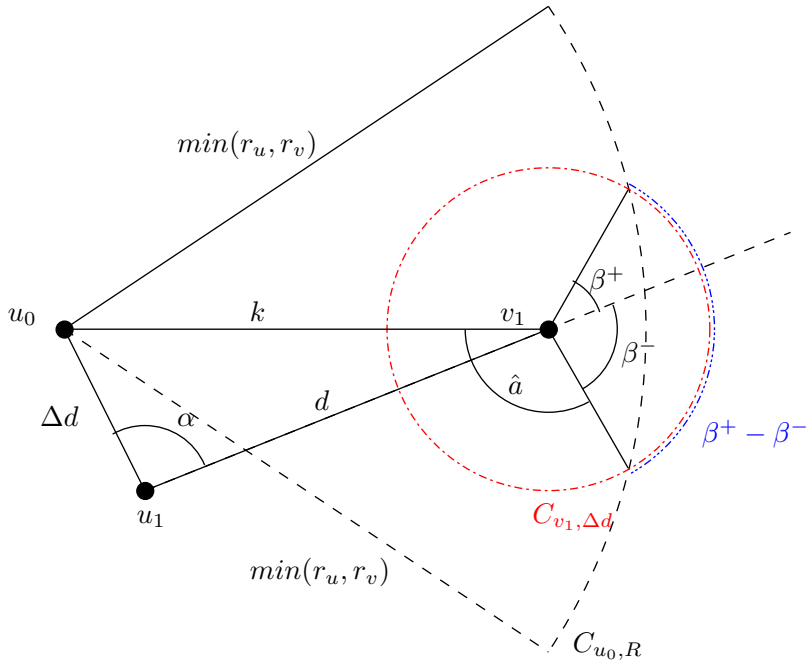


Figure 4.4: Zoom view

Computation of $\beta^- - \beta^+$ are given with notations from Figure 4.4. We can notice that:

$$2\pi = (\beta^+ - \beta^-) + 2\hat{a} \tag{4.10}$$

Since we consider bilateral neighbors we have:

$$\hat{a} = \arccos(\frac{\Delta d^2 + k^2 - min(R_u, R_v)}{2k\Delta d})$$

$$\Leftrightarrow \ (\beta^+ - \beta^-) = 2\arccos(\frac{min(R_u, R_v) - \Delta d^2 - k^2}{2k\Delta d})$$

Since we have $R_u \leq R_v$,

$$\Leftrightarrow \ (\beta^+ - \beta^-) = 2\arccos(\frac{R_u - \Delta d^2 - k^2}{2k\Delta d}) \tag{4.11}$$

Thus, when $d > d_{min}$:

$$P^{bi}_{R_u \leq R_v}(d, R_u, \alpha) = \begin{cases} \frac{1}{\pi}\arccos(\frac{R_u^2 - \Delta d^2 - k^2}{2d\Delta d})d\alpha & \text{if } \alpha > \alpha_{min} \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

From Eq.4.6, we derive:

$$P^{bi}_{R_u \leq R_v}(d, R_u, R_v) = \begin{cases} \frac{2}{\pi^2}\int_{\alpha_{min}}^{\pi} \arccos(\frac{R_u^2 - \Delta d^2 - k^2}{2d\Delta d})d\alpha & \text{if } d_{min} \leq d \leq R_u \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

with $\alpha_{min} = \arccos(\frac{d^2-2R_u\Delta d-R_u^2}{2d\Delta d})$ and $d_{min} = max(0, R_u - 2\Delta d)$.

Case 2: $\mathbf{R_u > R_v}$ Computing of $P^{bi}_{R_u>R_v}(d, R_u, R_v)$ is similar to the computing of the $P^{bi}_{R_u\leq R_v}(d, R_u, R_v)$. The differences mainly are in the values of d_{min}, $(\beta^+ - \beta^-)$ and α_{min} as shown in previous section. As a final result we get:

$$P^{bi}_{R_u>R_v}(d, R_u, R_v) = \begin{cases} \frac{2}{\pi^2} \int_{\alpha_{min}}^{\pi} \arccos(\frac{R_v^2-\Delta d^2-k^2}{2k\Delta d})d\alpha \\ \qquad \text{if } d_{min} \leq d \leq R_u \text{ and} (R_v - 2\Delta d) < R_u \\ 0 \qquad \text{otherwise} \end{cases} \tag{4.14}$$

with $\alpha_{min} = \arccos(\frac{d^2-2R_v\Delta d-R_v^2}{2d\Delta d})$ and $d_{min} = max(0, R_v - 2\Delta d)$.

Computing the number of new unilateral neighbors $\mathbf{E[N_{uni}]}$

Value that is interesting for our analysis the probability P^{uni} that at time t_0, when u and v were not neighbors given R_u, R_v and d. Once again, we distinguish two cases:

- **Case 1: $\mathbf{R_u \leq R_v}$** We note $P^{uni}_{R_u\leq R_v}$ the probability that v is a new unilateral neighbor of u if $R_u \leq R_v$; for this case $P(R_v) = \frac{1}{R_{max}}$ and $P(R_u \mid R_u < R_v) = \frac{R_v}{R_{max}}$,
- **Case 2: $\mathbf{R_u > R_v}$** We note $P^{uni}_{R_u>R_v}$ the probability that v is a new unilateral neighbor of u if $R_u > R_v$; for this case $P(R_v) = \frac{1}{R_{max}}$ and $P(R_u \mid R_u \leq R_v) = \frac{R_{max}-R_v}{R_{max}}$

From these two cases we have:

$$\begin{aligned} E[N_{uni}] &= \int_{R_u=0}^{R_{max}} \int_{R_v=0}^{R_{max}} \int_{d=0}^{R_u} P(R_u)P(R_v)\lambda\pi d \times P^{uni}(d, R_u, R_v)dddR_udR_v \\ &= \int_{R_u=0}^{R_v} \int_{R_v=0}^{R_{max}} \int_{d=0}^{R_u} \frac{R_v}{R_{max}} \times \frac{1}{R_{max}} \times \lambda\pi d \times P^{uni}_{R_u>R_v}(d, R_u, R_v)dddR_udR_v \\ &+ \int_{R_u=R_v}^{R_{max}} \int_{R_v=0}^{R_{max}} \int_{d=0}^{R_u} \frac{R_{max} - R_v}{R_{max}} \times \frac{1}{R_{max}} \times \lambda\pi d \times P^{uni}_{R_u>R_v}(d, R_u, R_v)dddR_udR_v \end{aligned} \tag{4.15}$$

Case 1: $\mathbf{R_u \leq R_v}$ In such a case, probability for the node v to be a new unilateral neighbor of the node u is null. Indeed, since $R_u \leq R_v$, this is indeed u which is bilateral neighbor of the node v and not the opposite, thus we have:

$$P^{uni}_{R_u>R_v}(d, R_u, R_v) = 0 \tag{4.16}$$

Case 2: $\mathbf{R_u > R_v}$ We compute the probability $P^{uni}_{R_u>R_v}$ for the node v to be a new unilateral neighbor of node u knowing R_u, R_v and d. This study is similar to the one given for the bilateral neighbors with the differences in values for d_{min}, β^+, β^- and α_{min} as it is given in the following paragraphs.

$$P^{uni}_{R_u>R_v}(d, R_u, R_v) = \begin{cases} \frac{2}{\pi^2} \int_{\alpha_{min}}^{\pi} \arccos(\frac{R_u^2 - \Delta d^2 - k^2}{2k\Delta d}) & if d_{min} \leq d \leq R_u \\ 0 \quad otherwise \end{cases} \tag{4.17}$$

with $\alpha_{min} = \arccos(\frac{d^2 - 2R_u\Delta d - R_u^2}{2d\Delta d})$ and $d_{min} = max(R_v, R_u - 2\Delta d)$.

Calculating $\mathbf{d_{min}}$, $(\beta^+ - \beta^-)$ and $\alpha_{\mathbf{min}}$ for new unilateral neighbors To calculate d_{min} we start from the definition of a new unilateral neighbor v of node u, if d is such that $d > R_v$. Then, similarly to the case for the bilateral neighbors, d_{min} is such that if $d > d_{min}$, nodes u and v were already neighbors at the time t_0, whatever the values α and β. Since, depending on the node speed, we may have $2\Delta d < min(R_u, R_v)$, we finally get:

$$d_{min} = max(R_v, 2\Delta d - min(R_u, R_v)) \tag{4.18}$$

Value $\beta^+ - \beta^-$ is obtained using the variables from Figure 4.4. In the case of the unilateral neighbors we consider

$$\hat{a} = \arccos(\frac{\Delta d^2 + k^2 - max(R_u, R_v)}{2k\Delta d}) = \arccos(\frac{\Delta d^2 + k^2 - R_u}{2k\Delta d})$$

deducing that:

$$(\beta^+ - \beta^-) = 2\arccos(\frac{R_u - \Delta d^2 - k^2}{2k\Delta d}) \tag{4.19}$$

For α_{min}, we, again, start from the definition of the new unilateral neighbor v of the node u in which we need $|u_0v_0| > max(R_u, R_v)$ for any β *i.e.* that

the disks delimited by $C_{v_1,\Delta d}$ and $C_{u_0,max(R_u,R_v)}$ overlap:

$$\begin{aligned}
& k - \Delta d > max(R_u, R_v) \\
\Leftrightarrow\ & k - \Delta d > R_u \\
\Leftrightarrow\ & k^2 > R_u^2 + \Delta d^2 + 2\Delta d \cdot R_u \\
\Leftrightarrow\ & d^2 - 2d\Delta d \cos\alpha > R_u^2 + 2\Delta d \cdot R_u \\
\Leftrightarrow\ & \cos\alpha < \frac{d^2 - R_u^2 - 2\Delta d \cdot R_u}{2d\Delta d} \\
\Leftrightarrow\ & \alpha < \arccos(\frac{d^2 - R_u^2 - 2\Delta d \cdot R_u}{2d\Delta d}) \\
\Leftrightarrow\ & \alpha_{min} = \arccos(\frac{d^2 - R_u^2 - 2\Delta d \cdot R_u}{2d\Delta d})
\end{aligned} \tag{4.20}$$

Computing the turnover

Similarly to TAP [Ingelrest 2007], we use this analysis to determine the optimum turnover on which nodes dynamically adapt their frequency. Since in our work all nodes do not have the same transmission range, we need to adapt the turnover to unilateral neighbors. The number of new neighbors during period Δt and depends on the speed of nodes (through parameter $\Delta d = V \times \Delta t$) and the Hello Frequency. The optimal turnover can be achieved when the Hello frequency is optimal. Since the optimal Hello frequency depends on the speed, the speed parameter is canceled in the optimal turnover formula which does not depend anymore of the speed. Every result presented above is unfortunately non closed formula. Nevertheless, some numerical results can be found with regards with different values of R_{max}.

4.2.2 Analysis on minimization of energy cost

In this chapter, we analyze the frequency and transmission range with regards to the optimum energy consumption. The energy spent in period Δt of time by a node u can be expressed as the number of messages sent by u in Δt multiplied by the cost of a message. The number of messages sent by u during Δt is $\Delta t \cdot f_u(R_u, t)$ where $f_u(R_u, t)$ is the Hello frequency of node u and R_u is the range of node u at time t. The cost of a message follows energetic model introduced in Section 2.3.3 and is as follows: $E(R_u) = R_u(t)^\alpha + C$. We assume that Δt is such that u does not change its range nor its frequency during this period of time, so $R_u(t) = R_u$. Thus, energy spent by node with transmission range R during Δt is:

$$cost_{\Delta t}(R) = \Delta t \cdot f(R) \times (R^\alpha + C) \tag{4.21}$$

Note that the Hello frequency also depends of transmission range. The higher transmission range, the lower Hello frequency. Also note that in the following analysis we have omitted units for the values that we are using in the analysis. Our main aim in this analysis is to make connection between consumption, which is given in *Joules* $[J]$, frequency of Hello messages, given in *Herz* $[Hz = \frac{1}{s}]$, and transmission range, given in *meters* $[m]$, while the rest of values and constants are expressed in such units so that they comply with rest of values in equations. For example, in equation 4.21 units for $cost_{\Delta t}$ and C are given in $[J]$ and also it should be noted that R^α should be multiplied by constant 1 with units $[J/m^\alpha]$ to comply with other values.

In order to find appropriate transmission range we need to find R that gives the minimum of the given function.

$$\frac{\partial cost(R)}{\partial R} = \frac{\partial}{\partial R}\left(\Delta t \cdot f(R) \times (R^\alpha + C)\right) \tag{4.22}$$

Partial derivative given is applied in order to get the minimum of the cost function:

$$\frac{\partial cost(R)}{\partial R} = \Delta t \alpha f(R) \times R^{\alpha-1} + \Delta t \frac{\partial f(R)}{\partial R} \times (R^\alpha + C) \tag{4.23}$$

minimum of this function is obtained when

$$\frac{\partial cost}{\partial R} = 0 \tag{4.24}$$

which gives us

$$\frac{\partial f(R)}{\partial R} = -\alpha f(R) \frac{R^{\alpha-1}}{R^\alpha + C} \tag{4.25}$$

This differential equation gives as a solution:

$$f\left(R\right) = \frac{C_1}{R^\alpha + C} \tag{4.26}$$

where C_1 is a constant defined by initial conditions. For parameters used later on in our simulations, *e.g.* $\alpha = 4$, $C = 10^8$ given in [Fleury 2009] and initial condition such that $C_1 = 2 \times 10^8$, it becomes:

$$f\left(R\right) = \frac{200000000}{R^4 + 100000000} \tag{4.27}$$

Optimum function $f(R)$ for energy consumption is depicted on Figure 4.5. Note that, as expected, the Hello frequency decreases when the range increases.

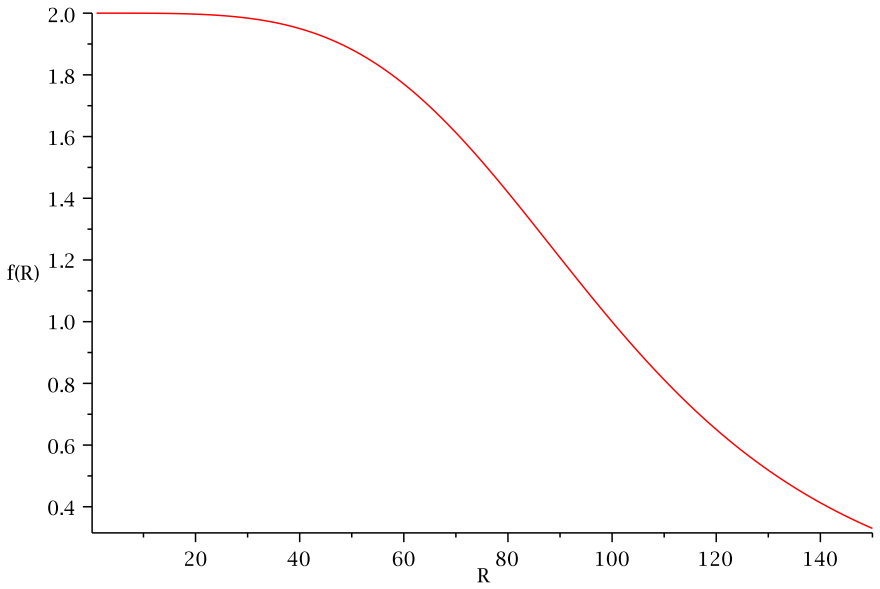


Figure 4.5: Solution of differential equation obtained for optimal energy cost.

4.3 Algorithms

In this section we present our contribution and the algorithms used to adapt both frequency of HELLO messages and transmission range. The first algorithm is based on [Ingelrest 2007] which relies on the calculation of turnover. The turnover of [Ingelrest 2007] has been redefined to capture both unilateral and bilateral neighbors (that appear because of different transmission ranges) along with optimal frequency of HELLO messages [Troel 2004]. This turnover helps in determining the Hello frequency. Algorithm is then declined in two variants to define the transmission range. Second algorithm is combining results for optimal frequency and dependency between f and transmission range to adapt both frequency and transmission range without knowledge of turnover.

Both algorithms are based on the existence of an optimal frequency of HELLO messages [Troel 2004]. Optimal frequency of HELLO messages is function of the relative speed of the nodes V, and their transmission range R multiplied by corrective factor a:

$$f_{opt} = \frac{2V}{aR} \tag{4.28}$$

4.3.1 Using the turnover

In order to define this algorithm first step is to calculate turnover, r, in a real scenario. Number of new neighbors (theoretically calculated in Section 4.2.1) is translated into the calculation of number of new neighbors with given weights.

- unilateral neighbors – the situation when the node finds new unilateral neighbor; it is quantified with θ_{uni} multiplied by the total number of new unilateral neighbors in given period of time,
- bilateral neighbors – the situation in which the node finds new bilateral neighbor; factor θ_{bi} is used to quantify new bilateral neighbor which is found and it is multiplied by total number of new bilateral nodes y found in given period of time.

Current turnover, r, is calculated by each node independently according to their own track of the changes between the types of the links, using equation:

$$r = \frac{\theta_{uni} \cdot x + \theta_{bi} \cdot y}{x_t + y_t} \cdot \frac{T_{HELLO}}{\Delta t} \tag{4.29}$$

where $\theta_{uni} = 1$, $\theta_{bi} = 2$, Δt is the time passed between updates of two tables that we are comparing, T_{HELLO} is the period of HELLO messages, $x_t + y_t$ presents total number of neighbors in neighborhood table and it is the sum of all unilateral and bilateral neighbors. These specific values for the θ parameters are used with the respect of the type of neighbor. For new unilateral neighbors we use smaller value of θ because we consider those links weaker and we want to give them less importance in calculation of turnover. Bilateral neighbors are considered stronger and they are multiplied by bigger θ.

First solution is detailed in Algorithm 3 *Turnover based Power Transmission Adjustment*. Algorithm 3 is executed by each node independently only based in the observation of is neighborhood. Algorithm 3 is TAP-fashion algorithm which adapt the hello frequency dynamically based on changes on node neighborhood. It aims to reach an optimal turnover previously computed thanks to computing of new neighbors provided in Section 4.2.1. Starting point for the algorithms are neighborhood table and *history table*, and all other values are calculated using these values, including the turnover. Neighborhood table is the standard neighborhood table, as explained in Section 2.3.2. History table is table in which we are preserving the values which we have obtained for a given time moment, number of new bilateral and unilateral neighbors and the changes between the type of the neighbors which allows us to calculate the turnover at that moment.

Adjustment of f is calculated through the period between two HELLO messages d_{HELLO}, where $f = \frac{1}{d_{HELLO}}$:

$$d_{HELLO} = \begin{cases} d_{HELLO} + \frac{d_{HELLO}}{4} \cdot g(r) & \text{if } r \leq r_{opt} \\ d_{HELLO} - \frac{d_{HELLO}}{4} \cdot g(r) & \text{otherwise} \end{cases} \tag{4.30}$$

Function $g(r)$ is retrieved using turnover:

$$g(r) = \begin{cases} (\frac{r - r_{opt}}{r_{opt}})^2 & \text{if } r < 2 \cdot r_{opt}, \\ 1 & \text{otherwise} \end{cases} \tag{4.31}$$

Algorithm 3 Turnover based Power Transmission Adjustment

1: **while** 1 **do**
2: **CalculateCurrentTurnover** $r = \frac{\theta_{uni} \cdot x + \theta_{bi} \cdot y}{x_t + y_t} \cdot \frac{T_{HELLO}}{\Delta t}$
3: **if** $r \leq r_{opt}$ **then**
4: *Lower* f with Eq. 4.30
5: **else if** $r > r_{opt}$ **then**
6: *Augment* f with Eq. 4.30
7: *Adapt Power of Transmission*

Two variants differ in the call to *Adapt Power of Transmission* function in Line 7 of Algorithm 3. First variant uses Eq. 4.28 while second variant uses the analysis of Section 4.2.2.

4.3.2 Using the optimal frequency of HELLO messages

The first variant assumes f obtained at Line 6 of Algo. 3 as f_{opt} (Eq. 4.28 and from this gives the information for the new value of power of transmission. It runs Algorithm 4 as *Adapt Power of Transmission* function.

Algorithm 4 Power Transmission Adjustment - Variant 1

1: Return $R = \frac{2V}{af}$

4.3.3 Minimizing the cost

The second variant adjusts the transmission range based on minimum energy consumption as defined by Eq. 4.26. It runs Algorithm 5 as *Adapt Power of Transmission* function.

Algorithm 5 Power Transmission Adjustment - Variant 2

1: Return $R = \sqrt[\alpha]{\frac{C_1}{f} - C}$

4.3.4 Minimizing the energy consumption

We now describe our second algorithm which is detailed in Section 6. It combines the results of optimal frequency and minimized energy cost and calculates R and f solely upon these two values based on Equations 4.26 and 4.28.

Algorithm 6 Cost Based Transmission Power Adjustment

1: To get R, solve $R^\alpha - \frac{aR}{2VC_1} + C = 0$ based on α values.
2: $f = \frac{2V}{aR}$
3: Return (f,R)

Note that there exists no real solution for $\alpha = 2$. The formula is not closed but we could solve it with numerical results. For parameters used later on in our simulations, *e.g.* $\alpha = 4$, $C = 10^8$ given in [Fleury 2009] and initial condition such that $C_1 = 2 \times 10^8$, $a = 0.3$, solving $R^\alpha - \frac{aRC_1}{2V} + C = 0$ gives several solutions among which only one or two are real and only one is real and lower than the maximum transmission range $R_{max} = 150$ of our nodes. Figure 4.6 plots this value of R with regards to the node speed. We can note the interesting behavior of R which is linear with regards to V. The faster the nodes, the bigger transmission range.

4.4 Experimental results

In this section we are going to present simulation results obtained using the models presented in previous sections and WSnet simulator [WSNet]. We use a model in which the range of transmission can be adapted according to the given values. In our simulations we use speed of the nodes, running the simulations for different scenarios with different speeds of nodes. Values for the range of transmission, period of HELLO messages, state of the nodes batteries, observed turnover and accuracy of obtained results are given as the functions of the nodes' speed. In these graphs we refer to the algorithms in the following way: Algorithm 6 is called *NoTAP* since it does not use turnover in calculations, Algorithm 4 is called *Fopt*, TAP algorithm [Ingelrest 2007] is referred as TAP and Algorithm 5 is called *Cost*

Simulation setup is made less realistic using ideal MAC layer in order to

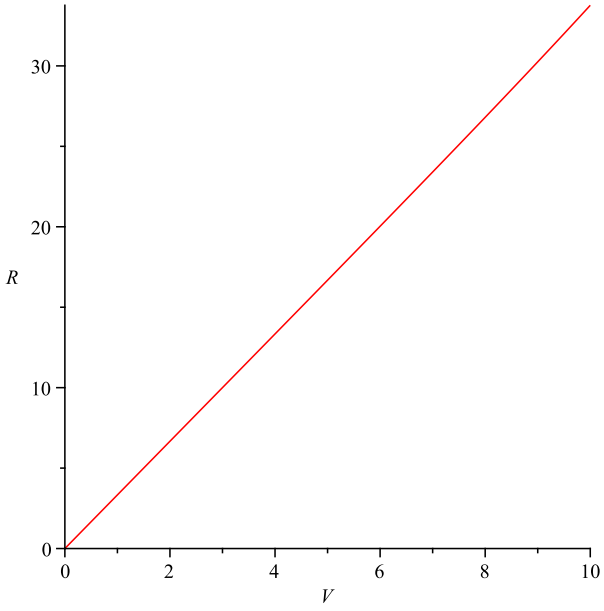


Figure 4.6: Values of R with regards to V when solving Eq. 4.26.

better observe and validate behavior of our algorithms. Simulation is run on the square area of 1000 × 1000m using 100 sensor nodes.

We will present results retrieved from the simulations with the comment on the performance of the TAP, Fopt and NoTAP algorithm. Separate discussion will be given on the obtained results with the Algorithm 5 for adaptation of the range using minimized energy consumption because results obtained using this algorithm are unrealistic (too good) and deserve additional explanation.

Transmission range as function of nodes' speed Figure 4.7 shows range adaptation as the function of speed of the nodes. We can notice that transmission range for the TAP algorithm is held on the same level, using the same approach (unit disk graph) as in [Ingelrest 2007]. In the case for Fopt and NoTAP algorithm ranges are increasing as the speed is increasing with the difference that NoTAP algorithm increases range linearly, in accordance with Figure 4.6. This behavior of these two algorithms is expected because when nodes are moving faster then they are also changing their neighbors faster so in order to maintain the number of new neighbors the algorithms are increasing the range. We have to note also that Fopt increases the range more than NoTAP since its range is not bounded to single value for a given speed as it is the case for NoTAP.

Period of HELLO messages as function of nodes' speed Figure 4.8 shows how the period of HELLO messages (and f_{HELLO}) is adapted for the

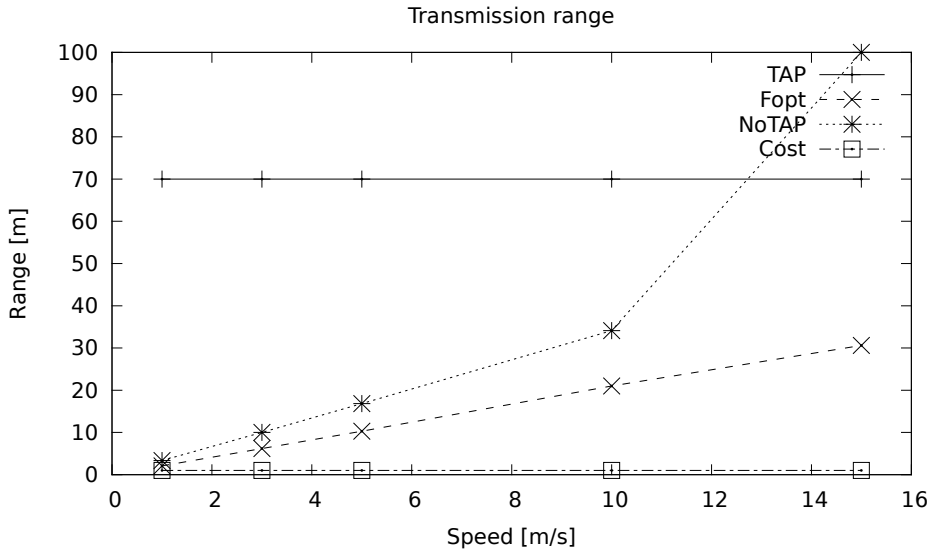


Figure 4.7: Transmission range as function of speed

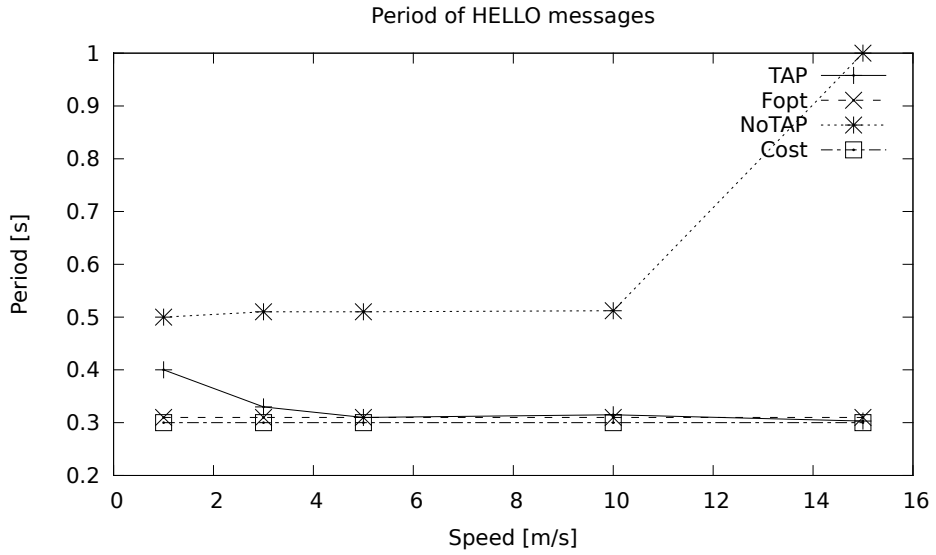


Figure 4.8: Period of HELLO messages as function of speed

different speed of the nodes. For the NoTAP algorithm period of HELLO messages is increasing following the results of the equation given in Algorithm 6 and Eq.4.28, TAP is decreasing HELLO period with the increase of speed because its range is fixed and the only way to adapt its behavior when nodes are moving faster is to lower the period (hence increase the frequency of HELLO messages). Fopt is keeping the period on almost constant value because the adaptation to the higher speed is done with an increase of the range.

Change of battery level as the function of speed In this case we have to point out that we used linear discharge model for the battery, decreasing certain amount energy from the battery multiplied by range of transmission each time when we transmit packet, constant value for each received packet and loss of energy in idle mode represented by constant value multiplied by the time spent in idle mode (the time between receptions or between reception and transmit and vice versa). This representation of battery is not the best one since it overstates the impact of transmission range which is in real case smaller and is given with the increase of power of transmission.

From Figure 4.9 we can see that all algorithms better balance energy than TAP which keeps transmission range on the same and due to this fact has worse results than others. We can also observe that energy loss is bigger as the speed of the nodes increases this is due to the adaptation of algorithms and their attempt to balance the values of turnover, frequency and range which compensates in higher energy loss.

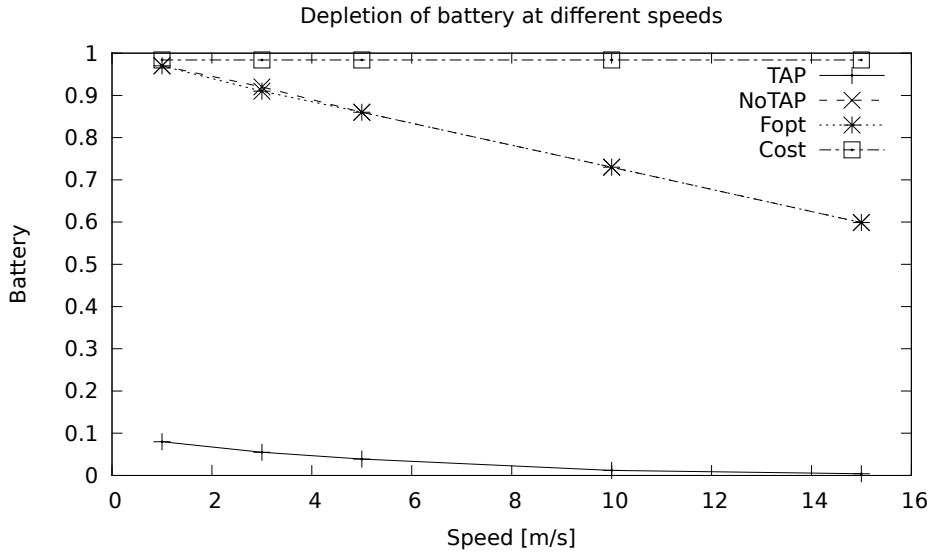


Figure 4.9: Battery level as function of speed

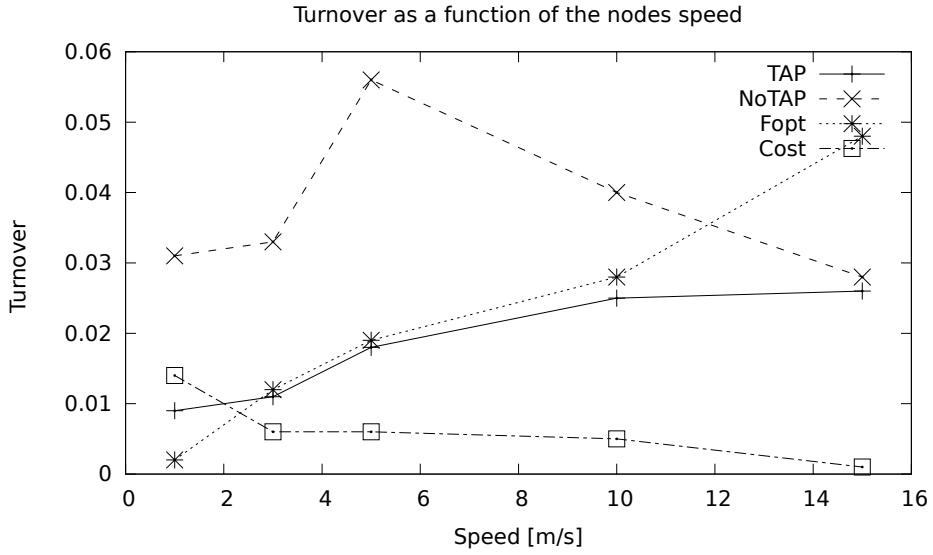


Figure 4.10: Turnover as function of speed

Turnover as a function of speed The results for these values, shown in Figure 4.10, are more straightforward, we can see that for the algorithms that take turnover (Fopt and TAP) into account there is an increase of turnover with the increase of speed while NoTAP is having changes of turnover not depending on the speed. Fopt has slightly bigger change of turnover as a result of adaptation both frequency and range.

Accuracy of the neighborhood table One of the important values to follow is the accuracy of the neighborhood tables, shown on the Figure 4.11. Accuracy is calculated checking the state of neighborhood table periodically and checking if all the nodes listed as the neighbors are still in the neighborhood of the node that is being checked. All algorithms are showing tendency of increasing accuracy with decrease of the speed which is logical since the faster nodes spend less time in physical neighborhood of their neighboring node.

Comment on the Cost algorithm Cost algorithm shows almost perfect results when taking into account state of the battery and accuracy of the neighborhood table, but looking into dependency of the turnover and range gave us the clue what is the reason for this. This algorithm tries to minimize the energy, and its doing it well, but at the cost of keeping the range of transmission on the lowest value thus gaining new neighbors occasionally and with big accuracy. Also since the transmission range is minimal it also preserves battery in the best way. What can we conclude from this is that the model that we imposed for this algorithm is too ideal and does not take into account the nuances this adaptation might have.

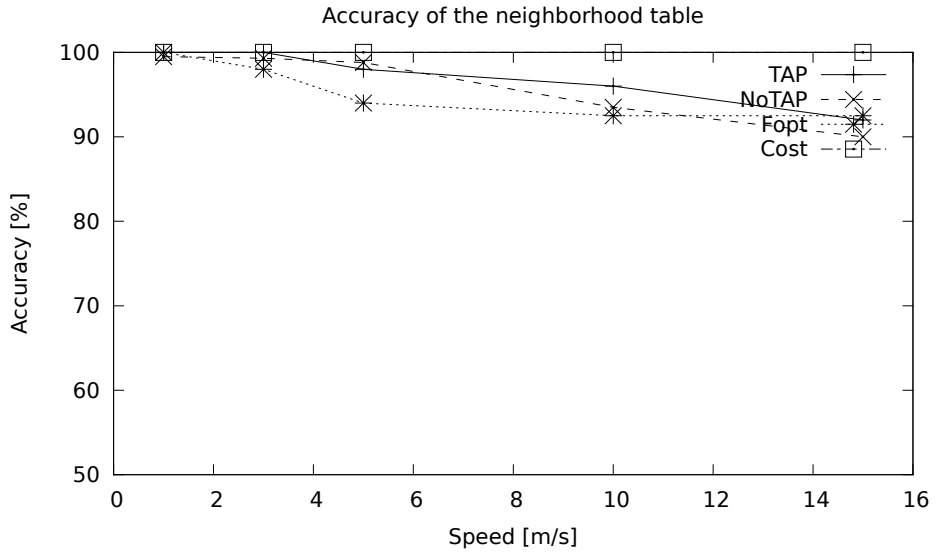


Figure 4.11: Accuracy of the neighborhood table as function of speed

4.5 Conclusion

To conclude this chapter, we can say that we have presented one approach to find energy efficient neighborhood discovery in the mobile wireless sensor networks. To the best of our knowledge this is the first attempt to apply this kind of approach which combines adaptation of frequency of HELLO messages and the range (power) of transmission. We present three intuitive algorithms which are balancing energy consumption of sent messages and the range at which they are sent. They all prove to be good solution for the given constraints except Cost algorithm for which these assumptions were not realistic thus it gave us results improbable in real situation.

Future work would include testing of these algorithms in more realistic environment with better given function between the range and power of transmission and energy cost. Also it would be nice to test these algorithms in real environment using mobile agents or sensors placed on the moving objects.

Chapter 5

Emulation of large scale wireless sensor networks

Contents

5.1 Introduction

Experimenting in a wireless environment is mandatory to validate some algorithms. Indeed, the behavior of real hardware and of the wireless medium might be so unexpected that plain simulation can not highlight these features. Nevertheless, experimentation in wireless sensor networks has always been the biggest issue. Problems with the experimentation in wireless sensor networks reflect through: *(i)* the cost of sensor units multiplied by the number of used sensor nodes – which has to be big in order to get accurate results comparable with the simulation; *(ii)* the cost in time needed to setup the whole network, and to facilitate – charge the sensors the sensors and update the program that they are running each time when we have changes in our code; *(iii)* collecting the results from the sensor network – this is usually being done in the place of experiment.

Existence of large scale experimentation testbeds like SensLAB [Senslab], Wisebed [WISEBED] or GreenOrbs [GreenOrbs] has simplified the way of obtaining the results in real environment using real sensor devices. Still these testbeds, although they are good tool for retrieving experimental results, have some limitations: usually they have fixed topologies, specific types of sensors are used in these sites so the user needs to get accustom himself to the limitations of the hardware that is being used and finally there is a cost of maintenance for these testbeds.

We propose usage of small wireless sensor networks, of up to 50 sensor nodes, which would be used to obtain the results and emulate the behavior of the large scale wireless sensor network. This small network is accompanied with appropriate graph generation tool which can generate different topologies and different emulation scenarios. In this way we can obtain scalable solution for experimentation, using small number of sensors easily allows us to follow the execution of our algorithms in large scale network.

5.2 Basic principle

Emulation is a process in which we use 1-hop environment to approximately calculate behavior of a large scale network instead of constructing large scale network itself. 1-hop environment is constructed in such a way that we have one source node S, currently holding the packet to be transmitted, located in the center of the coordinate system $(0, 0)$ with all other nodes, located in the circle of the radius R, within reach of central node S like depicted by Figure 5.1. Node D is a virtual destination node, assumed static, located on the distance potentially much bigger than the transmission range of S and thus requires multi-hop communications.

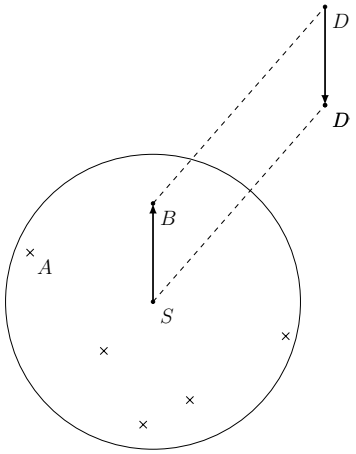


Figure 5.1: Calculation of the next step in 1-hop environment

Node S runs its routing algorithm to reach node D. Let us assume that node B is selected as next forwarder. In a complete experimentation, once B has received the packet sent by S, it runs the algorithm again to reach D. Since we have only a small set of sensors in 1-hop communication of S, B has no neighbor in forwarding direction of D and thus, it could not perform correctly its task. So, emulation we propose performs as follows. At each step, node S acts as the sending node. S takes the role of B. Since B has been

'moved' to S, destination D also needs to be 'moved' accordingly at each step. D is 'moved' through vector $\overrightarrow{SB}$ to position D' (see Figure 5.1).

This process of calculating new destination is repeated as long as the virtual destination is out of reach for source node S or that routing fails.

5.2.1 1-hop hexagonal grid

Sensors in our 1-hop environment are placed in a particular way to ensure that common neighborhood of nodes S and B is kept at the following step. Indeed, on Figure 5.1, node A is neighbor of both S and B. Once B is virtually relocated at node S position, node A should still exist in its neighborhood to better fit experimentation settings. To ensure such an imperative, nodes are placed over an hexagonal grid like shown on Figure 5.2.1, within the circle with radius R.

Placement of the nodes in the hexagonal grid ensures that in successive steps each node which is in the intersection of two successive 1-hop neighborhoods translates to one of the nodes of our grid (due to central and axial symmetry property of the hexagonal grid structure). For instance, on Figure 5.3, node 37 is chosen. Nodes 38, 13, 6 and 29 are common neighbors of S and 37 and thus should still be in the neighborhood of 37 at the following step. Thanks to central and axial symmetry property of the hexagonal grid structure, nodes 39, 15, 25 and 26 respectively can take their role at the next step.

This hexagonal grid consists of

- 6 nodes at distance a (nodes A to F on Figure 5.2.1),
- 6 nodes at distance $2a$ (nodes G to L on Figure 5.2.1) forming two regular hexagons,
- 6 nodes in the middle of each edge of the larger hexagon (distance $a\sqrt{3}$ from the source node) and
- 24 nodes in the center of each equilateral triangle which is formed by the 18 nodes already placed, among them
- 6 nodes are at distance $\frac{a\sqrt{3}}{3}$ from S, forming regular hexagon rotated for $30°$ counterclockwise comparing to the hexagon $A-F$
- 6 nodes are at distance $\frac{2a\sqrt{3}}{3}$ from S, which are in the centers of the equilateral triangles which are formed from two endpoints from hexagon $A-F$ and one endpoint from the middle of each edge of $G-L$ hexagon

- 12 nodes are at distance $\frac{a\sqrt{21}}{3}$ from S, formed with two endpoints from hexagon $G - L$ and nodes in the middle of its edges and one endpoint from the smaller hexagon $A - F$.

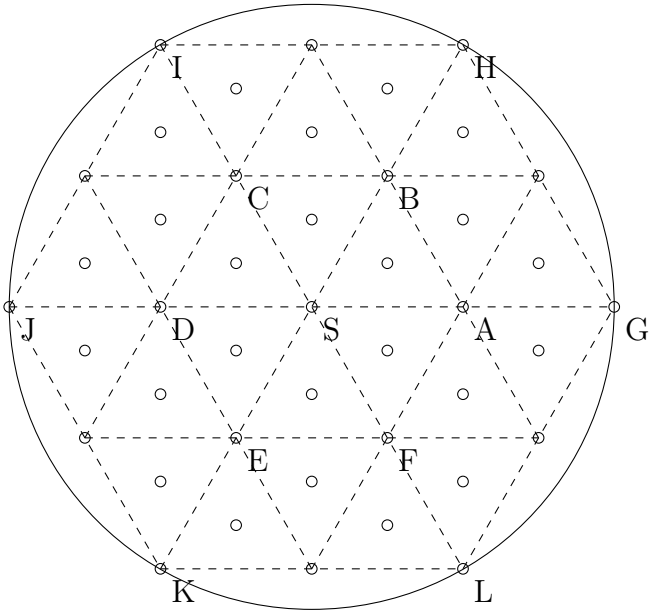
Nodes in this 1-hop neighborhood are numbered in an increasing order with the respect of the distance of the nodes from the source and the angle that they form with source and x-axis (for example: the closest neighbors are the nodes at distance $\frac{a\sqrt{3}}{3}$ with angles 30, 90, 150, 210, 270 and 330 degrees and they are numbered from 1 to 6 respectively with the increasing order of angle.

In this approach, if all nodes are used, obtained results would not fully correspond to realistic situations because every source node at every step sees the same neighborhood. In order to avoid this issue, only a subset of nodes are used, different at every step. More precisely, nodes in intersection of two successive steps (nodes that are in intersection of circles encircled in source node S and chosen neighbor with radius of 1-hop environment) and a random number of the rest of the nodes are used. In this way, a more realistic emulation is provided and the density of large scale network can be controlled.

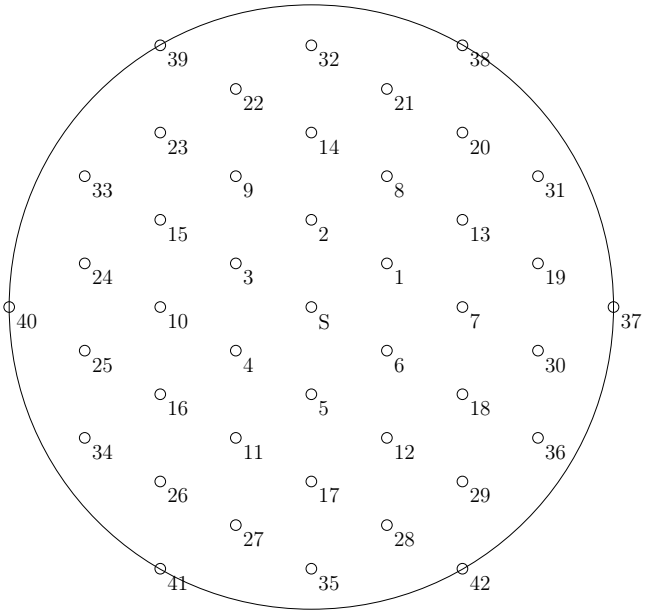
Example of emulation process using only small subset of nodes in the hexagonal grid is shown on Figure 5.3. Let us assume that the message to be transmitted is currently held by source sensor S and that routing algorithm gives as a next destination node 37 (Figure 5.3(a)). Node S has only 7 neighbors among the 42 nodes of the grid. Next destination 37 is translated to node S position together with all the common neighbors between S and 37. At the next step, node 37 becomes S so in order to keep placement of all of the nodes in the common area, they are translated along vector -$\overrightarrow{SD'}$, virtually moving the circle of radius R to 37. Due to the properties of symmetry of the hexagonal grid structure each node will be exactly mapped to one of the existing nodes (Figure 5.3(c)), in our example: $37 \mapsto S$, $13 \mapsto 15$, $6 \mapsto 25$, $29 \mapsto 26$, $38 \mapsto 39$ and $S \mapsto 40$. Now we have the nodes common with the previous hop and out of the rest we can choose (using appropriate algorithm) subset for of the nodes for the next step.

5.2.2 Emulation large scale graph

In order to unify our representation of the graph and to facilitate easier usage we have chosen specific representation of the graph that we are using in simulation, emulation and in the experiments. This kind of representation allows us to run routing algorithm both within 1-hop neighborhood but also not on a general graph. Graph is represented as an array of unique entries in the following form:

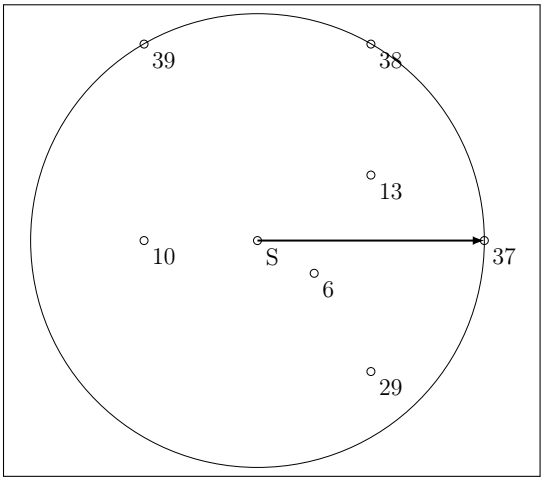


(a) Placement of the sensors

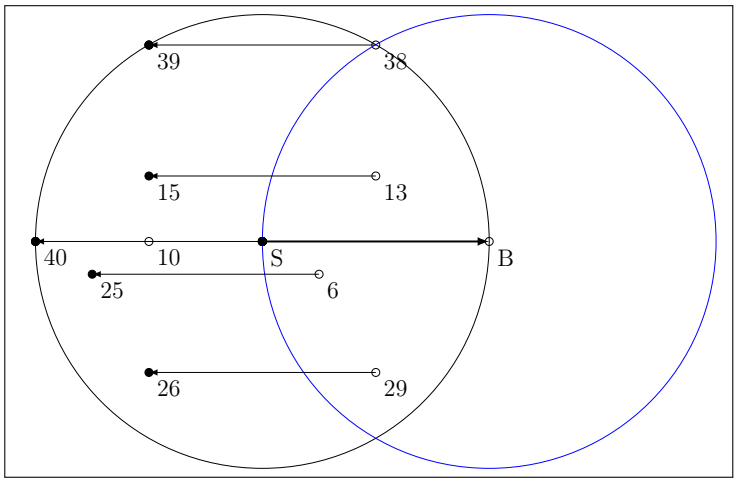


(b) Enumeration of the sensors

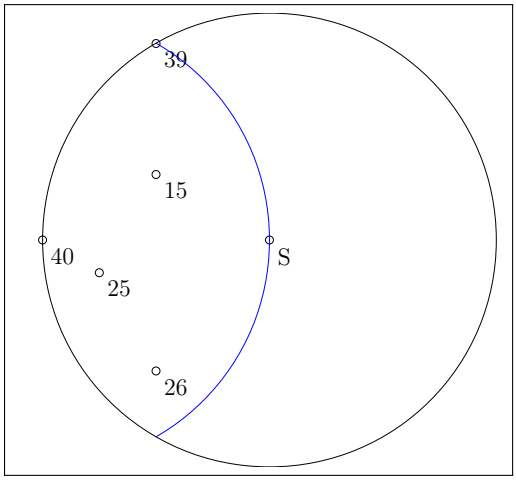
Figure 5.2: Hexagonal grid used for the placement of sensors.



(a) Choosing next destination



(b) Translation of existing nodes



(c) Mapping of the nodes from current step to the next step

Figure 5.3: Emulation steps using subset of sensor nodes in hexagonal grid.

$$node_{absID}(x_{absID}, y_{absID})$$

$$neighbor_{relativeID_1}(node_{absID})$$
$$neighbor_{relativeID_2}(node_{absID})$$
$$\vdots$$
$$neighbor_{relativeID_n}(node_{absID})$$

where $n \leqslant 42$.

In this representation each unique entry describes one node in the graph with its unique identification number ($absID$), its absolute position (x_{absID}, y_{absID}), followed by the list of neighbors (maximum number of 42) with corresponding relative identification number in the 1-hop neighborhood and their absolute ID. Graph has its proprietary format that we called graph description file (*.gdf) which has one unique entry for each generated node. Generated GDF file allows using the same graph during both simulation and emulation. This will be described in the following example.

In order to simulate generated topology, we just need a part of the information provided i.e. node absolute ID and x and y absolute position (geographical coordinates). List of the neighbors could be created by one successful HELLO message exchange. On the other side emulation takes use of the whole GDF file. Let us say that we want to emulate routing from node A to node B. Initially, source node (always the same central node in the case of emulation) should be provided with information about node's A (initial source) and node's B (destination) geographical positions and list of neighboring nodes (subset of maximal 42) that will be used in this step. Source node will find the next forwarder (one of the 1-hop neighbors) according to selected routing algorithm. After decision has been made, source node will search the GDF for the unique entry with same absolute ID as the selected forwarder. Once entry has been found, it is used as current central node in the following step, providing the new subset of neighboring nodes. This process will repeat until destination is reached or algorithm fails.

For both emulation and simulation we are using the graph which is built upon presented grid, and represented in graph description format, using the MIN-DPA algorithm [Onat 2008] which tends to distribute node degree more uniformly maintaining the connectivity. To make it suitable for our placement of the nodes we needed to adapt algorithm to the new hexagonal grid layout. Certain small changes had to be introduced: instead of randomly placing new nodes around node with smallest number of neighbors, its position is randomly chosen among one of the 42 predefined position in hexagonal grid

(cf. Figure 5.2.1). Other steps in MIN-DPA algorithm were kept the same. To control the generated graph density algorithm takes as input number of nodes that are to be generated and the surfaces of the area.

Why did we choose this type of graph for our simulations and emulation? Authors are claiming [Onat 2008] that graph has properties same to the ones we can find in real graphs, moreover it preserves so it was suitable for the range of routing algorithms that we were planing to test (mostly greedy-like algorithms). Here we can point out that basically each graph or the tool for graph generation can be used, the only thing which is important is to follow the hex-grid when generating graph. In this way it is ensured that we have graph which is suitable for our platform.

We also have to emphasize the fact that in this way we can generate different kinds of scenarios for the including the emulation of obstacles in network. In this case everything depends on the way we generate the graph, as long as it follows described representation it can be used in emulation and further on it all depends on the algorithm used for routing and if it can manage all the problems imposed to it.

5.3 Experimental results

In this section, we compare simulation and emulation results. More detailed, being in progress, we first decided to check our approaches on the simulator. Using the same generated graph we are running simple simulation without taking care of 1-hop neighborhood and simulation of emulation which is running with the constraint of 1-hop neighborhood.

5.3.1 Simulation setup

Simulations are run over the WSNet [WSNet] simulator, an event-driven simulator for large scale wireless networks. All simulated network topologies were generated as explained in previous section using MIN-DPA algorithm. They consist of roughly 1000 nodes deployed in the area of dimension 1000x1000 m.

Similarly to [Lukic 2009], each node performs an initialization phase where it broadcasts 1024 message, separated in 8 rounds, in order to evaluate link quality i.e. to measure ETX link metric. Since this phase directly impacts routing decision, we have compared in our simulations the influence of the choice of the underlying radio propagation model on the routing decision. For each of the simulated topologies we have run the initialization phase with 3 different propagation models: unit disk model, free space and Rayleigh fading. Results underline the importance of good choice of the underlying propagation

model on the simulation results of tested routing protocol based on link quality and the superiority of emulation as the most realistic one since is uses real nodes.

For each of the simulated topologies we run 4 routing algorithms - XTC, GARE, COP_GARE, as in our previous work [Lukic 2009], and LearnG. For each run we have 4 source nodes, situated in the 4 different corners of the network. The routing is performed across the diagonals to reach the destination situated in the opposite corner.

Apart from whole network simulation we perform the simulation of the emulation approach (step by step using GDF file as described before). Same simulation setup is used in this case.

5.3.2 Emulation setup

As mentioned earlier 1-hop neighborhood consists of 43 sensors, one central source node, placed in $(0, 0)$ and 42 neighbors placed in presented hexagonal grid. Source node holds the information about the large scale network that we want to emulate (GDF file) and possible starting points and destinations for routing algorithm (4 different corners of the large scale network). In our emulation, nodes of first hexagon (nodes A to F on Figure 5.2.1) are placed at distance $a = 2.5m$ from S.

Emulation of wireless sensor networks is executed in two steps. Since we aim for the realistic results and influence of realistic environment we are first running initialization phase in which each node is sending 8 rounds of 128 packets (altogether 1024 packets, like in simulations) and each of its neighbors is counting the total number of received messages. Each node in our 1-hop neighborhood is passing this phase while all the others are gathering statistics for currently active link (between the node that is sending messages currently and itself). In the end of this process source node S, placed in $(0, 0)$ gathers statistics for all 1-hop neighbors. At this point we have probability of reception for each pair of nodes, and we can use this data in latter experiments (since this data is held in FLASH memory of the source sensor).

1-hop neighborhood is now ready to run different routing algorithms in order to prove them or to test their reliability. In our setup we are running routing algorithms that use probability of the reception as one of the variables in weight function. Same algorithms, GARE, COP_GARE, XTC and LearnG are evaluated in this environment in order to compare results obtained from the simulation. Important parameters that we are following are: total number of hops needed to reach destination and total number of sent messages with retransmissions.

5.3.3 Comparison of the results

Algorithm that we used for generation of the graph generates connected graph with another constraint that we imposed on it, and that is that each node need to have at least two edges. Graph generated in this way favors greedy-like algorithms and they pass in whole 4 directions for all 4 corners of the generated network. Success rate for all algorithms was 100% but with different numbers of hops and retransmissions.

The results that we have obtained are similar to the ones presented in [Lukic 2009] both for the simulation on the graph and the simulation of the emulation. XTC algorithm is shown that has biggest number of hops per routing, in the range $680 - 710$ and the smallest number of retransmissions per route. The smallest number of hops had LearnG algorithm around $330 - 350$ which used the longest links, smaller differences are noticed in the performance of this algorithm. Since it favors longest links it also needed the biggest number of retransmissions but also in case of simple simulation we had slightly smaller number of hops per route which can be explained by the fact that in this case algorithm was not constrained to 1-hop neighborhood but it could also used the ones outside 1-hop neighborhood. Performance of the COP_GARE algorithm is in between LearnG and GARE algorithm again with higher number of retransmission. GARE performs using the medium edges (not the shortest like XTC and not longest like LearnG) giving around 440 hops per route.

We must also emphasize that real emulation using real sensor nodes was not performed in this scale due to the problems with constrained nature of sensor node. Memory space of the sensor node has shown as the biggest problem since the graph represented as an array with 1000 nodes with all additional data would take all the memory on the sensor node. We were not satisfied with possible size of graph of 150 nodes which can be put on the sensor node at maximum since it would give us the network which can be reached from end to end in few hops. Currently we are working on the solution that would include PC together with source node, we would use PC only to collect data from sensor node and to feed the sensor node with data for the routing, namely, whole graph structure would be placed in PC and the part needed for the computation of the next hop would be given to the source node when needed.

5.4 Conclusion

In this chapter we have presented our idea on building the scalable testbed containing small number of sensors to capture behavior of the large scale network. We have shown the consistency of this kind of structure running

the simulations with this structure and with regular simulation using the same graph we generated using MIN-DPA algorithm and representation of the graph that we proposed in previous sections. Running these simulations for greedy-like algorithms has shown that the structure we proposed can give us accurate results.

Next step will be the implementation of the given structure in hardware, using WSN430 sensor nodes. The first steps on the practical implementation have been already taken, implementation was done using 43 WSN430 sensor nodes but due to the limited resources on the sensor nodes we could not run the full size emulation experiment on it.

Future work would include both practical implementation of this structure as well as giving the new possibilities to it. At the moment we only used it for the emulation of routing greedy-like algorithms on one type of the graph. But it also can be used for the experimentation of other parts of protocol stack of wireless sensor network and also for different graphs. Then, we intend to dress a whole comparison between real experimentation and emulation results in order to show how close to experimentation the emulation is.

Chapter 6
Conclusion

Contents

In the beginning of this manuscript we gave a small discussion about models in general and we tried to explain our meaning of the word realistic and our approach to the problems in wireless sensor networks. Word realistic, and the approach as we defined it, was common point for the work which was spread over three big research areas in wireless sensor networks. What is missing the most in this work is this common point, but from the upper layer, and that would be the future work that we would like to see the most. Further on we will give brief overview of achieved results and possible advances.

6.1 Results

In our work, we addressed three different problems in wireless sensor networks: topology control, neighborhood discovery and emulation of large scale wireless sensor networks.

We presented topology control algorithm based on the relative neighborhood graph reduction which uses information gathered from the sensor hardware and reconfigure sensor network when some of its parts reaches the critical state characterized by low voltage level on their batteries. Compared to more typical RNG based solutions, this algorithm performs better maintaining connectivity in the network. This part of the work was practical using WSN430 nodes to retrieve information from its hardware and then the algorithm is tested and validated on the SensLAB platform.

Our contribution to the neighborhood discovery in mobile wireless sensor networks can be seen through the algorithms which we designed starting from the theoretical analysis of discovery of new neighbors and energy cost of the transmission. We have designed algorithms which take into account both energy and the rate of transmission and perform better than the algorithms

that do not handle both values. This part of the work was mostly done in the simulator.

Emulation of the large scale wireless sensor networks was a mix of two approaches. We have shown (in simulator) that the idea of using small wireless sensor networks to obtain behavior of large scale network is feasible. We have chosen specific layout and parameters for wireless sensor network nodes to assure reproducibility and accuracy of the results and easier handling more complex scenarios for the experimentation.

6.2 Future advances

With the presented ideas and the results there is a lot of possible directions for the future work. Our algorithm for the topology control can be extended in several ways. Just to give a few ideas of the possible improvements:

- additional weight function, or additional pass in the algorithm, which would ensure planarity of reduced graph,
- possible application of this algorithm, or this approach in general, to the mobile wireless sensor networks
- pairing this algorithm with appropriate algorithm for power transmission adaptation – this would ensure longer lifetime of a network,
- pairing this algorithm with some routing algorithm to evaluate this algorithm with real traffic.

Our neighborhood discovery algorithms with the adaptation of power of transmission could be improved taking into account more realistic model and different kind of adaptation functions. It would be interesting to see how these algorithms work in real hardware. Also, interesting idea would be to use approach of the sensor hints (used in our topology control algorithm) using the sensors which can give us information about the mobility of the sensor node, like GPS or accelerometer, to improve neighborhood discovery and adaptation of transmission range.

Emulation gives the most possibilities for the future work. It could be used as a tool for testing of different network protocols. Capabilities of such tool could be increased with:

- facilitating the experiments with different densities of the networks,
- emulation of interferences and obstacles in the network,
- 3D emulation,

- emulation of different parts of protocol stack other than routing.

Since we were talking a lot about common things for this work it would be interesting to see some of the ideas presented in this work functioning together on the same sensor device. It is obvious that emulation is a bit different because it is conceived with the idea to use it for testing but energy efficient neighborhood discovery working together with the topology control algorithm is a bit more realistic.

Bibliography

[Ahmad Kassem 2010] Ahmad Ahmad Kassem and Nathalie Mitton. *Adapting dynamically neighbourhood table entry lifetime in wireless sensor networks*. In Proc. 10th International Conference on Wireless Communications and Signal Processing, page 000, China, October 2010. 23, 45

[Bose 1999] Prosenjit Bose, Pat Morin, Ivan Stojmenovic and Jorge Urrutia. *Routing with guaranteed delivery in ad hoc wireless networks*. In DIAL-M, pages 48–55, 1999. 19

[Canonico 2007] Roberto Canonico, Pasquale Di Gennaro, Vittorio Manetti and Giorgio Ventre. *Virtualization Techniques in Network Emulation Systems*. In Euro-Par Workshops, pages 144–153, 2007. 24

[Cohen 2011] Reuven Cohen and B. Kapchits. *Continuous Neighbor Discovery in Asynchronous Sensor Networks*. Networking, IEEE/ACM Transactions on, vol. 19, no. 1, pages 69 –79, feb. 2011. 23

[Ephremides 2002] A. Ephremides. *Energy concerns in wireless networks*. Wireless Communications, IEEE, vol. 9, no. 4, pages 48 – 59, august 2002. 21

[Figuera 2009] C. Figuera, I. Mora-Jimenez, A. Guerrero-Curieses, J.L. Rojo-Alvarez, E. Everss, M. Wilby and J. Ramos-Lopez. *Nonparametric Model Comparison and Uncertainty Evaluation for Signal Strength Indoor Location*. Mobile Computing, IEEE Transactions on, vol. 8, no. 9, pages 1250 –1264, sept. 2009. 18

[Fleury 2009] Eric Fleury and David Simplot-Ryl, editeurs. Réseaux de capteurs. Hermes Science - Lavoisier, 2009. 16, 17, 20, 57, 61

[FreeRTOS] FreeRTOS. *A Free RTOS for Embedded Systems*. http://www.freertos.org. 12

[Gabriel 1969] Ruben K. Gabriel and Robert R. Sokal. *A New Statistical Approach to Geographic Variation Analysis*. Systematic Zoology, vol. 18, no. 3, pages 259–278, September 1969. 19

[Grau 2009] Andreas Grau, Klaus Herrmann and Kurt Rothermel. *Efficient and Scalable Network Emulation Using Adaptive Virtual Time*. International Conference on Computer Communications and Networks, vol. 0, pages 1–6, 2009. 24

[GreenOrbs] GreenOrbs. *A Long-Term Kilo-Scale Wireless Sensor Network System in the Forest.* 69

[Ingelrest 2007] François Ingelrest, Nathalie Mitton and David Simplot-Ryl. *A Turnover based Adaptive HELLO Protocol for Mobile Ad Hoc and Sensor Networks.* In MASCOTS, pages 9–14, Istanbul, Turkey, October 2007. IEEE Computer Society. 23, 45, 46, 47, 56, 58, 61, 63

[Javaid 2009] N. Javaid, A. Javaid, I. A. Khan and K. Djouani. *Performance study of ETX based wireless routing metrics.* In Proc. 2nd Int. Conf. Computer, Control and Communication IC4 2009, pages 1–7, 2009. 18

[Judd 2005] Glenn Judd and Peter Steenkiste. *Using Emulation to Understand and Improve Wireless Networks and Applications.* In NSDI, 2005. 24

[Ke 2000] Qifa Ke, David A. Maltz and David B. Johnson. *Emulation of Multi-Hop Wireless Ad Hoc Networks.* In Proceedings of the Seventh International Workshop on Mobile Multimedia Communications (MOMUC 2000), IEEE Communications Society, October 2000. 24

[Khadar 2009] F. Khadar and D. Simplot-Ryl. *From theory to practice: topology control in wireless sensor networks.* In MobiHoc'09, pages 347–348, 2009. 20, 37, 39

[Labrador 2009] Miguel A. Labrador and Pedro M. Wightman. Topology control in wireless sensor networks: with a companion simulation tool for teaching and research. Springer, 2009. 17

[Li 2003] Ning Li, Jennifer C. Hou and Lui Sha. *Design and Analysis of an MST-Based Topology Control Algorithm.* In INFOCOM, 2003. 19

[Li 2011] Xu Li, Nathalie Mitton and David Simplot-Ryl. *Mobility Prediction Based Neighborhood Discovery in Mobile Ad Hoc Networks.* In Networking (1), pages 241–253, 2011. 23

[Long 2009] Hengyu Long, Yongpan Liu, Yiqun Wang, Robert P. Dick and Huazhong Yang. *Battery allocation for wireless sensor network lifetime maximization under cost constraints.* In Proceedings of the 2009 International Conference on Computer-Aided Design, ICCAD '09, pages 705–712, New York, NY, USA, 2009. ACM. 21

[Loscrì 2010] Valeria Loscrì, Enrico Natalizio and Carmelo Costanzo. *Simulations of the Impact of Controlled Mobility for Routing Protocols.* EURASIP J. Wireless Comm. and Networking, 2010. 22, 45

[L.Sichitiu 2005] Mihail L.Sichitiu and Rudra Dutta. *Benefits of Multiple Battery Levels for the Lifetime of Large Wireless Sensor Networks*. In Raouf Boutaba, Kevin Almeroth, Ramon Puigjaner, Sherman Shen and James P. Black, editeurs, NETWORKING 2005, volume 3462 of *Lecture Notes in Computer Science*, pages 403–408. Springer Berlin / Heidelberg, 2005. 21

[Lukic 2009] Milan Lukic, Bogdan Pavkovic, Nathalie Mitton and Ivan Stojmenovic. *Greedy Geographic Routing Algorithms in Real Environment*. In MSN, pages 86–93, 2009. 76, 77, 78

[Maier 2007] Steffen Maier, Daniel Herrscher and Kurt Rothermel. *Experiences with node virtualization for scalable network emulation*. Computer Communications, vol. 30, no. 5, pages 943 – 956, 2007. Advances in Computer Communications Networks. 24

[Mališić 1989] Jovan Mališić. Slučajni procesi: teorija i primene. Gradjevinska knjiga, 1989. in serbian. 47

[McGlynn 2001] Michael J. McGlynn and Steven A. Borbash. *Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks*. In Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing, MobiHoc '01, pages 137–145, New York, NY, USA, 2001. ACM. 23

[Moy 1994] J. Moy. *OSPF - Open Shortest Path First*, March 1994. 22

[Onat 2008] Furuzan Atay Onat, Ivan Stojmenovic and Halim Yanikomeroglu. *Generating random graphs for the simulation of wireless ad hoc, actuator, sensor, and internet networks*. Pervasive and Mobile Computing, vol. 4, no. 5, pages 597–615, 2008. 75, 76

[Prim 1957] R. C. Prim. *Shortest connection networks and some generalizations*. Bell System Technology Journal, vol. 36, pages 1389–1401, 1957. 19

[Ravindranath 2010] Lenin S. Ravindranath, Calvin Newport, Hari Balakrishnan and Samuel Madden. *"Extra-Sensory Perception" for Wireless Networks* . In HotNets-IX, Monterey, CA, October 2010. 21, 27

[Ravindranath 2011] Lenin S. Ravindranath, Calvin Newport, Hari Balakrishnan and Samuel Madden. *Improving Wireless Network Performance Using Sensor Hints* . In 8th USENIX Symp. on Networked Systems Design and Implementation (NSDI), Boston, MA, March 2011. 27

[Santi 2005] Paolo Santi. Topology control in wireless ad hoc and sensor networks. Wiley, 2005. 16, 17, 20

[Senslab] Senslab. *Very large scale open wireless sensor network testbed.* http://www.senslab.info/. 28, 69

[Sobeih 2006] Ahmed Sobeih, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Honghai Zhang, Wei-Peng Chen, Hung-Ying Tyan and Hyuk Lim. *J-Sim: a simulation and emulation environment for wireless sensor networks.* Wireless Communications, IEEE, vol. 13, no. 4, pages 104 –119, aug. 2006. 24

[Srinivasan 2006] Kannan Srinivasan and Philip Levis. *RSSI is Under Appreciated.* In In Proceedings of the Third Workshop on Embedded Networked Sensors (EmNets, 2006. 18

[Supowit 1983] Kenneth J. Supowit. *The Relative Neighborhood Graph, with an Application to Minimum Spanning Trees.* J. ACM, vol. 30, no. 3, pages 428–448, 1983. 19

[Toussaint 1980] Godfried T. Toussaint. *The relative neighbourhood graph of a finite planar set.* Pattern Recognition, vol. 12, no. 4, pages 261–268, 1980. 19

[Troel 2004] Arnaud Troel. *Prise en compte de la mobilité dans les interactions de proximité entre terminaux à profils hétèrogènes.* PhD thesis, Universite de Rennes, France, 2004. In French. 22, 23, 46, 58

[Tsiftes 2011] Nicolas Tsiftes and Adam Dunkels. *A database in every sensor.* In Proceedings of the ACM Conference on Networked Embedded Sensor Systems, ACM SenSys, Seattle, USA, 2011. 12

[Wang 2009] Qingsi Wang, Xinbing Wang and Xiaojun Lin. *Mobility increases the connectivity of K-hop clustered wireless networks.* In MOBICOM, pages 121–132, 2009. 45

[WASP] WASP. *Wirelessly Accessible Sensor Populations.* http://wasp-project.org/. 22

[Wattenhofer 2004] R. Wattenhofer and A. Zollinger. *XTC: a Practical Topology Control Algorithm for Ad hoc Networks.* In Proc. 4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN), Santa Fe, MN, USA, 2004. 20

[Wightman 2008] P. M. Wightman and M. A. Labrador. *A3: A Topology Construction Algorithm for Wireless Sensor Networks*. In Proc. IEEE Global Telecommunications Conf. IEEE GLOBECOM 2008, pages 1–6, 2008. 17

[WISEBED] WISEBED. *Wireless Sensor Network Testbeds*. http://www.wisebed.eu/. 69

[WSN430] WSN430. *Kit Developer's Guide*. http://perso.ens-lyon.fr/eric.fleury/Upload/wsn430-docbook/. 5, 13, 29

[WSNet] WSNet. *An Event-driven Simulator for Large Scale Wireless Networks*. http://wsnet.gforge.inria.fr/. 61, 76