# THÈSE

PRÉSENTÉE À

## L'UNIVERSITÉ PIERRE ET MARIE CURIE

ÉCOLE DOCTORALE INFORMATIQUE,
TÉLÉCOMMUNICATIONS ET ÉLECTRONIQUE

Par **Bramas Quentin**

POUR OBTENIR LE GRADE DE

## DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

---

# Réseaux de Capteurs sans Fil Efficaces en Énergie

---

Thèse dirigée par Sébastien Tixeuil préparée au Laboratoire d'Informatique de Paris 6 (LIP6), équipe NPA, **soutenue publiquement le 4 Octobre 2016,**

**après avis des rapporteurs :**

| | | |
|---|---|---|
| Nicola Santoro | – | Professeur, Carleton University |
| Nathalie Mitton | – | Chargée de Recherche, INRIA Lille |

**et devant le jury composé de :**

| | | | |
|---|---|---|---|
| *Rapporteur* | Nicola Santoro | – | Professeur, Carleton University |
| *Rapporteur* | Nathalie Mitton | – | Chargée de Recherche, INRIA Lille |
| *Examinateur* | Arnaud Casteigts | – | Maitre de conférence, LABRI Bordeaux |
| *Examinateur* | Franck Petit | – | Professeur, UPMC |
| *Directeur* | Sébastien Tixeuil | – | Professeur, UPMC |

– 2016 –

# Energy-Centric Wireless Sensor Networks

Quentin Bramas

in fulfillment of the requirements for the degree of
Doctor in the subject of
Computer Science

October, 4th, 2016

| | | | |
|---|---|---|---|
| Referees | Nicola Santoro | Professor - Carleton University | |
| | Nathalie Mitton | Researcher - INRIA Lille | |
| Committee | Nicola Santoro | Professor - Carleton University | Referee |
| | Nathalie Mitton | Researcher - INRIA Lille | Referee |
| | Sébastien Tixeuil | Professor - UPMC | Advisor |
| | Arnaud Casteigts | Associate professor, LABRI Bordeaux | Referee |
| | Franck Petit | Professeur, UPMC | Referee |

# Résumé

Les réseaux de capteurs sans fil sont constitués de noeuds capteurs, capables de récolter des données, de les analyser et de les transmettre. Ces réseaux ont plusieurs applications, en fonction de la zone où ils sont déployés. Application militaire ou de sauvetage dans des zones pouvant être inaccessibles aux humains ; application sanitaire avec des capteurs déployés sur et dans le corps humain ; application de surveillance avec des capteurs sur les voitures d'un ville, ou les arbres d'une forêt. Les noeuds sont autonomes en énergie et il est primordial d'assurer leur longévité sans retarder la récolte des données. La tache principale réalisée par les réseaux de capteurs sans fils consiste à effectuer des mesures et à envoyer ces données jusqu'à un noeud coordinateur. Cette tache d'agrégation est effectuée régulièrement, ce qui en fait la plus consommatrice d'énergie. L'étude approfondie de la consommation d'énergie des capteurs, qui au centre de ma thèse, peut se traduire de différentes manières.

Premièrement, nous avons étudié la complexité du problème de l'agrégation de données en utilisant un modèle simplifié pour représenter un réseau de capteurs sans fils. Dans ce modèle, nous avons montré que la recherche d'une solution optimale permettant d'agréger les données d'un réseau de capteurs sans fil quelconque n'était pas réalisable en pratique, même pour un algorithme centralisé connaissant l'évolution futur du réseaux. De plus nous avons étudié la résolution de ce problème par un algorithme distribué fonctionnant en temps réel. Nous avons montré que le problème n'avait pas de solution en général, sans connaissance supplémentaires.

Secondement, nous nous sommes concentrés sur l'estimation de cette durée de vie. Les simulateurs existant implémentent souvent des modèles trop simplistes de consommation et de batterie. Par ailleurs, les modèles plus réalistes de batterie, implémentés dans des simulateurs généralistes, sont trop complexes pour être utilisés pour les réseaux de capteurs possédant de nombreux noeuds, et dont la durée à simuler peut atteindre des mois, voire des années. WiSeBat est un modèle de batterie et de consommation d'énergie optimisé pour les réseaux de capteurs, implémenté dans le simulateur WSNET. Après validation, nous l'avons utilisé pour comparer les performances des algorithmes de broadcast efficaces en énergie.

# Abstract

A wireless sensor network is an ad-hoc network connecting small devices equipped with sensors. Such networks are self-organized and independent of any infrastructure. The deployment of a WSN is possible in areas inaccessible to humans, or for applications with a long lifetime requirement. Indeed, devices in a wireless sensor network are usually battery-powered, tolerate failure, and may use their own communication protocols, allowing them to optimize the energy consumption. The main application of WSNs it to sense the environment at different locations and aggregate all the data to a specific node that logs it and can send alerts if necessary. This task of data aggregation is performed regularly, making it the most energy consuming. As reducing the energy consumed by sensor is the leading challenge to ensure sustainable applications, we tackle in this thesis the problem of aggregating efficiently the data of the network. Then, we study lifetime evaluation techniques and apply it to benchmark existing energy-centric protocols.

# Acknowledgments

This PhD thesis wrap up my schooling so there are many people I want to thank for all their support in this long path. But, more simply, it represents the work I have done in the last three years. That is why, I first want to thank the referees Nicola Santoro and Nathalie Mitton for accepting to review my thesis and giving detailed feedback about it. I also want to thank the other members of the committee, Franck Petit and Arnaud Casteigts for their participation, not only for the defense but, through the thesis. I thank my supervisor Sébastien Tixeuil for his continuous support, his motivation and all the precious advises he gave me. I also thank Maria Potop-Butucaru who welcomed me when I arrived in the laboratory and was always present during the thesis. The SMART-BAN project was a great opportunity to work with people in different domains and resulted in interesting collaborations, especially with Julien Sarrazin, Solofo Razafimahatratra and Wilfried Dron.

I thank all the NPA team for everything. The permanent researchers such as Promethee Spathis, Marcelo Dias de Amorim and Lelia Blin, and all the Phd students, Amr Abdelfattah, Mustapha Abuteir Rabee, Salah Belouanas, Fadwa Boubekeur, Florent Coriat, Antonella Del Pozzo, Grassi Giulio, Alexandre Maurer, Narcisse Nya, Filippo Rebecchi, Matteo Sammarco and Badreddine Wafa, and especially Benjamin Baron and Alexandre Ragaleux, who always answered my questions about networking. All our discussions helped me to write better papers, to find new research paths and even to get through proofs I was struggling with for weeks.

I would like to thank all the other people I met during my thesis. I really enjoyed all the discussions with Thanh Dang Nguyen, and I thank him for the great moments we spend. This collaboration continued with Xavier Defago and François Bonnet, who made me discover their work and filled my mind with a lots of interesting research paths. The Chapter 6 is the result of a collaboration with Toshimitsu Masuzawa. It was an honor and a pleasure to work with him. I also thank the PhD students I meet in the AlgoTel conferences for all the interesting discussions we had, especially Noël Gillet. I also thank Jérémy Esteves for the proofreading.

Finally, since this thesis is also the end of my schooling, I would like to thank my family for their support. Especially my parents who have always encouraged and motivated me, and my brothers who supported me. I also want to thank my parents in law who encouraged me. I thank my wife and my daughter for their presence, their support and their smile.

*To my daughter and my wife.*

# Contents

# List of Figures

# List of Tables

Chapter 1

# Introduction

## Contents



In the last decade, the number of personal devices connected to the Internet has grown to exceed the human population on earth. The decreasing cost of small devices has driven the development of the Internet of things, which consists in connecting devices, such as printers, fridges, or lamps, *directly to the Internet*. Their connectivity allows users to send requests from their smartphones, *e.g.* to turn the heat on before getting back home, to ensure the lights are off, or to check whether the door is locked.

From this abundance of devices, another paradigm has gain popularity: connecting devices together to form an independent network connected to the Internet through a unique device, called the gateway. In this context devices can communicate together with lightweight protocols (instead of the legacy Internet protocol) and it is still possible to communicate with them through the gateway. In return, there is no infrastructure that organizes their communication. This kind of network have lots of advantages: their low computational needs allow them to be battery powered; devices are small and relatively inexpensive; they can be used in areas where no internet connection or other infrastructure networks exist.

Usually, devices that compose such networks consist of a microcontroller, a battery, a radio transceiver and a sensor. A device uses its sensor (*e.g.* an accelerometer, a gyroscope, or a temperature sensor), and communicates with the other devices through wireless transmissions. The networked connection of these devices form a *Wireless Sensor Network*.

The health monitoring of a patient is a particularly challenging application of wireless sensor networks. Numerous examples of diseases would benefit from continuous or prolonged monitoring, such as cardiovascular disease, diabetes, hypertension, asthma, renal failure, etc. Monitoring is usually used post-operative, for systematic prevention (*e.g.* to prevent the sudden infant death syndrome), or to enhance the quality of life (*e.g.* a pump administering the correct dose of insulin to diabetics based on the glucose level measurements).

To perform the monitoring, intelligent physiological sensors can be integrated into a wearable network, called Body Area Network (see Figure 1.1). All the data from all the sensors are aggregated periodically to the gateway (*e.g.* a smartphone).

The gateway analyzes the aggregated data and logs it, or sends an alert to the hospital if there is a risk for the patient. In this context, the network must be reliable, sustainable and predictable. At the same time, devices should be small, battery powered (for some devices to be implanted inside the body), and having a lifetime of months or even years without intervention.

Despite the short distance between sensor nodes, communications are mutli-hop. In case of short-range and high-throughput wireless technologies (*e.g.* 60 GHz transmissions), signals travel solely by line-of-sight, and are blocked by the body. Hence, the mobility of the sensor nodes on the body may disconnect and connect nodes unexpectedly, preventing the use of the direct path between the source of a transmission and the gateway.

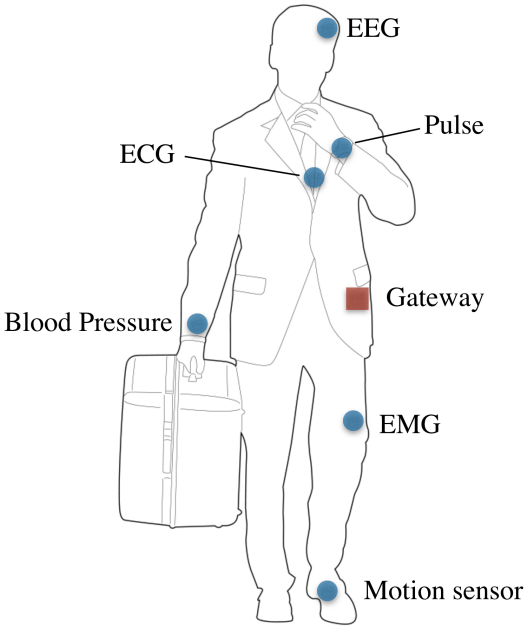



Figure 1.1 – A body area network.

A key challenge is to evaluate the lifetime of a wireless sensor application aiming to run a long period of time. Ensuring that the application runs at least for the time given by the specification is crucial and may have a direct impact on the life of the patient. In this thesis, we tackle this problem by providing accurate models to evaluate the lifetime of the network using simulations.

The fundamental task of data aggregation is done regularly, making it the most energy-consuming. In this thesis, we analyze this task from a theoretical point of view. We assume that the minimal amount of transmissions is used (to optimize the energy consumption) and try to limit the duration of the data aggregation. Indeed, it is important to have a low delay between the detection of a problem by a device and its reception by the gateway.

## 1.1 Context of the Thesis

**Wireless Sensor Networks** A wireless sensor network paradigm was defined, as we know it now, between the years 1990 and 2000 [Bam98, GW02], in the prolongation of radio and wireless networks. It consists of devices that communicate together by mean of wireless transmissions. The devices are autonomous and there is no centralized infrastructure, which makes the network self-organized. When a device wants to communicate, it can only transmit a message to nearby devices. When the destination is far from the source, intermediate devices are induced to

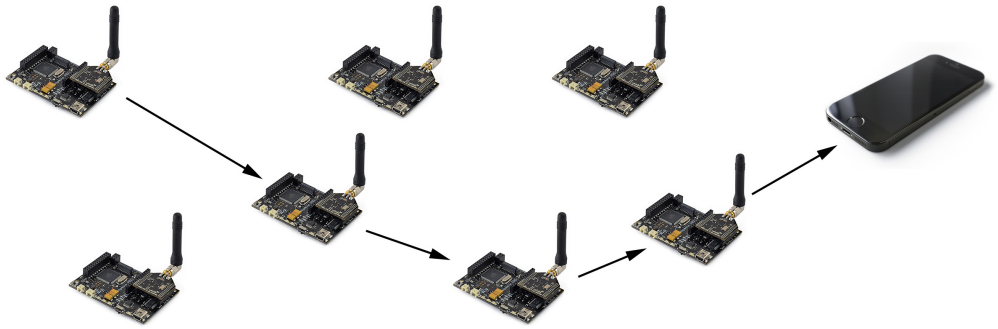


Figure 1.2 – A wireless sensor network connected to a smart-phone

forward the message, creating multi-hop communications. We present a typical architecture of a wireless sensor network in Figure 1.2. It consists of devices that cooperate to send data to a gateway, which in turn may be part of a bigger network.

**Performance Analysis** A key characteristic of wireless sensor networks is the shared medium used to propagate transmission signals. Simple models for the signal propagation lead to a representation of a wireless sensor network as a graph, which represents the devices and their communication links. When devices are mobile, the changes in the topology over the time are captured by a *Time-Varying Graph*, or simply by a sequence of graphs, that represents the evolution of the network over the time. For a more accurate estimation of the performance of a protocol, it is important to simulate entirely the way devices work and how the signals propagate and interfere using realistic signal propagation model.

**Energy Consumption** One fundamental challenge in a wireless sensor network is the limited amount of energy available to the devices. The energy is not only used by a device to perform its own actions, but is also used for unpredictable actions. This includes forwarding messages between two devices that cannot communicate directly, making sure that the device is still part of the network by sending control messages, informing incoming devices about the network, etc. Each decision made by the layers of the communication stack may impact the energy consumption of devices. That is why, optimizing and evaluating the energy consumption of devices in a wireless sensor network is the common thread of this thesis.

## 1.2 Thesis Organization

**Part One: Context** In this first part, we introduce the wireless sensor networks, their characteristics, and their applications. We give an overview of their components and highlight several challenges that partly motivate this thesis. Then, we present the model we use to represent wireless sensor networks with different levels

of abstraction.

**Part Two: Data Aggregation Problem** The second part of this thesis extends results from [BT15, BMT16]. We consider the problem of data aggregation. First [BT15], we investigate, from a centralized point of view, the complexity of finding the optimal solution in static and dynamic wireless sensor networks. We also present the first approximation algorithm for the problem in a dynamic network. Then [BMT16], we investigate the problem from a distributed and online point of view. We show several impossibility results and present optimal algorithms when the interactions that occur in the networks are random.

**Part Three: Lifetime Estimation of a Wireless Sensor Network** The last part of this thesis is related to results presented in [BDBF+15, BT16]. We focus on benchmarking energy-centric protocols in wireless sensor networks. First [BDBF+15], we present a model for the energy consumption and the battery of a sensor node, called WiSeBat. We evaluate its implementation in the WSNet simulator [FCF07] against real devices and show that it outperforms the default energy model of the simulator. Then [BT16], we use this model to benchmark several energy-centric broadcasting protocols and show that their performance in a realistic environment differs from the original results.

**Reading map** Figure 1.3 summarizes dependencies between chapters of this thesis.

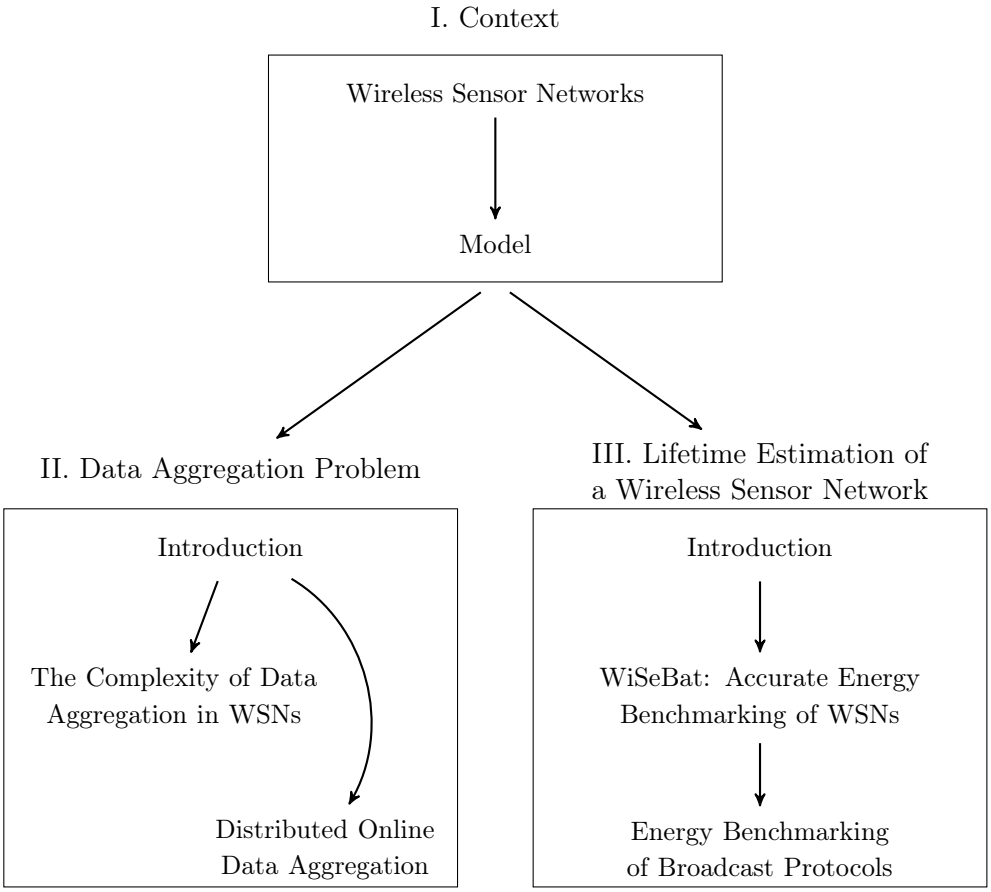



Figure 1.3 – Dependencies between chapters of this thesis.

# Part I

# Context

Chapter 2

# Wireless Sensor Networks

In a wireless sensor network, mWO¶qp§n!rf is important, but avoiding interference is more important.

**Contents**



The growing number of sensor nodes with sensing, computing and communication capabilities, was made possible by recent technological advances. This growth was encouraged by a variety of applications and contributes to the widespread interest in practical and theoretical aspects of wireless sensor networks. Sensor nodes should be inexpensive, small and sustainable in order to be easily deployed in a dangerous area, inside a human body or in vehicles, generally for monitoring applications. Miniaturization and cost reduction permit, for a single application, the deployment of multiple sensor nodes in an area. One particularity of those sensor nodes, is that they do not rely on an infrastructure. Each node is equipped with a radio transceiver that enables the communication with nearby nodes. The network composed by those sensor nodes is called a Wireless Sensor Network (WSN). A WSN is created on the fly, without infrastructure. In that sense, it is an ad hoc network.

By nature, ad hoc networks such as WSNs are self-organized, tolerate unpredictable behavior and can be dynamic. This raises various challenges, such as energy (sensors are battery powered) and delay efficiency (information is relevant for a short period of time only). The delay to transmit a message from one node to another can easily grow to become a problem. This is due to the multi-hop transmissions *i.e.*, when a node wants to transmit a message to a node that is not in its communication range, it has to use intermediate nodes to forward the message between sensors that are close to one another toward the destination. The choice of the intermediate node and the way messages are forwarded may impact the energy consumption of nodes and thus the lifetime of the network.

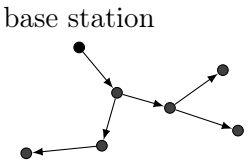



Figure 2.1 – A broadcast to the base station

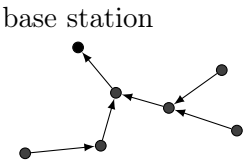



Figure 2.2 – A convergecast to the base station

The two fundamental tasks of a WSN are monitoring and tracking [YMG08]. The data obtained by each sensor node is then transmitted to a base station that analyzes the data. The base station is either controlled by an end-user or is used as a gateway between the WSN and another network such as the Internet. Those tasks usually require two kinds of transmission scheme, a transmission from the sensor nodes to the base station, and a transmission from the base station to the sensor nodes. The former one is used to retrieve the data obtained by sensor nodes to the base station for further analysis, and the latter one can be used to send application updates, queries, configuration, or control messages. We say we perform a *broadcast* when the base-station sends a message to every node in the network (Figure 2.1). We say we perform a *convergecast* when every node in the network sends a message to the base-station (Figure 2.2).

Another important feature of WSNs is that communications are performed through a shared medium and in all directions (we assume here that antennas are omnidirectional). This can be used to reduce the number of transmissions by broadcasting a message to all the neighbors at once. However, it can also lead to problems as simultaneous transmissions generate interference that can prevent the correct reception of messages. Indeed, in a real environment, any wireless signal is subject to several phenomena, such as attenuation over the distance, and the superposition with other wireless signals, before being received by a receiver. To be properly received, the signal corresponding to the message must be decoded considering the sum of all other incoming signals as noise.

**Types of Sensor Networks** Sensor nodes can be deployed in various types of environment. The simplest deployment concerns terrestrial WSNs where a number of nodes are statically deployed in an area, randomly located, or with predefined position such as on a grid. Terrestrial WSNs usually offer good conditions for the radio transmissions, and allow energy harvesting with solar panels for instance. Underground or underwater WSNs [PMA06] require appropriate equipment and are more expensive to deploy and maintain. Also, the topology may be different than simple terrestrial WSNs due to the possible placement in three dimensions. Mobile WSNs (or MANETs for Mobile ad hoc networks) consist of a collection of moving nodes, which creates a topology that changes over the time. The mobility of the nodes can be active (nodes can change their position *e.g.* with motorized wheels) or passive (nodes undergo mobility *e.g.* when located on animals or on vehicles).

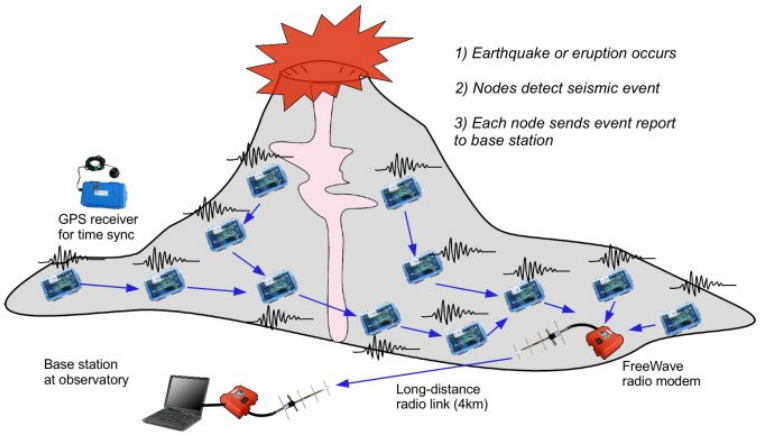



Figure 2.3 – Monitoring of a volcano [WALR+06]. (c) Harvard University, 2006

Finally, wireless body area networks consist of nodes located on or in the body. This type of WSN is particularly challenging as nodes are moving on a three dimensional space and the body is an heterogeneous environment extremely complex to model [iNWK+15, PMS+15].

## 2.1 Applications

As explained by J.Yick *et al.* [YMG08], WSN applications can be classified in two categories: monitoring and tracking.

**Monitoring Applications** Sensor nodes can be used to monitor one or several types of data in a specific environment, in order to alert the base station when something is wrong or just to study the evolution of a feature over the time.

Health monitoring applications [BAB+07] gains a lot of attention and focuses on all the challenges as nodes have to be reliable, sustainable, and with nodes located on or in the body, it has to be safe. Moreover, since the data may be sensitive and personal, it has to be secured. The objectives includes post-operative or intensive care, or long-term surveillance of chronically ill patients or the elderly

Smart cities consist in the deployment of heterogeneous sensor nodes with different monitoring capabilities. The objectives include traffic and parking management, garbage level, noise and atmospheric pollution, crossroads traffic light regulation, smart lights, and finally, buildings, bridges and roads structural monitoring.

On Volcàn Reventador in northern Ecuador, sensor nodes equipped with seismographers and microphones monitor the volcano and alert the base station in case of a volcanic event [WALR+06] (see Figure 2.3).

In the *Macroscope of redwood* case study [TPS+05], sensor nodes were deployed on redwood trees in California, and at different heights, to monitor the air temperature, the relative humidity and the photo-synthetically-active solar radiation. The

data are used by Plant biologist to validate biological theories.

**Tracking Applications** Sensor nodes can be deployed on animals to track their movements over a period of time. For instance, *ZebraNet* [ZSLM04] system deployed several sensor nodes (with GPS unit) into zebra's collar in order to track their movement over several weeks. The positions log is sent multi-hop across zebras to the base station.

There exists a number of dataset available online containing GPS mobility data recorded by sensor nodes deployed on humans (*e.g.* contributed by I.Rhee *et al.* [RSH+09]) or on vehicles (*e.g.* taxi cabs in San Francisco [PSDG09] or in Roma [BBL+14]). While those data were obtained by sensor nodes that do not communicate, they can be used to simulate WSNs having the same mobility as the one recorded. For instance, what would be the delay to broadcast a message to all the taxi cabs in Roma, using only the multi-hop communication across them?

Human interactions can also be recorded using WSNs. For instance if sensor nodes, sending beacons at regular intervals, are deployed in a group of people, each node can record the received beacon (*i.e.*, its neighbors) over the time. Again, those data can be used to propose efficient routing protocols that may be used in future application (*e.g.* the analyze by P.U. Tournoux *et al.* [TLB+09] on the *rollernet* dataset [BL09])

Sensor nodes can also be deployed in an area to track specific objects. In the military *PinPtr* [SML+04] system, sensor nodes are statically deployed in an area to detect and locate snipers.

**Great Duck Island Example** The monitoring in the Great Duck Island of the Leach's Storm Petrels [MCP+02] is representative of many applications in this domain. The deployment of sensors aims to understand (1) the usage pattern of nesting burrows over a particular 24-72 hour cycle, (2) the changes observed in the burrow and surface environmental parameters during the course of the approximately 7 month breeding season and (3) the differences in the micro-environments with and without large numbers of nesting petrels.

The study could have been done in field by researcher. However, they are becoming increasingly concerned about the potential impacts of human presence in monitoring plants and animals in field conditions. Disturbance effects are of particular concern in small island situations. Research in Maine [And95] suggests that even a 15 minute visit to a cormorant colony can result in up to 20% mortality among eggs and chicks in a given breeding year. In this context, WSNs is a significant advances as it can be deployed prior to the onset of the breeding season or other sensitive period.

Each goal of the study requires unique data needs and suitable acquisition rates. The WSN is connected to a base station that has internet-connectivity to send data and receive management messages. The network consists of sensor nodes that have to run for 9 months from non-rechargeable power sources (except for the gateway

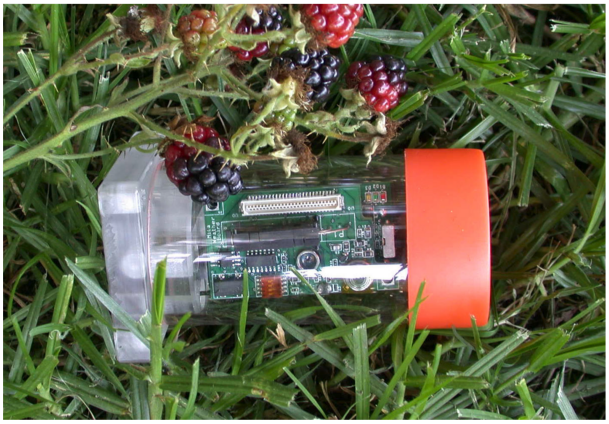


Figure 2.4 – Acrylic enclosure used for deploying the Mica mote. From [MCP+02]

that is harvested by solar panels). Each node (a Mica mote shown in Figure 2.4) runs on a pair of AA batteries. By simply dividing the available capacity of the battery by the number of days, researchers decided to give a "power budget" of 6.9 mAh per day. This budget is used to predict what a node is able to perform during a day including message reception, transmission, sensor reading, etc. After 4 weeks of deployment, researchers have calculated that the motes have sufficient power to operate for the next six months, which is below the specification. This study shows the importance of reducing energy consumption of protocols in WSNs and of accurate lifetime estimation.

## 2.2 Sensor Components

Sensor nodes are composed of two main parts: the hardware and the software.

### 2.2.1 Hardware

A sensor node consists of four units: processing, transceiver, sensing, and power units [HNL08, AZAFA14]. To reduce manufacturing cost, sensor nodes are also available as platforms that usually include the processing, transceiver, and power units. The sensing unit is either optional or included, depending on the application needs. Popular platforms, in research institutions at least, include the Mica2, MicaZ and Telosb/Tmote Sky [HNL08]. One particularity of sensor node units are the different operational modes. For instance the processing unit usually has several modes to work with or without the flash memory, at different clock speeds, and also has several sleep modes: a sleep mode with a fast wake up, and a low power sleep mode that has the lowest energy consumption but requires more time to wake up from. The energy consumption is usually represented by its instantaneous current, measured in ampere (A).

The sensing unit can contain several types of sensor depending on the application. The most used include a temperature sensor, a pressure sensor, an accelerom-

eter/gyroscope, an image sensor, and a light sensor. Depending on the sensor, the instantaneous current can vary from tens of microampere to tens of milliampere.

The processing unit consists in a low power microprocessor. The most used ones, which include the Atmel Atmega 128L and the TI MSP430F series, have a clock speed between 4 and 16 MHz, with less than 10KB of RAM. The typical instantaneous current of a processing unit is a few milliampere.

The transceiver unit contains a low power radio. The majority of platforms used the Chipcon CC2420 compatible with the IEEE 802.15.4. In addition to the four common modes, transmitting (TX), receiving (RX), idle, and sleep, the radio may propose modes for several transmission powers. Using a small transmission power implies a lower energy consumption but a smaller transmission range. The typical instantaneous current is between 10 and 20 mA for the TX and RX modes, and around $1\mu A$ for the sleep mode.

The power unit consists of a voltage regulator and either a simple battery or a storage device such as a rechargeable battery or super-capacitor. The storage device can be harvested from the environment with photovoltaics, piezoelectric, or other devices. The capacity of a battery is measured in ampere hour (Ah). Rechargeable batteries are usually lithium-ion battery and can have a capacity from tens of mAh to a few Ah.

**TMote Sky Platform** As an example, we detail here the characteristics of the TMote Sky platform (see Figure 2.5). It is an ultra low power module designed at the University of California, Berkeley. It has integrated humidity, temperature, and light sensors. Programming and data collection are performed via USB. Tmote Sky is powered by two AA batteries. AA cells may be used in the operating range of 2.1 to 3.6V DC, however the voltage must be at least 2.7V when programming the microcontroller flash or external flash. The main electric characteristics are presented in Table 2.1.

### 2.2.2 Software

**Operating System** There exist a variety of operating systems supporting different system platforms and designed to optimized node longevity and reliability. The most used include TinyOS [LMP+05] and ContikiOS [DGV04]. The amount of RAM required to run is particularly important. We further detail the architecture of ContikiOS.

A running Contiki system consists of the kernel, libraries, the program loader, and a set of processes. A process is basically defined by an event handler function. Interprocess communication is done by posting events. The events are dispatched by the kernel (which is simply a lightweight event scheduler) to run processes. Additionally the kernel periodically calls processes' polling handlers. A key feature of ContikiOS is the protothread-based [DSVA06] processes to run efficiently while keeping a good flow control. A protothread is a programming abstraction between multi-threading and event-driven programming. Contiki Processes cooperatively

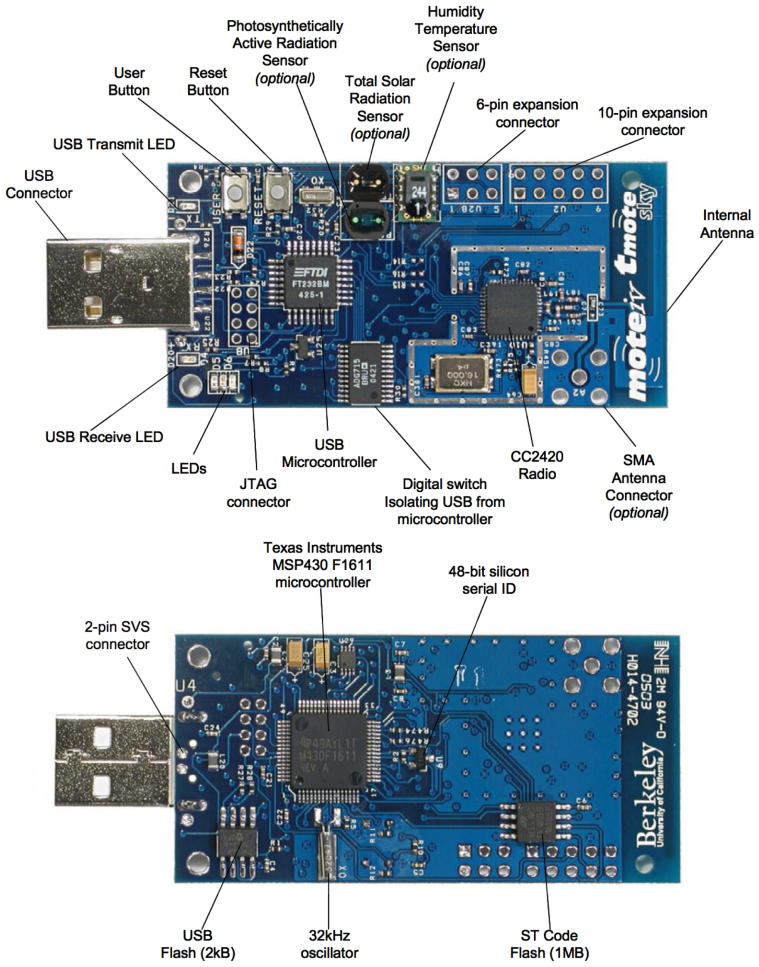



Figure 2.5 – Front and Back of the Tmote Sky platform. From [inc]

| Radio Chipcon CC2420 | | CPU Texas Instruments MSP430 F1611 | |
|---|---|---|---|
| Tx 0dB | 17.4 mA | | |
| Tx -1dB | 16.5 mA | Run 8MHz 3V | 500 $\mu$A |
| Tx -3dB | 15.2 mA | Reading flash | 4 mA |
| Tx -5dB | 13.9 mA | Sleep | 2.6 $\mu$A |
| Tx -7dB | 12.5 mA | Voltate cut-off | 2.7 V |
| Tx -10dB | 11.2 mA | Humidity and Temperature Sensor Sensirion SHT15 | |
| Tx -15dB | 9.9 mA | | |
| Tx -25dB | 8.5 mA | Measuring | 0.55 mA |
| Rx | 19.7 mA | Sleep | 0.3 $\mu$A |
| Idle | 365 $\mu$A | | |
| Sleep | 1 $\mu$A | | |
| Volt. Regulator | 20 $\mu$A | | |

Table 2.1 – Voltage specification of the TMote Sky Hardware

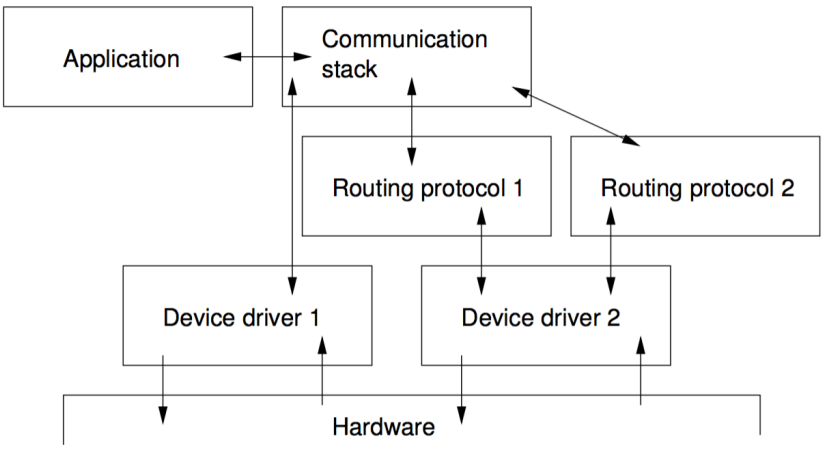



Figure 2.6 – Communication stack processes in ContikiOS. From [DGV04]

run alternatively by explicitly yielding control back to the kernel at regular intervals. When a process is called (trigger by an event from an hardware event or from another process), it is executed until it explicitly yields. When it yields, its state is saved (using only a few tens of bytes) and the scheduler can proceed with the next event.

Contiki works on a lot of platforms (including TMote Sky). Furthermore, the communication stack may be split into different processes as shown in Figure 2.6. This enables run-time replacement of individual parts of the communication stack. Communication processes use the interprocess communication mechanism to call each other and synchronous events to communicate with application programs [DGV04].

Contiki is developed by a world-wide team of developers, support a wide range of platforms, as several Radio Duty Cycling drivers (for the MAC layer) and has its own simulator (Cooja).

**Communication Standards** The IEEE 802.15.4 standard[80203] specifies the physical layer and the Media Access Control (MAC) layer for low-rate wireless personal area network. The standard can be extended by developing the upper layer, such as ZigBee [Z+06] ISA100.11a [ISA], WirelessHART [HAR], etc. The MAC layer defined by the IEEE 802.15.4 standard can be replaced by compatible layers that consume less energy, such as duty cycling MAC protocols (ContikiMAC [Dun11], B-MAC[PHC04], X-MAC[BYAH06], BoX-MAC[ML08], etc).

The IEEE 802.15.6 standard [802b] specifies short-range, wireless communications in the vicinity of, or inside, a human body (but not limited to humans) *i.e.*, the physical layer and the MAC layer of wireless body area networks. This standard considers effects on portable antennas due to the presence of a person (varying with male, female, skinny, heavy, etc.), radiation pattern shaping to minimize the specific absorption rate into the body, and changes in characteristics as a result of the user motions.

The routing protocol is particularly important as it can have a great impact on the energy consumption of the device. Also, there is a number of protocols designed for a specific application such as broadcast or convergecast, and for a specific kind of node, such as nodes with variable transmission range.

One of the most fundamental tasks performed by sensor nodes in a WSN is to disseminate data and to retrieve data to the base station. The convergecast can use data compression or data aggregation to reduce communication cost and increase reliability of data transfer [YMG08]. The data-compression technique consists in reducing the size of the data before the transmission. There is no lost of information, all the data are retained, and the decompression occurs at the base station. With data aggregation, the data from multiple sensors is combined before the transmission to the base station. Part of the data may be lost, but the most important part is received by the base station. For instance, when querying the maximum temperature in the network, if a node receives the temperature sensed by a neighbor, it can compare the value with its own value and forward only the greatest value.

## 2.3 Challenges

To conclude this chapter we present the major challenges of WSN. S.Sharma *et al.* [SBB13] also highlighted many other challenges that are more application-specific, such as security, robustness, multimedia communication, etc.

**Energy** Sensor nodes are battery-powered and may be deployed in area not accessible by humans (due to physical risks such as military or past-catastrophic applications, or due to application requirements such as monitoring the Great Duck Island as explained before). This characteristic forces the sensor nodes to be sustainable. This can be done by predicting accurately the amount of energy needed by the application to run according to the specification during a given duration. Another way to ensure that the node remains periodically powered is to allow the node to use an energy harvesting component, such as solar panels.

**MAC Layer** The MAC layer has a direct impact on the energy consumption of wireless sensor nodes. Indeed, listening, sending and receiving control messages and acknowledgments are actions that originate from the MAC layer and consume lots of energy. However, they also participate to the reliability and low latency of application communications. Thus, the challenge is to find a good trade-off.

**Routing** Routing protocols impact the way communications are handled in the network. A bad routing protocol may cause a subset of nodes to consume much energy unnecessarily. Also, the routing protocol may be responsible for weak node reachability, for instance if the protocol has an incomplete view of the topology of the network.

**Evolving Topology** When the topology evolves over the time, the majority of existing protocols may fail as they assume static (or at least stable) topology. However, there are an increasing amount of applications that produce dynamic topologies, including drone or robot networks, or when monitoring animals.

**Limited Memory** The limited amount of memory can be a challenge when it comes to storing the local state of the network, or to buffer the data reads from the sensor before transmission. However, it can also be a solution for different problems. For instance, oblivious protocols imply fault-tolerant applications that perform well in dynamic topology.

Chapter 3

# Model

In open country, the most probable place to find a drunken man who is at all capable of keeping on his feet is somewhere near his starting point!

Karl Pearson

**Contents**



In this chapter we present the model used in the remaining of the thesis. As any other networks, it is convenient to represent a WSN as a simple graph where the nodes represent the devices and the edges the communication links between them. However, we show that in a wireless context, one may not simply model communication links independently. In fact the existence of a communication link (that represents the availability for a node at one extremity to transmit a message to the node at the other extremity) is subject to other factors, such as the transmission of another node somewhere else in the network.

In Section 3.1, we present the model for the physical layer that defines how the simultaneous transmission signals are received by the nodes. WSN simulators use this model to evaluate accurately the performances of protocols. Then, in Section 3.2 and 3.3, we show a simple way to model static and dynamic WSNs as graphs and evolving graphs respectively. Despite their simplicity, these models offer a good abstraction to analyze protocols with a theoretical point of view. Finally, we present several models to generate random static and dynamic graphs.

## 3.1 Physical Layer Modeling

In a WSN, messages are sent with a radio transceiver on a shared medium. In this context, modeling the way the signal propagates is crucial to evaluate how the message is received by other nodes. Physical layer modeling includes the radio range modeling, the radio link modeling, and the interference modeling [HCG09].

### 3.1.1 Radio Range Modeling

The range modeling is based on the Signal to Noise Ratio (SNR) of links. The SNR of a link $(i, j)$, denoted $snr(i, j)$, is defined as the path-loss $PL_{i,j}$ times the ratio between the power of the signal $P_i$ over the power of the background noise $N_j$ at node $j$:

$$snr(i, j) = \frac{PL_{i,j}P_i}{N_j}$$

Let $\theta$ be the SNR threshold. Assuming the system interference free, the state of the radio link is binary: if the signal to noise ratio of a link is above the threshold, the link is on, otherwise, the link is off.

**Propagation models** The path-loss may depend on the transceiver property, the distance between the nodes, and other environmental aspects [PSG13]. It defines the way a signal sent by a node is attenuated when received by another node. Analytic models that are commonly used in wireless network simulators include the Freespace model. In a complex environment, other effects such as shadowing and fading appears. To efficiently take these effects into account, there exist several models that add a random variable to a path-loss model to account for additional fading and shadowing in the wireless channel. For instance, the Log-Normal Shadowing model and the Rayleigh Fading model are commonly used. The shadowing corresponds to slow variations of the signal due to obstacles. With multiple obstacles, the signal may be split in different paths and different copies of the signal create a fading effect, corresponding to fast variations in the signal amplitude.

### 3.1.2 Radio Link Modeling

To model the radio link, one can replace the fixed SNR threshold by a random variable representing the Packet-Error-Rate (PER). The PER is a function of the Bit-Error-Rate (BER), which is derived from the radio properties and the modulation. Given a modulation, there exist different techniques to compute the BER [WG03].

### 3.1.3 Interference Modeling

Interference occur when multiple signals from different transmissions overlap. At some point, interference can prevent the correct reception of a message. Interference mostly comes from signals in the same frequency band. To take the interference into

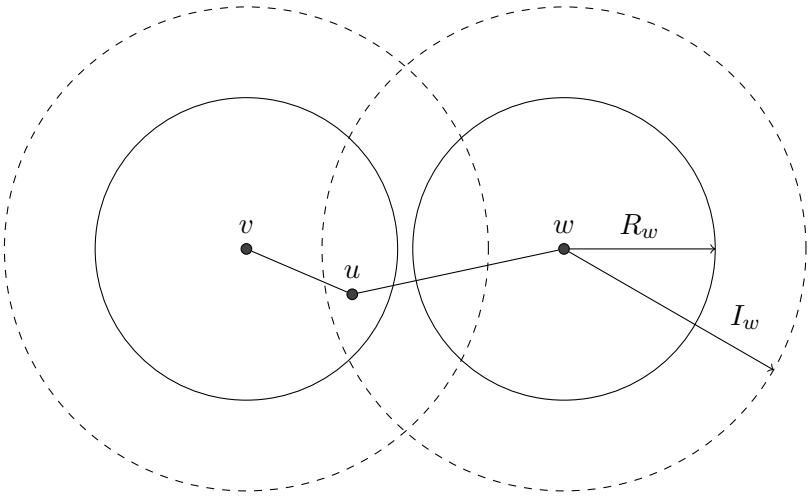



Figure 3.1 – If $v$ and $w$ transmit simultaneously, $u$ will not receive the message from $v$, because it is in the interference range of $w$

account, we can replace the SNR ratio of a link with the Signal-to-Interference-plus-Noise Ratio (SINR) defined as follow:

$$sinr(i,j) = \frac{PL_{i,j}P_i}{N_j + \sum_{k \neq i,j} PL_{k,j}P_k}$$

To study by simulation the efficiency of an algorithm that is executed by devices communicating through wireless signals, it is necessary to consider an interference model. The survey by P. Cardieri [Car10] show the impact of the interference model on various network layers. A. Iyer *et al.* [IRK09] show that the interference model has a huge impact on both scheduled transmission networks and random access networks. Their conclusion is that all models used for quantitative evaluation purposes should at least include SINR considerations. The vast amount of research in this topic highlights the fact that it is not only necessary to have an interference model when analyzing WSN, but it is essential to have a good one.

## 3.2 WSN as a Unit-Disk Graph

A simple and abstract way to model a WSN is to assume that nodes are deployed in a 2-dimensional Euclidean space[1] and that each node $u$ has a communication range $R_u$ and an interference range $I_u \geq R_u$. Then, a communication link exists from a node $u$ to a node $v$ if and only if the Euclidean distance between them is smaller than the communication range $R_u$ of $u$. Then, we can model the network as its *communication graph* $G(V, E)$ where $V$ is the set of nodes and $E$ the set of

1. We suppose here that the area where the nodes are deployed is a two dimensional plane, but our results naturally extend to greater dimensions

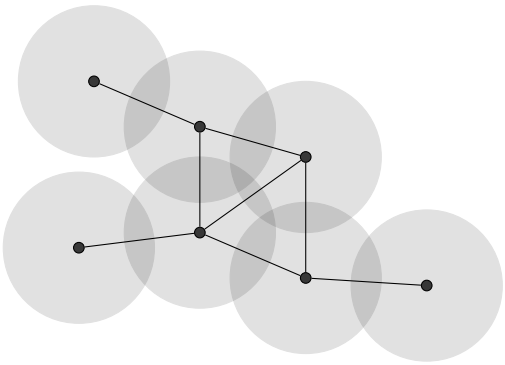


Figure 3.2 – An example of a unit-disk graph

directed communication links that verifies

$$(u, v) \in E \Leftrightarrow d(u, v) \leq R_u$$

Where $d(u, v)$ denotes the Euclidean distance between nodes $u$ and $v$. In the same way we define the interference graph $G^i(V, E^i)$ where edges in $E^i$ verify

$$(u, v) \in E^i \Leftrightarrow d(u, v) \leq I_u$$

The interference graph informs us where collisions occur when a given subset of nodes transmit simultaneously. When a node $u$ transmits a message, a neighbor $v$ in the communication graph is able to receive the message, only if there is no node $w$ transmitting at the same time and having $v$ as neighbor in the interference graph (see Figure 3.1). Formally, a node $v$ receives a message from a transmitting node $u$, if and only if $(u, v) \in E$ and there is no transmitting node $w \in V \setminus \{u\}$ such that $(w, v) \in E^i$.

For the sake of simplicity, we can assume that all the nodes are identical, their communication ranges are normalized to 1, and their interference ranges equal their communication ranges. In this case, the communication graph and the interference graph coincide and the condition for a node $v$ to receive a message from a neighbor $u$ is that no other neighbor transmits at the same time. Also, the communication graph is a special case of intersection graph called Unit-Disk Graph (UDG) [CCJ90] (see Figure 3.2) *i.e.*, a node is represented by a disk of radius 1/2 in the plane, and an edge exists between two nodes if the intersection between their disks is not empty (disks are supposed closed, so that tangent disks intersect).

In this simple model, we usually assume discrete time and normalized the duration of the transmission of a message to 1 time unit. In this context, we are able to use the language of graph theory. For instance, we say that the transmission of a message sent by a node $u$ to a node $v$ is done along a path from $u$ to $v$, which is a sequence of edges where the first edge contains $u$, the last edge contains $v$ and two consecutive edges have one node in common. Then, the length of the path is exactly

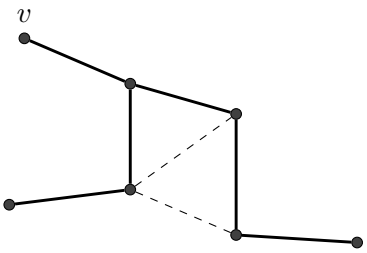



Figure 3.3 – A shortest path tree

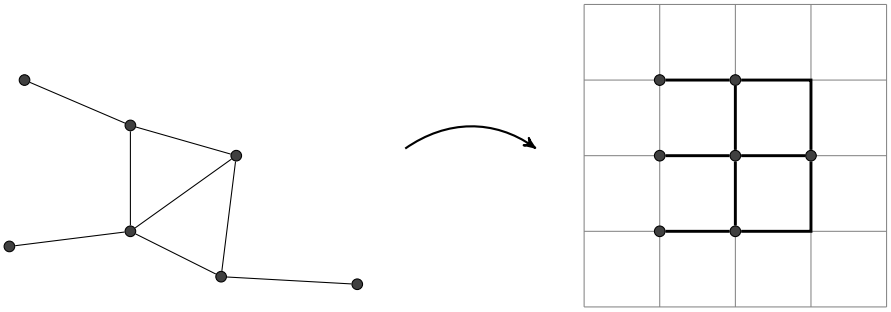


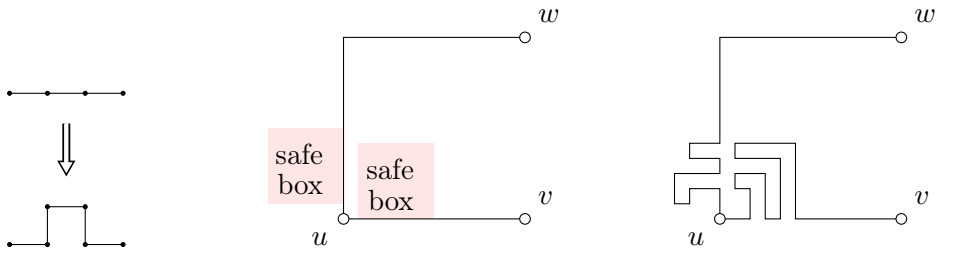
Figure 3.4 – A planar graph and its embedding in the $5 \times 5$ grid.

the duration between the transmission of the message by $u$ and its reception by $v$.

In the sequel, $n$ denotes the cardinal of $V$ and $\Delta$ denotes the maximum node degree of $G$.

**Shortest Path Tree** A shortest path between two nodes of a graph $G(V, E)$ is a path with minimal length. In a network, it usually represents the best route for the transmission of a message between these nodes. An interesting extension is to look at all the shortest paths from a given node $v$ to all the other nodes (or equivalently, from all the other nodes to $v$). From this we can obtain a tree, where each path from $v$ to another node is a shortest path. This tree, called the *shortest path tree*, is not unique, but gives an optimal strategy to perform a broadcast from $v$ (or a convergecast to $v$), see Figure 3.3. This definition is given in a static graph and we show in Section 3.3.3 one way (among others) to extend it in a dynamic graph.

**Orthogonal Planar Embedding** An orthogonal planar embedding (or drawing) of a planar graph $G(V, E)$ is a mapping of each vertex to a distinct point in a grid, and of each edge to a curve made of horizontal and vertical segments such that edges intersect only at a vertex (see Figure 3.4). The embedding of a planar graph can be used to create a UDG from a planar graph, with the help of the following Lemma, used in chapter 5.




(a) replacement of an edge of length 3 by an edge of length 5

(b) positions of the safe boxes

(c) both edges have the same length

Figure 3.5 – Construction of the orthogonal embedding

**Lemma 3.1**

*Let $H$ be a planar graph with $n > 6$ nodes and maximum degree $\Delta \leq 4$, there exists an orthogonal planar embedding of $H$ such that each edge has the same length. This embedding can be computed in time polynomial in $n$.*

**Proof:** From [BK98], we know that there exists an orthogonal embedding of $H$ in a grid $g$ of size $n$ where each edge has at most 2 bends (so each edge has length smaller than $3n$). Let $g_i$ be the grid $g$ where the unit has been divided by $4i$, of size $4in$. We consider the corresponding embedding of $H$ in $g_i$ (so that the coordinates of vertice has been multiplied by $4i$). In $g_i$, the maximum length of an edge is $3 \times 4in$.

We can increase the length of an edge by 2 by replacing a piece of edge between two points with integer coordinates by a new piece of edge, like in figure 3.5a. This is possible if there are no other edges around.

By choosing $i \geq 48n$, there is enough space in a subgrid $(2i-2) \times (2i-1)$, called *safe box*, to contain an edge of length $3 \times 4in$. So that, if $i \geq 48n$, since all the length are even, we can lengthen all the edges to make them have a length $3 \times 4in$. Indeed, we can lengthen a given edge in the first $(2i-2) \times (2i-1)$ subgrid of an end point, in the anticlockwise direction (see figure 3.5). ■

## 3.3 Dynamic Networks

A dynamic network is a network whose topology (links' existence and properties) and characteristic (nodes' properties) evolve over time. This domain of computer science has grown in the past decade. A big difference with a static network, is that in a dynamic network, this change of property, or link appearance and disappearance, is not seen as a failure but is part of the normal evolution of the network.

There exists a number of models to represent such networks. Here we present the two most common models. The general Time-Varying Graph model that is complex but that encompasses all the other models, and the simpler Evolving graph model that can be used to model the majority of networks. The lifetime $\mathcal{T}$ of a dynamic

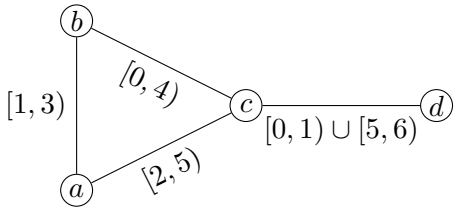



Figure 3.6 – A simple TVG. The interval(s) on each edge $e$ represents the periods of time when it is available, that is, $\{t \in \mathcal{T} : \rho(e,t) = 1\}$. From [CFQS11].

network is the time instants used to represent the evolution of the graph. It is a subset of a *temporal domain* $\mathbb{T}$, which is usually $\mathbb{N}$ for discrete-time systems or $\mathbb{R}_+$ for continuous-time systems.

### 3.3.1 Time-Varying Graphs

Time-varying graphs have been defined by A.Casteigts *et al.* [CFQS11] in the following way:

**Definition 3.1**

*A Time-Varying Graph (TVG) is a tuple $(V, E, \mathcal{T}, \rho, \zeta)$, where:*

- *$(V, E)$ is a labeled graph*
- *$\rho : E \times \mathcal{T} \mapsto \{0, 1\}$ is the* presence *function that indicates whether a given edge is available at a given time.*
- *$\zeta : E \times \mathcal{T} \mapsto \mathbb{T}$ is the* latency *function that indicates the time it takes to cross a given edge if starting at a given date.*

An example of a TVG is presented in Figure 3.6.

### 3.3.2 Evolving Graphs

We consider the special case of discrete TVGs with temporal domain $\mathcal{T} = \mathbb{T} = \mathbb{N}$ and a constant latency function $\rho$ that equals 1 for every edge at any time. Under those assumptions, the graph is called an *evolving graph i.e.*, a sequence of snapshots, where each snapshot represents the time-varying graph at a given time $t \in \mathbb{N}$ (see Figure 3.7). Since the latency is 1, messages can travel at most one hop at a given time[2]. An evolving graph can be defined using a simpler formalism than a TVG.

**Definition 3.2**

*An evolving graph is a couple $(V, E)$ where $V$ is a set of nodes and $E = (E_t)_{t\in\mathbb{N}}$ is a sequence that represents the edges between nodes over the time. The snapshot at time $t$ is the graph $(V, E_t)$ and represents the topology of the network at time $t$.*

2. One can also consider that the latency is null, so that a message can travel along a full path at a given time

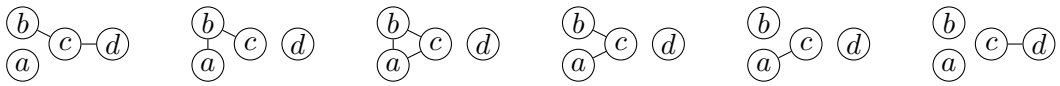



Figure 3.7 – The evolving graph corresponding to the TVG of Figure 3.6

In this model, an edge is a couple $(e, t)$ where $e$ is an edge between two nodes in the snapshot $(V, E_t)$. $t$ is called the time of occurrence or the edge. In the remaining of the thesis, we only consider evolving graphs or its variant.

### 3.3.3 Definitions and Preliminaries

If it is clear from the context, $\Delta$ denotes the maximum node degree among all the snapshots of $G$.

**Underlying graph** The *underlying graph* of an evolving graph $G = (V, (E_t)_{t\in\mathbb{N}})$ is the graph $\overline{G} = (V, \bigcup_{t\in\mathbb{N}} E_t)$. This graph is also called the *footprint* of $G$. This graph captures all the edges of the evolving graph *i.e.*, if an edge exists in the underlying graph then this edge exists at a given time. However this information is sometimes not helpful as it may become obsolete after some time (*i.e.*, if a link appears only a finite number of times). That is why we can also consider the *eventual underlying graph*. The eventual underlying graph of an evolving graph $G = (V, (E_t)_{t\in\mathbb{N}})$ is the graph $\overline{G}^{\infty} = (V, \bigcap_{t'\in\mathbb{N}} \bigcup_{t>t'} E_t)$ *i.e.*, an edge exists in $\overline{G}^{\infty}$ between two nodes if those nodes are connected in $G$ infinity often.

**Journey** A journey from a node $u$ to node a $v$ is a sequence of edges $((e_1, t_1), \ldots, (e_r, t_r))$ such that $(e_1, e_2, \ldots, e_r)$ is a path from $u$ to $v$ in the underlying graph and

$$\forall i \in [1..r-1],\ t_i < t_{i+1} \qquad \forall i \in [1..r],\ e_i \in E_{t_i}$$

For a journey $J$, we denote by $departure(J)$ the starting time $t_1$ and by $arrival(J)$ the arrival time $t_r + 1$ of the journey. The arrival time corresponds to the time of the existence of the last edge plus the latency to travel along the last edge. Then, $duration(J) = arrival(J) - departure(J)$ denotes the duration of the journey. We denote by $\mathcal{J}_{(u,v)}$ the set of journeys from $u$ to $v$ and by $\mathcal{J}_{(u,v)}^{[t_s,t_e]}$ the subset of journeys that start and end between $t_s$ and $t_e$.

**Foremost Convergecast Trees** We introduce the notion of convergecast trees and foremost convergecast trees.

**Definition 3.3**

> *Let $G(V, E)$ be an evolving graph. A* convergecast tree *to node $s$ is a pair $(T, c)$, where $T(V, E_T)$ is a tree rooted at $s$, and $c$ is a function $c : E_T \to \mathbb{N}$ that satisfies: if $u$ is a descendant of $v$ in $T$ and $(e_1, e_2, \ldots, e_r)$ is the path from $u$ to $v$ in $T$,*

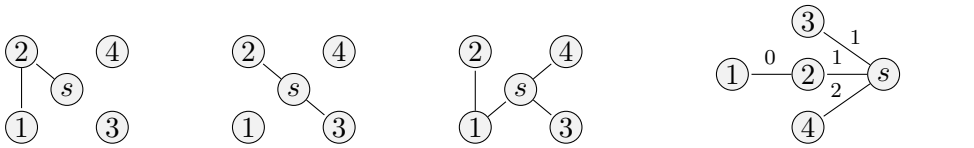



Figure 3.8 – An example of foremost convergecast tree.

*then*

$$((e_1, c(e_1)), (e_2, c(e_2)) \ldots, (e_r, c(e_r)))$$

*is a journey in $G$ called the journey from $u$ to $s$ induced by $T$.*

The *departure* (respectively, the *arrival*) of the convergecast tree is the departure of the first journey in $T$ (respectively, the arrival of the last journey in $T$):

$$departure(T, c) = \min_{e \in E_T} c(e) \qquad \text{and} \qquad arrival(T, c) = \max_{e \in E_T} c(e) + 1$$

**Definition 3.4**

*Let $G(V, E)$ be an evolving graph. A Foremost Convergecast Tree (FCT) to node $s$ starting at time $t_s$ is convergecast tree $(T, c)$ to $s$ such that $departure(T, c) \geq t_s$ with minimum arrival time.*

$FCT(G, s, t_s)$ denotes the set of foremost convergecast trees of $G$ to node $s$ starting after time $t_s$. The common duration of foremost convergecast trees starting after $t_s$ is denoted $FCTD(G, s, t_s)$.

In dynamic WSNs, a foremost convergecast tree plays the same role as a shortest path tree in static WSNs. Indeed it gives an optimal routing strategy to perform a convergecast to $s$. Figure 3.8 shows an example of the unique foremost convergecast tree to the sink node $s$ starting at time 0 of a simple evolving graph.

### 3.3.4 Hierarchy of Dynamic Graphs

A hierarchy of classes of dynamic graphs has been identified in previous work [CFQS11]. Here we present only the few we are interested in.

— $\mathcal{C}$ *(Connectivity over the time)*: there exists a journey between any two nodes. $\forall u, v \in V$:

$$\mathcal{J}_{(u,v)} \neq \emptyset$$

— $\mathcal{RC}$ *(Recurrent connectivity)*: there exists a journey between any two nodes, infinity often. $\forall u, v \in V$, $\forall t \in \mathbb{N}$:

$$\mathcal{J}_{(u,v)}^{[t,+\infty)} \neq \emptyset$$

— $\mathcal{BRC}$ *(Time-bounded recurrent connectivity)*: there exists a bound $T$ such that, there exists a journey between any two nodes in every interval of duration $T$.

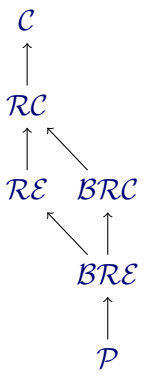



Figure 3.9 – Relations of inclusion between classes

$\forall u, v \in V,\ \forall t \in \mathbb{N}$:

$$\mathcal{J}_{(u,v)}^{[t,t+T]} \neq \emptyset$$

— $\mathcal{RE}$ *(Recurrence of edges)*: the graph is connected over the time and if an edge appears once, it appears infinitely often. $\forall u, v \in V\ \mathcal{J}_{(u,v)} \neq \emptyset$ and:

$$(\exists t,\ (u,v) \in E_t) \Rightarrow \left(\forall t \in \mathbb{N},\ (u,v) \in \bigcup_{t'>t} E_{t'}\right)$$

— $\mathcal{BRE}$ *(Time-bounded recurrence of edges)*: the graph is connected over the time and there exists a bound $T$ such that, $\forall u, v \in V$:

$$(\exists t,\ (u,v) \in E_t) \Rightarrow \big(\forall t \in \mathbb{N},\ \exists t' + \in [t, t+T], \quad \text{s.t.} \quad (u,v) \in E_{t'}\big)$$

— $\mathcal{P}$ *(Periodic)*: the graph is connected over the time and there exists $T \in \mathbb{N}$ such that:

$$\big(\exists t,\ (u,v) \in E_t\big) \Rightarrow (\forall k \in \mathbb{N},\ (u,v) \in E_{t+kT})$$

## 3.4 Random Graphs

In this section we present several models to generate random graphs.

**Erdős–Rényi graphs** An Erdős–Rényi graph [ER59, Bol98] is a static graph constructed by connecting nodes randomly. Each edge is included in the graph with probability $p$ independent from every other edge. Equivalently, all graphs with $n$ nodes and $M$ edges have equal probability of

$$p^M (1-p)^{\binom{n}{2}-M}$$

The connectivity of Erdős–Rényi graphs is well-known: If $p < \frac{(1-\varepsilon)}{n}$, then the graph almost surely contains isolated vertices; if $p > \frac{(1+\varepsilon)}{n}$ the graph is almost surely connected. Where almost surely means that the probability tends to 1 as $n$ tends to infinity. An evolving graph $G(V, E)$ can be constructed with this model if we assume that each snapshot $E_t$ is an Erdős–Rényi graph.

**Edge-Markovian Evolving Graphs** An edge-Markovian graph [CMM+10] is an evolving graph constructed in the following way. The first snapshot can be any graph (either given, or randomly generated). Then, at every time step, every edge changes its state (existing or not) according to a two-state Markovian process with probabilities $p$ and $q$. If an edge exists at time $t$, then at time $t + 1$ it dies with probability $q$ (i.e. death-rate). If instead the edge does not exist at time $t$, then it will come into existence at time $t + 1$ with probability $p$ (i.e. birth-rate). Observe that when $q = 1 - p$, then the construction is time-independent and correspond to an evolving graph where each snapshot is an Erdős–Rényi graph with parameter $p$.

**Random Unit-Disk Graph** Here, and in the sequel, we assume that the positions are chosen in a simulation area $A$ that is convex (usually we consider a square). A random unit-disk graph with $n$ nodes in an area $A$ is an unit-disk graph where each node has a random position in the area $A$.

**Random Walk** [Pea05] To generate an evolving graph from a random walk, with step $s$ in an area $A$, we initially consider either a given or a random unit-disk graph. Then, a snapshot at time $t$ is constructed by assigning a node a random position in $A$ at distance at most $s$ from its position at time $t - 1$.

**Random Waypoint** [JM96] To generate an evolving graph using the random waypoint model, with speed $s$ in an area $A$, we initially consider either a given or a random unit-disk graph. Then each node chooses a random destination in $A$ and starts to move toward it with a speed $s$. Once a node reaches its destination, it chooses a new destination randomly in $A$. The speed can be fixed, or different for each node, chosen uniformly at random at the beginning of the construction in an give interval, or varying over time.

A variant of the random waypoint called the Manhattan random waypoint assumes that movement can only be horizontal or vertical, to simulate the movement in the streets of a city. There are other variants too, for instance where the movement are smoothed. One can also consider other kind of simulation area. For instance with the random waypoint on a torus or the random waypoint on a sphere [BV05].

**Markovian Evolving Graph** All above-mentioned graph models are Markovian Evolving Graph (MEG) [AKL08]. Formally, the process is a MEG if the sequence of snapshots, seen as random variables, $(G(V, E_t))_{t\in\mathbb{N}} = G(V, E_1), G(V, E_2), \ldots$ is

Markovian, that is, if the distribution of $G(V, E_t)$ is completely determined by the distribution of $G(V, E_{t-1})$.

One can observe that all the random graph models we defined generates evolving graphs in the class $\mathcal{RE}$.

# Part II

# Data Aggregation Problem

Chapter 4

# Introduction of Part II

> The scientist does not study nature because it is useful to do so. He studies it because he takes pleasure in it, and he takes pleasure in it because it is beautiful.
>
> Henri Poincare

**Contents**



In the second part of this thesis, we deal with an important problem in WSNs: the data aggregation problem. We saw in chapter 2 that WSNs are usually made of small, often battery-powered, sensor nodes. These nodes are deployed in an area to perform a task. The nodes may be controlled indirectly by an end-user located at a specific node called *gateway*. The control is indirect because there may not exist a direct link between every node and the gateway, such that intermediate nodes are necessary to forward messages between two distant nodes. In many applications, each node is assigned to a small task (*e.g.* collecting or generating data from its environment). Then one of the fundamental communication tasks is to retrieve the result of each node to the gateway. Then, the gateway is able to analyze the data and either inform the end-user of the final result or perform a new task in response. In this context, the gateway is called the *sink*.

There are many ways to retrieve the data from a set of nodes. A simple way is to order every node to transmit its data to the sink, without looking at what the other nodes are doing. Intermediate nodes can forward the other nodes' data with respect to their routing protocol. This is called a convergecast. However, in the context of energy efficiency consideration, this may generate too many transmissions,

especially for the nodes close to the sink that are responsible for forwarding a large number of messages. In total, up to $\Omega(n^2)$ transmissions are necessary to perform a convergecast from $n$ nodes to a sink node.

In order to minimize the number of transmissions, we can assume several data can be merged together by an aggregation function. Examples of such a function include min, max, etc. Then a node can wait to receive data from its neighbors before forwarding the aggregated data toward the sink. This assumption implies that $n$ transmissions are sufficient to aggregate the data from $n$ nodes to the sink. This idea was made popular when the minimum duration of the data aggregation has been analyzed by Chen *et al.* [CHZ05] and Kesselman *et al.* [KK05] after the work of Anamalai *et al.* [AGS03]. The problem was initially defined in WSNs with static topology where interference due to simultaneous transmissions can cause collisions *i.e.*, if two nodes transmit simultaneously a message to a common neighbor, no data is received due to wireless signal interference. To avoid collisions while ensuring a small overall delay is a challenge and finding the optimal solution is NP-complete [BT15]. When the collisions are handled by a MAC layer, the problem is more practical and the challenges are different.

After extending the problem of data aggregation to dynamic WSNs, we observe that the geometric nature of WSN does not play an important role anymore, even though the possible collision between transmissions is still the root of the problem when seen with a global point of view (no collision causes the problem to be trivial for a centralized algorithm). Finally, our study on distributed solutions for this problem shows that in dynamic networks, collisions play a second role and the network can be fully abstracted and modeled by a simple sequence of interactions. In this context, the amount of information given to the nodes assumed by an algorithm is directly linked with its performance.

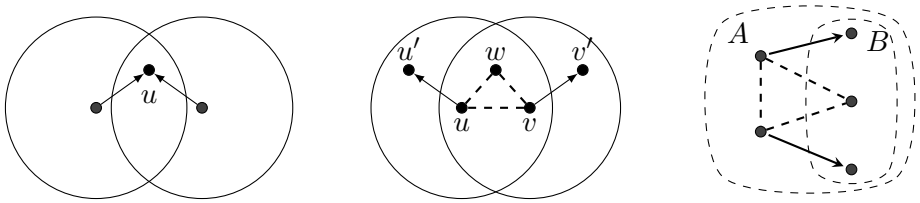
## 4.1 Problem

Let $V$ be a set of $n$ nodes that initially have a data. We assume that the time is discrete and the set of time instants is represented by the set of positive integers $\mathbb{N}$. At a given time $t \in \mathbb{N}$, a node *that has a data* can receive data from its neighbors and aggregate it with its own data. For the sake of simplicity, the duration for the reception and the aggregation of a data is normalized to one. The goal is to aggregate all the data from nodes in $V$ to a sink node $s \in V$. In the sequel $V$ always denotes the set of nodes, $n \geq 3$ its size, and $s$ the sink node.

One can observe that the way we define the aggregation implies that a node can transmit its data at most once. With this rule, the aggregation is performed with the minimal number of transmissions *i.e.*, $n - 1$. In the context of WSN, this implies that an algorithm solving the data aggregation problem is energy optimal, considering only at the consumption of radio transmissions, which is one of the most important parts of the consumed energy.

**Definition 4.1 (*Minimum Data Agregation Time Problem*)**

*The problem of aggregating the data from the nodes in $V$ to the sink $s$ with*




(a) Node $u$ does not receive data

(b) Node $u'$ receives data from $u$ and $v'$ from $v$

(c) Data is aggregated from $A$ to $B$

Figure 4.1 – Communication constraints

*minimum duration, assuming nodes can transmit at most once, is called the Minimum Data Aggregation Time (MDAT) problem.*

An instance of the MDAT problem is a couple $(G, s)$ where $G(V, E)$ models an evolving graph, and $s \in V$ represents the sink node.

We defined the problem independently from the nature of the network, static or dynamic, with or without geometric constraints, and independently from the collision avoidance capabilities of the nodes. In the following subsections, we precise the definition of the problem, as it can take different forms and can cause different challenges, depending on these parameters.

### 4.1.1 Minimum Data Aggregation Time Problem in a WSN

Here, we suppose that the network is composed of sensor nodes that communicate through wireless transmission, and that the collisions due to interference are not handled by a MAC layer. We suppose that the topology of the network can evolve over the time. Since the problem is defined with discrete time instants, we choose to model the network as an evolving graph (see definition 3.2).

Due to the wireless nature of the communication, transmissions are subject to collisions. The simplest model of interference, as defined in chapter 2, constraints the communication with the following rules. Sensor nodes can send or receive data, but cannot do both at the same time. Moreover, if two nodes send their data simultaneously, all their common neighbors do not receive anything, due to interference (see Figure 4.1a) *i.e.*, a node must have a unique transmitting neighbor in order to receive data from it. So, a solution of the MDAT problem in a WSN is a collision-free schedule, telling when each node has to transmit, so that all the data of the network is aggregated at the sink node with minimum duration. This problem is studied in chapter 5.

One can observe that the term "collision-free" does not mean that no collision occurs, but that the data that have been aggregated are received without collision. Indeed, it is possible that two nodes $u$ and $v$ transmit simultaneously their data to nodes $u'$ and $v'$ (*e.g.* in Figure 4.1b, $u$ is the only neighbor of $u'$ and $v$ is the only neighbor of $v'$) and still a collision occurs at a node $w$ (*e.g.* $u$ and $v$ are neighbors of $w$).

Let $G = (V, (E_t)_{t\in\mathbb{N}})$ be an evolving graph where each snapshot $(V, E_t)$ is a UDG.

**Definition 4.2 (*Aggregating Data From a Set to Another*)**

*Let $A \subseteq V$ and $B \subseteq A$. We say that* data is aggregated from $A$ to $B$ at time $t$ *(see Figure A.2c) if nodes in $A \setminus B$ transmit their data simultaneously and all the data is received by at least one node in $B$. Formally if:*

$$\forall u \in A \setminus B,\ \exists v \in B,\ \forall u' \in A \setminus B \setminus \{u\} :$$
$$(u, v) \in E_t \wedge (u', v) \notin E_t$$

Then, the definition of a dynamic data aggregation schedule follows.

**Definition 4.3 (*Dynamic Data Aggregation Schedule*)**

*A* dynamic data aggregation schedule *to $s$ of duration $l$ is a decreasing sequence of sets $V = R_0 \supseteq R_1 \supseteq \ldots \supseteq R_l = \{s\}$ such that, for all $t = 0, 1, \ldots, l-1$, data is aggregated from $R_t$ to $R_{t+1}$ at time $t$.*

For an instance of the dynamic MDAT problem $(G, s)$, a solution is a dynamic data aggregation schedule to $s$ with minimum duration. The minimum duration is denoted by $MDAT_{Opt}(G, s)$.

**Remark 1**

*The MDAT problem may have no solution, even in an evolving graph $G \in \mathcal{C}$ connected over time. Indeed, consider a set of edges defined as follow: $E_0 = V \times V$ and $\forall i > 0$, $E_i = \emptyset$. Then, the graph is connected, but only one node can send its data to the sink node at time 0, and the other nodes are never able to send their data. A simple sufficient (but not necessary) assumption that ensures the existence of a solution is that the graph is in the class $\mathcal{RC}$ of recurrent connected graphs (see our algorithm GDAS in the sequel).*

**Tolerating Multiple Simultaneous Transmissions** An interesting extension of the MDAT problem in WSNs is to consider wireless communications with better collision capabilities. For instance, we can assume that up to $K$ simultaneous transmissions can be received by a node, and that a collision occurs at a node if $K + 1$ neighboring nodes transmit simultaneously. Let $K \in \mathbb{N}^*$, we define the $MDAT_K$ problem as the $MDAT$ problem, with the additional assumption that nodes can simultaneously receive up to $K$ messages from $K$ different neighbors. We show in chapter 5 that the results under this model are intuitive prolongations of the case $K = 1$.

### 4.1.2 Distributed Online Data Aggregation in an Arbitrary Dynamic Network

Here, we consider the MDAT problem in an arbitrary dynamic network, such as sensors deployed on a human body, cars evolving in a city that communicate with each other in an ad hoc manner, etc. We focus our study on the distributed aspect of the problem. To do so, we consider that nodes execute the same algorithm and have to make a decision in an online manner *i.e.*, each node processes its input in the order that the input is fed to the algorithm, without having the entire input available from the start.

The essence of such a data aggregation algorithm is to decide whether or not to send a node's data when encountering a given communication neighbor. Also, a node may base its decision on its past experience (past interactions with other nodes) and initial knowledge only. Then, an algorithm accommodating those constraints is called a Distributed Online Data Aggregation (DODA) algorithm. The existence of such an algorithm is conditioned by the (dynamic) topology, initial knowledge of the nodes (*e.g.* about their future communication neighbors, or some partial information about the evolving graph), etc.

For the sake of simplicity, we assume that interactions between the nodes are carried out through pairwise operations (so that no collision occurs). Anytime two nodes $a$ and $b$ are communication neighbors (or, for short, are interacting), either no data transfer happens, or one of them sends its data to the other. The receiver executes the aggregation function on both its previously stored data and the received data and the output replaces its stored data. In the sequel, we use the term *interaction* to refer to a pairwise interaction.

We assume that an adversary controls the dynamics of the network, that is to say, the adversary decides what are the interactions. As we consider atomic interactions, the adversary decides what sequence of interactions is to occur in a given execution. Then, the sequence of static graphs to form the evolving graph can be seen as a sequence of single edge graphs, where the edge denotes the interaction that is chosen by the scheduler at this particular moment. Therefore, the time when an interaction occurs is exactly its index in the sequence.

This leads to model the dynamic network as a couple $(V, I)$, where $I = (I_t)_{t\in\mathbb{N}}$ is a sequence of pairwise interactions (or simply interactions). In the sequence $(I_t)_{t\in\mathbb{N}}$, the index $t$ of an interaction also refers to its *time of occurrence*. One can observe that this model is a restriction of the evolving graph model, where each snapshot consists of a graph with a single edge.

In general, we consider that nodes in $V$ have unique identifiers, unlimited memory and unlimited computational power. However, we sometimes consider nodes with no persistent memory between interactions; those nodes are called *oblivious*.

During an interaction $I_t = \{u, v\}$, if both nodes still own data, then one of the nodes has the possibility to transmit its data to the other nodes. If a node decides to transmit its data, then it does not own any data, and is not able to receive other's data anymore.

**Distributed Online Data Aggregation Algorithms** The distributed online data aggregation problem consists in choosing at each interaction whether a node transmits (and which one) or not so that after a finite number of interactions, the sink is the only node that owns a data. An algorithm solving this problem is called a Distributed Online Data Aggregation (DODA) algorithm.

A DODA algorithm takes as input an interaction $I_t = \{u, v\}$, and its time of occurrence $t \in \mathbb{N}$, and outputs either $u$, $v$ or $\perp$. If a DODA outputs a node, this node is the receiver of the other node's data. For instance, if $u$ is the output, this means that before the interaction both $u$ and $v$ own data, and the algorithm orders $v$ to transmit its data to $u$. The algorithm is able to change the memory of the interacting nodes, for instance to store information that can be used in future interactions. In the sequel, $\mathcal{D}_{\mathsf{ODA}}$ denotes the set of all DODA algorithms. And $\mathcal{D}^{\emptyset}_{\mathsf{ODA}}$ denotes the set of DODA algorithms that only require oblivious nodes.

A DODA algorithm can require some knowledge to work. A knowledge is a function (or just an attribute) given to every node that gives some information about the future, the topology or anything else. By default, a node $u \in V$ has two pieces of information: its identifier $u.ID$ and a boolean $u.isSink$ that is true if $u$ is the sink, and false otherwise. A DODA algorithm may use additional functions associated with different knowledge. $\mathcal{D}_{\mathsf{ODA}}(\mathfrak{i}_1, \mathfrak{i}_2, \ldots)$ denotes the set of DODA algorithms that use the functions $\mathfrak{i}_1, \mathfrak{i}_2, \ldots$. For instance, we define for a node $u \in V$ the function $u.meetTime$ that maps a time $t \in \mathbb{N}$ with the smallest time $t' > t$ such that $I_{t'} = \{u, s\}$ *i.e.*, the time of the next interaction with the sink (for $u = s$, we define $s.meetTime$ as the identity, $t \mapsto t$). Then $\mathcal{D}_{\mathsf{ODA}}(meetTime)$ refers to the set of DODA algorithms that use the information $meetTime$.

**Adversary Models** We consider three models of adversaries:

— The oblivious adversary. This adversary knows the algorithm's code, and must construct the sequence of interactions before the execution starts.

— The adaptive online adversary. This adversary knows the algorithm's code and can use the past execution of the algorithm to construct the next interaction. However, it must make its own decision as it does not know in advance the decision of the algorithm. In the case of deterministic algorithms, this adversary is equivalent to the oblivious adversary.

— The randomized adversary. This adversary constructs the sequence of interactions by picking pairwise interactions uniformly at random.

**Definition of Cost** To study and compare different DODA algorithms, we use a tool slightly different from the competitive analysis that is generally used to study online algorithms. The competitive ratio of an algorithm is the ratio between its performance and the optimal offline algorithm's performance. However, in our case, one can hardly define objectively the performance of an algorithm. For instance, if we just consider the number of interactions before termination, then an oblivious adversary can construct a sequence of interactions starting with the same interaction

repeated an arbitrary number of times. In this case, even the optimal algorithm has infinite duration. Moreover, the adversary can choose the same interaction repeatedly after that the optimal offline algorithm terminates. This can prevent any non optimal algorithm from terminating and makes it have an infinite competitive-ratio.

To prevent this, we define the cost of an algorithm. Our cost is a way to define the performance of an algorithm, depending on the performance of the optimal offline algorithm. We believe our definition of cost is well-suited for lots of problems where the adversary has a strong power, especially in dynamic networks. One of its main advantages is that it is invariant by trivial transformation of the sequence of interactions, like inserting or deleting duplicate interactions.

For the sake of simplicity, a data aggregation schedule with minimum duration (performed by an offline optimal algorithm) is called a *convergecast*. Consider a sequence of interactions $I$. Let $opt(t)$ be the ending time of a convergecast on $I$, starting at time $t \in \mathbb{N}$. If the ending time is infinite (if the optimal offline algorithm does not terminate) we write $opt(t) = \infty$. Let $T : \mathbb{N}_{\geq 1} \mapsto \mathbb{N} \cup \{\infty\}$ be the function defined as follows:

$$\begin{aligned} T(1) &= opt(0) \\ \forall i \geq 1 \quad T(i+1) &= opt(T(i)+1) \end{aligned}$$

$T(i)$ is the duration of $i$ successive convergecasts (two convergecasts are consecutive if the second one starts just after the first one completes).

Let $duration(A, I)$ be the termination time of algorithm $A$ executed on the sequence of interactions $I$. Now, we define the cost $cost_A(I)$ of an algorithm $A$ on the sequence $I$, as the smallest integer $i$ such that $duration(A, I) \leq T(i)$:

$$cost_A(I) = \min\{i \mid duration(A, I) \leq T(i)\}$$

This means that $cost_A(I)$ is a tight upper bound on the number of successive convergecasts we can perform during the execution of $A$, on the sequence $I$. It follows from the definition that an algorithm performs an optimal data aggregation if and only if $cost_A(I) = 1$.

Also, if $duration(A, I) = \infty$, then it is possible that $cost_A(I) < \infty$. Indeed, if $i_{\max} = \min_i\{i \,|\, T(i) = \infty\}$ is well-defined, then $cost_A(I) = i_{\max}$, otherwise $cost_A(I) = \infty$.

## 4.2 Related Work

There are two fundamental tasks in WSNs: data dissemination and data aggregation. After a quick review of the existing results on data dissemination, we present the related work on data aggregation. The small number of work related to

the data aggregation problem in dynamic networks under our model motivates our study.

### 4.2.1 Data Dissemination Problem

The time to disseminate a message in the network is also called the *flooding time*. In Markovian evolving graphs, Clementi *et al.* [CMM+10, CMPS09, CST15] show tight bounds on the time to disseminate a data. First, in edge-Markovian evolving graphs an upper bound of $O\left(\frac{\log n}{\log(1+np)}\right)$ time steps holds With high probability (w.h.p.) [1] and is proven to be tight for a wide range of settings. The flooding time in stationary MEGs is studied in [CMPS09]. In the case of geometric-MEG, with $R$ denoting the transmission radius and $r$ denoting the speed, Clementi *et al.* [CMPS09] showed an upper bound of $O\left(\frac{\sqrt{n}}{R} + \log\log R\right)$ and a lower bound $\Omega\left(\frac{\sqrt{n}}{R+r}\right)$ time steps. After similar results for specific mobility models [CMS13], Clementi *et al.* [CST15] provide an upper bound to the flooding time of any MEG. Their upper bound holds if the MEG converges to a unique stationary distribution, which is a probability distribution over n-node graphs and is called the stationary (random) graph $\mathcal{G}$. The upper bound is function of the worst-case probability of appearance of an edge, the degree of independence among edges in $\mathcal{G}$ and, the mixing time of the MEG (the worst-case time, required to reach a distribution which is "close" to $\mathcal{G}$).

In a more practical setting, W.Badreddine *et al.* [BCPPB15] reviewed and compared different strategies to disseminate information in BANs, depending on the delay and the number of messages used.

### 4.2.2 Data Aggregation Problem

The problem of data aggregation has been widely studied in the context of wireless sensor networks. There exists also some work on dynamic networks. The literature on this problem can be divided in two groups depending on the assumption made about the collisions being handled by an underlying MAC layer.

**When Collisions are not Handled by the MAC Layer** This case corresponds to the minimum data aggregation time problem defined in subsection 4.1.1. This problem has been initially studied by Anamalai *et al.* [AGS03]. The authors assume that a fixed number of channels is available for a transmission, and a collision occurs at a receiver whenever two of its neighbors transmit on the same channel at the same time. The authors propose an algorithm that constructs a collision-free convergecast tree that can also be used for broadcasting.

1. An event $A$ occurs with high probability, when $n$ tends to infinity, if $P(A) > 1 - O\left(\frac{1}{\log(n)}\right)$

Then, Kesselman *et al.* [KK05] and Chen *et al.* [CHZ05] formalized the problem, equivalent to the convergecast problem defined by Anamalai *et al.* with a unique channel, and gave the first analytic bounds. However, the problem defined by Kesselman *et al.* differs slightly as no particular node plays the role of the sink, and the transmission range of nodes are variable (so that the communication graph is directed). Chen *et al.* [CHZ05] presented a well-defined model for the study of the MDAT problem (as we defined it in subsection 4.1.1) and proved that the problem is NP-complete, even in graphs of degree at most four (more precisely, they restricted the problem to networks whose topology is a sub-graph of the grid, which cannot be considered directly as a UDG). They stated that a lower bound for the MDAT problem is the minimum between the height of shortest path tree (which gives an optimal solution if there is no collision) and $\log(|V|)$ (In the best case, at each time, the number of nodes with data can be divided by two). They also gave a $(\Delta - 1)$-approximation algorithm.

After the work of Chen *et al.* [CHZ05], a variety of papers proposed centralized and distributed approximation algorithms in static WSNs using geometric aspect of the MDAT problem to improve the data aggregation delay. Yu *et al.* [YLL09] give a distributed algorithm with an upper bound at $24D + 6\Delta + 16$ (where $D$ is the diameter, and $\Delta$ the maximum degree of the graph). Xu *et al.* [XLM+11] and Ren *et al.* [RGL10] proposed centralized algorithms with upper bounds at $16R + \Delta - 14$ and $16R + \Delta - 11$, respectively (where $R$ is the radius of the graph). The best known bound is due to Nguyen *et al.* [NZC11], as they give a centralized algorithm that takes at most $12R + \Delta - 11$ time slots to aggregate all data.

**When collisions are handled by the MAC layer** Various problems related to data aggregation have been investigated. The general term *in-network aggregation* includes several problems such as gathering and routing information in WSNs, mostly in a practical way. For instance, a survey [FRWZ07] relates aggregation functions, routing protocols, and MAC layers with the objective of reducing resource consumption. *Continuous aggregation* [AMadH14] assumes that data have to be aggregated, and that the result of the aggregation is then disseminated to all participating nodes. The main metric is then the delay before aggregated data is delivered to all nodes, as no particular node plays the role of a sink.

Most related to our concern is the work by Cornejo *et al.* [CGN12]. In their work, they consider a finite time evolving graph. Pairwise interaction occurs between nodes *i.e.*, for every snapshot, each node is included in at most one edge. Each node starts with a token and no particular node plays the role of a sink node. Then, when two nodes interact, one of them can decide to send or not its token. Token must not be lost nor duplicated *i.e.*, each token must be owned by a unique node at any time. However, a node that transmits its token can receive it again later.

The goal is to minimize the number of nodes that own at least one token (such nodes are called uploaders) at the end of the execution. An algorithm has to decide for each interaction whether a node transmits its token or not. The number

of uploaders at the end of the execution of an algorithm is then compared with the optimal offline algorithm to compute the competitive ratio. Cornejo *et al.* [CGN12] prove that there are dynamic graphs where the optimal offline algorithm can aggregate to a single node, but with high probability any randomized algorithm will aggregate to $\Omega(n)$ nodes, where $n$ is the network size *i.e.*, the competitive ratio of any randomized algorithm is $\Omega(n)$ with high probability against an oblivious adversary.

This impossibility result leads the author to consider the problem in $p$-cluster graphs (where $p \in (0, 1]$). A dynamic graph is a $p$-cluster if by the end of the execution, every node has interacted with at least $p$-fraction of all nodes. Cornejo *et al.* described a randomized algorithm $ClusterAggregate_p$ that with high probability aggregates all tokens to $O(\log n)$ nodes when executing on a $p$-cluster.

This problem differs from the MDAT problem, mainly because of the fact that there is no sink node. This implies that transmitting is always a good choice. In the MDAT problem, reducing the number of nodes that own data is not always useful, since a node without data does not participate to the process anymore.

## 4.3 Contribution of Part II

### 4.3.1 The Complexity of Data Aggregation in WSNs

The results on the complexity of data aggregation in WSNs are presented in Chapter 5. On this topic, the contribution is fourfold. First, in order to compare the complexity of the data aggregation in static and dynamic WSN, we give a tight bound for the complexity of the MDAT problem in static WSN. In more details, we show that, in a *static* WSN, the problem remains NP-complete when the graph is a partial grid of degree at most three (a particular case of WSN topology). As it is trivial to solve the problem in static graph of degree at most two, our result implies that the problem is intrinsically difficult for any practical setting. This result closes the complexity gap in the static case.

Second, we introduce an extension of MDAT problem in dynamic WSNs, and we prove that the MDAT is NP-complete in a *dynamic* WSN of degree at most two (and it is trivial to solve the problem if the graph is of degree at most one). This result does not use the geometric aspect of the graph, compared to the static case, as it remains true for arbitrary graphs and not only for unit disk graphs. We also show that allowing simultaneous transmissions to the same node is not intrinsically helpful as it only delays the complexity wall: we show that the problem remains NP-complete if a node can correctly receive up to $K > 1$ simultaneous packets from different neighbors, if the maximum node degree of the graph is $K + 2$ in the static case (and $K + 1$ in the dynamic case).

Third, we give the first lower and upper bounds for the dynamic MDAT problem. More precisely, the minimum time to aggregate all data in a dynamic network is greater than the duration of a foremost convergecast tree (this result is valid in *any* graph, and for any degree $\Delta$, there exists a dynamic graph such that the bound is attained) and is smaller than the duration of $n-1$ independent foremost convergecast

trees (this later bound is valid for any graph, but actually obtained for dynamic graphs of degree $n-1$). If we restrain the class of dynamic graphs to those of degree smaller than $n-1$, we prove that the upper bound is greater or equal to the duration of $l$ independent foremost convergecast trees (with $l = (\Delta-1)\log_\Delta(n(\Delta-1)+1)-\Delta+2$), which prevents previous approximation algorithms for the static case to be extended to the dynamic case.

Four, we observe that, even in periodic graphs, optimal solutions cannot be computed by an algorithm that is unaware of the future of the graph or by a distributed algorithm (even if each node knows its own future). This motivates our simple approximate algorithm presented in section 5.4 to be centralized with full knowledge (yet, it does not assume that the graph is a dynamic WSN in the sense that it can perform on arbitrary graphs). The approximation factor is $T(n-1)$ if there exists a bound $T$ such that there is a journey between every two nodes in every time interval $[t, t+T]$ *i.e.*, in the $T$-time-bounded recurrent connectivity class of graphs.

### 4.3.2 Distributed Online Data Aggregation

The results on the distributed online data aggregation problem are presented in Chapter 6. It turns out that the problem difficulty strongly depends on the power of the adversary (that chooses which interactions occur in a given execution).

For the oblivious and the online adaptive adversaries, we give several impossibility results when nodes have no knowledge about the future evolution of the dynamic graph, nor about the topology. In more details, we prove $(i)$ that there exists an online adaptive adversary that generates, for every DODA algorithm, a sequence of interactions such that the cost of the algorithm is infinite and $(ii)$ that there exists an oblivious adversary that generates, for every oblivious (randomized) DODA algorithms, a sequence of interactions such that the cost of the algorithm is infinite.

Also, when nodes are aware of the underlying graph, the data aggregation is impossible in general. To examine the possibility cases, we use our cost function to compare the performance of a DODA algorithm to the optimal offline algorithm on the same sequence of interactions. Our results show that if all interactions in the sequence occur infinitely often (*i.e.*, in the $\mathcal{RE}$ class), there exists a distributed online data aggregation algorithm whose cost is finite. Moreover, if the underlying graph is a tree, we present an optimal algorithm.

For the randomized adversary, we first present tight bounds when nodes have full knowledge about the future interactions in the whole graph. In this case, the best possible algorithm terminates in $\Theta(n\log(n))$ interactions with high probability. Then, we consider nodes with restricted knowledge, and we present two optimal distributed online data aggregation algorithms that differ in the knowledge that is available to nodes. The first algorithm, called *Gathering*, assumes nodes have no knowledge whatsoever, and terminates in $O(n^2)$ interactions in expectation, which we prove is optimal without knowledge. The second one, called *Waiting Greedy*, terminates in $O\left(n^{3/2}\sqrt{\log(n)}\right)$ interactions with high probability, which we show is optimal when each node only knows the time of its next interaction with the sink

(the knowledge assumed by Waiting Greedy).

We believe our research paves the way for stimulating future researches, as our proof arguments present techniques and analysis that can be of independent interest for studying dynamic networks.

Chapter 5

# The Complexity of Data Aggregation in Wireless Sensor Networks

**Contents**



In this chapter we study the complexity of the Minimum Data Aggregation Time (MDAT) problem in static and dynamic wireless sensor networks. We said in the previous chapter that, in a static WSN, this problem is NP-complete [CHZ05], but the question is: what is the smallest set of graphs, with regard of the maximum node degree, for which the problem is NP-complete? After answering this question in static and in dynamic WSNs, we give the first upper and lower bounds for the problem that are valid in a dynamic WSN. Finally, we give the first approximation algorithm for the MDAT problem in dynamic networks.

## 5.1 NP-Completeness

In this section we show what are the settings for which the MDAT is NP-complete, in static and in dynamic WSNs.

### 5.1.1 Static grid graphs of degree at most three

A grid graph is a unit disk graph where all disks have centers with integer coordinates and radius 1/2 *i.e.*, an induced sub-graph of the grid. However, a sub-graph of the grid (not necessarily induced, called partial grid) is not necessarily a grid graph.

Chen *et al.* prove in [CHZ05] that finding the minimum data aggregation time is $NP$-hard, even when the network is restricted to partial grid (with maximum degree

$\Delta = 4$). On the other side, if the maximum degree of a static graph is $\Delta = 2$, the graph is either a line or a cycle and the minimum data aggregation time is easy to compute. Let $\varepsilon$ be the eccentricity of the sink node and $n$ the number of nodes. If $n$ is odd and $\varepsilon = (n-1)/2$ (the graph is either a cycle or a line with the sink node in the middle) then the MDAT is $\varepsilon + 1$. Otherwise the MDAT is $\varepsilon$.

In this section we close the complexity gap of the MDAT problem in static networks by proving that the MDAT is $NP-$hard, even when restricted to grid graphs with maximum degree $\Delta = 3$.

We use a construction that is similar to that of Chen *et al.* [CHZ05] with some improvements about the topology (grid graph instead of partial grid) and about the maximum node degree (3 instead of 4).

Before stating our first theorem, we recall the definition of the restricted planar 3-SAT [Lic82]. Let $\varphi$ be a $3-$SAT formula composed by a set $C$ of $m$ clauses $c_1, \ldots, c_m$ over a set $V$ of $n$ variables $v_1, \ldots, v_n$. We define the corresponding formula graph $G_\varphi = (V \cup C, E_1 \cup E_2)$, where $E_1 = \{(x_i, c_j) : x_i \in c_j \text{ or } \bar{x}_i \in c_j\}$ and $E_2 = \{(x_i, x_{i+1}) : 1 \leq i \leq n-1\} \cup \{(x_n, x_1)\}$. $\varphi$ is said to be *planar* if the formula graph $G_\varphi$ is planar. $\varphi$ is said to be *restricted* if *(i)* each variable appears in at most three clauses, *(ii)* both negated and unnegated forms of each variable appears at least once, and *(iii)* clauses drawn on the same side of the cycle $E_2$ must share the same literal if they share the same variable (*i.e.*, at a variable vertex in $G_\varphi$, incident edges from one side correspond to the same literal). It is known that restricted planar 3-SAT is NP-complete [Lic82].

**Theorem 5.1**

*The MDAT problem restricted to grid graphs of degree at most three is NP-complete.*

**Proof:** The proof is by reduction from restricted planar $3-$SAT. Let $\varphi$ be an instance of the restricted planar $3-SAT$ on $n$ variables and $m$ clauses. From the planar formula graph $G_\varphi$, we construct a planar graph $G$ with maximum degree $\Delta = 3$. The idea behind the construction is that, in order to have a fast data aggregation, the schedule must "choose" between two sides (*i.e.*, the data are aggregated along one of two possible paths) representing the true or false instantiation of a variable. The aggregation is fast if all clauses are connected to the correct side of at least one variable. First we construct the sub-graph $X_i$ that represents the variable $x_i$ (see figure Fig. 5.1.1).

$X_i$ is composed of a cycle of nodes $e_i$, $l_i$, $r_i$, $s_i$, $o_i$, $\bar{s}_i$, $\bar{r}_i$, $\bar{l}_i$. Then, we connect to $l_i$ (resp. $\bar{l}_i$) a line $L_i$ (resp. $\bar{L}_i$) of length $5i-3$. Thus $l_i$ cannot sends its data before aggregating data from $L_i$ *i.e.*, before $5i-3$ timeslots. For $1 \leq i < n$ we connect $o_i$ to $e_{i+1}$ and we connect to $e_1$ a new node $e_0$.

Each clause $c_j$ is represented by a single node and for each variable $x_i$ (resp. negation $\bar{x}_i$) in clause $c_j$, we connect $c_j$ to $r_i$ or $s_i$ (resp. $\bar{r}_i$ or $\bar{s}_i$) by a line $\Psi_{i,j}$ of length $(5i-2)$ to $r_i$ (resp. $\bar{r}_i$) or $(5i-1)$ to $s_i$ (resp. $\bar{s}_i$). Let $G = \bigcup_{1 \leq i \leq n}^{1 \leq j \leq m} \left(X_i \cup L_i \cup \bar{L}_i \cup \Psi_{i,j}\right)$ (see figure Fig. 5.1.1 and Fig. 5.1.2).

In order to use the previous lemma we need to be able to change the distance between nodes. So we define $G_T$ obtained from $G$ by replacing every edge by a

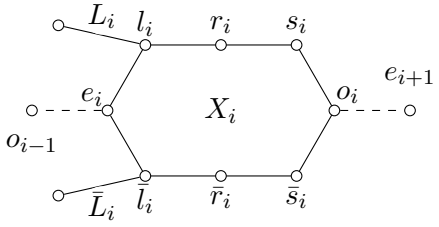



Fig. 5.1.1: sub-graphs $X_i$, $L_i$ and $\bar{L}_i$

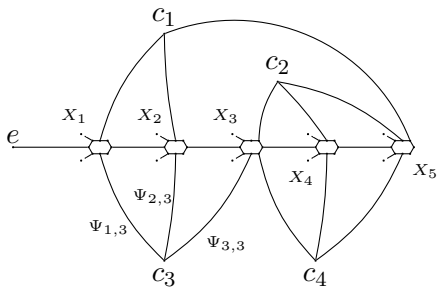



Fig. 5.1.2: example for $\varphi = (x_1 \vee x_2 \vee \bar{x}_5) \wedge (x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_3 \vee x_4 \vee x_5)$

line of length $T$ *i.e.*, by adding $T-1$ nodes between two connected nodes, and by adding a pending node $e$ connected to $e_0$.

In the next three Claims we show that there exists a $T$ such that $G_T$ is a grid graph (of degree at most 3) and that the minimum time to aggregate data from $G_T$ to $o_n$ is $5nT+1$ if and only if $\varphi$ is satisfiable. Then, the theorem follows from the NP-completeness of restricted planar 3−SAT [Lic82].

**Claim 1.** *There exists a $T$ such that $G_T$ is a grid graph.*

From lemma 3.1 we deduce that $G$ has an orthogonal embedding such that every edge has the same length $l \geq 1$ in a grid of size $s$. We divide the unit by 4 so that the embedding is in a grid of size $4s$ and every edge has length $4l$ (every vertex has its coordinates multiplied by 4). Then, we replace, in its embedding, each node by a disk of radius 1/2, and every edge by a chain of $4l-1$ disk of radius 1/2, centered at integer coordinates along the edge. Finally, we add a disk of radius 1/2, centered at integer coordinates, at distance 1 from $e_0$ and at distance greater than 1 from other disks. The corresponding unit disk graph (that is also a grid graph) is exactly $G_{4l}$.

**Claim 2.** *For all $T \geq 1$, if $\varphi$ is satisfiable, then the minimum time to aggregate data from $G_T$ to $o_n$ is $5nT+1$.*

First of all, we remark that the distance between $e$ and $o_n$ is $5nT+1$, so that $5nT+1$ is a lower bound for the minimum data aggregation time.

Now suppose that $\varphi$ is satisfiable and let $I$ be an interpretation of the truth-functional propositional calculus satisfying $\varphi$. Let $i \in [1..n]$ and suppose $I(x_i) = true$. Suppose that $e_i$ has aggregated at time $(5i-4)T+1$ data from previous nodes $X_k \cup (\Psi_{k,j} - \{c_j\})$, $k < i$ and from clauses $c_j$ containing for every $j < i$, $x_j$ if $I(x_j) = true$, and $\bar{x}_j$ otherwise. As we said before, $l_i$ (resp. $\bar{l}_i$) can aggregate all data from $L_i$ (resp. $\bar{L}_i$) at time $(5i-3)T$.

At the negative side of $X_i$, $\bar{l}_i$ can also aggregate all data of the $T-1$ nodes between $e_i$ and $\bar{l}_i$ before time $(5i-3)T$. If $\bar{r}_i$ is connected to a clause $c_j$, it can receive all data from nodes between $\bar{r}_i$ and $c_j$ ($c_j$ excluded) at time $(5i-2)T-1$. Then $\bar{r}_i$ can receive all data from $\bar{l}_i$ at time $(5i-2)T$. If $\bar{s}_i$ is connected to $c_j$, it can receive all data from nodes between $\bar{s}_i$ and $c_j$ ($c_j$ exclude) at time $(5i-1)T-1$. Then $\bar{s}_i$ can aggregate all data from $\bar{r}_i$ at time $(5i-1)T$. Thus $o_i$ can aggregate all data from $\bar{s}_i$ at time $5iT$.

On the other side of the cycle, $l_i$ has to wait time $(5i-3)T+1$ to receive data

from $e_i$. Again, if $r_i$ is connected to $c_j$, it can receive all data from nodes between $r_i$ and $c_j$ ($c_j$ included) at time $(5i-2)T$. Then $r_i$ can receive all data from $l_i$ at time $(5i-2)T+1$. If $s_i$ is connected to $c_j$, it can receive all data from nodes between $s_i$ and $c_j$ ($c_j$ included) at time $(5i-1)T$. Then $s_i$ can aggregate all data from $r_i$ at time $(5i-1)T+1$. Finally, $o_i$ can aggregate all data from $s_i$ at time $5iT+1$.

If $i=n$, then it's over, otherwise $o_i$ can start transmitting to the next block $X_{i+1}$ and $e_{i+1}$ can aggregate data from $o_i$ at time $(5(i+1)-4)T+1$, the data includes data from clauses containing $x_i$. If $I(x_i)=false$, the schedule on the positive and negative side of the cycle are exchanged in order to aggregate data from clauses containing $\bar{x}_i$ instead of $x_i$. Recursively, since all clauses are connected to a variable $x_i$ with $I(x_i)=true$ or to $\bar{x}_i$ with $I(x_i)=false$, $o_n$ aggregates all data at time $5nT+1$.

**Claim 3.** *For all $T \geq 1$, if the minimum time to aggregate data from $G_T$ to $o_n$ is $5nT+1$, then $\varphi$ is satisfiable.*

Suppose that all data from $G_T$ are aggregated to $o_n$ at time $5nT+1$. Since $e$ is at distance $5nT+1$, its data is sent directly through a shortest path *i.e.*, there is a shortest path $P$ from $e$ to $o_n$ such that, when a node in $P$ receives at time $t$ a data from $e$ (an aggregation that contains $e$'s data) it sends the aggregation to the next node in the path at time $t+1$. There are $2^n$ shortest paths from $e$ to $o_n$, indeed, at each block $X_i$, the path can use the positive or the negative side, and implicitly choose to interpret $x_i$ as true or false. Let $i \in [1..n]$, and suppose $P$ uses the positive side of $X_i$. As we saw before, data from $L_i$ and $\Psi_{i,j}$ for clauses $c_j$ that contain $x_i$ can be aggregated to $l_i$, $r_i$ and $s_i$ before the data from $e$. The first observation is that $o_i$ receives data from $e$, $L_i$ and $\Psi_{i,j}$ for clauses $c_j$ containing $x_i$ at time $5iT+1$ and has to send the aggregation just after (it cannot receive data after time $5iT+1$).

On the other side, $\bar{l}_i$ can start sending at time $(5i-3)T$ (because it must receive data from $\bar{L}_i$ first), and its data must be aggregate through a path to $o_n$ of length smaller than or equals to $5nT+1-(5i-3)T$. Because of its length, this path must pass through $o_i$, by the negative side of $X_i$. Thus data from $\bar{l}_i$ must be aggregate through a path to $o_i$ of length smaller than or equals to $3T+1$ (the maximum length of the path minus the minimum distance from $o_i$ to $o_n$). Adding this to the first observation, we know that data from $\bar{l}_i$ must be received by $o_i$ at time $5iT$ (and cannot be received before, because $\bar{l}_i$ and $o_i$ are at distance $3T$).

Thus, data from $\bar{l}_i$ cannot be delayed (except by $o_i$) *i.e.*, if a node receives a data from $\bar{l}_i$ at time $t$, it must send the aggregation to the next node at time $t+1$. Thus $\bar{r}_i$ and $\bar{s}_i$ can receive data from other nodes only before time $(5i-2)T$ and $(5i-1)T$, and so this data cannot include data from a connected clause $c_j$ (only from $\Psi_{i,j}-\{c_j\}$). Since all paths from a clause $c_j$ to $o_n$ passing trough $X_i$ contain a node $r_i$, $\bar{r}_i$, $s_i$ or $\bar{s}_i$, a connected clause can send its data through $X_i$ only if it contains $x_i$.

The same thing happens if $P$ uses the negative side of $X_i$: a connected clause can send its data through $X_i$ only if it contains $\bar{x}_i$.

For a shortest path $P$ from $e$ to $o_n$ used to aggregate $e$'s data, we say a variable $x_i$ is true if $P$ uses the positive side of $X_i$, and $\bar{x}_i$ is true otherwise. We have shown that if the data of a clause $c_j$ is received by $o_n$ on time (before time $5nT+1$), then $c_j$ contains a true literal.

Finally, if data from all clauses is received on time, then every clauses contains at least one true literal, and the formula is satisfiable.

■

Now, we extend the previous results to the MDAT$_K$ problem, which is the MDAT problem with the additional assumption that nodes can simultaneously receive up to $K$ messages from $K$ different neighbors. Note that simultaneously receiving $K+1$ or more messages still results in a collision. We show that allowing one more simultaneous transmission results in increasing by one the maximum node degree of the graph for the problem to be NP-complete. This may seems obvious at first sight, but the fact that the problem concerns unit-disk graphs makes the proof slightly technical. Indeed, every time we want to create a collision, to force the algorithm to make a choice, we have to create a node of degree $K+2$, and in a unit-disk graph, if $K$ is large enough, such a node must have two neighbors that are connected. So that we cannot connect an arbitrary amount of lines to a node while keeping the whole network as a unit-disk graph. The idea of the proof is as follow. For each node where we need a collision (see proof of theorem 5.1), we add a single special line of nodes. The resulting graph has now a maximum node degree of 4 and still have an embedding in the grid. Then, after replacing each edge in the embedding by nodes to create a unit-disk graph, we replace each special line by a unit-disk graph that creates the desired collision.

**Theorem 5.2**

*The MDAT$_K$ problem restricted to graphs of degree at most $K+2$ is NP-complete.*

**Proof:** Let $\varphi$ be an instance of the restricted planar $3-SAT$ with $n$ variables and $m$ clauses. From the planar formula graph $G_\varphi$, we construct a graph $G$ as in the proof of Theorem 5.1, with 5 additional lines $C_1$, $C_2$, $C_3$, $C_4$, $C_5$ connected to nodes $r_i, l_i, \bar{r}_i, \bar{l}_i$, and $o_i$. We connect $C_1$ (resp. $C_2$) of length $5i-2$ to $r_i$ (resp. $\bar{r}_i$), $C_3$ (resp. $C_4$) of length $5i-1$ to $l_i$ (resp. $\bar{l}_i$), $C_5$ of length $5i+1$ to $o_i$. As in the proof of Theorem 5.1 we define $G_T$ obtained from $G$ by replacing every edge by a line of length $T$ *i.e.*, by adding $T-1$ nodes between two connected nodes, and by adding a pending node $e$ connected to $e_0$. $G$ has a maximum node degree of 4, so it has an orthogonal embedding such that every edge has the same length $l \geq 1$ in a grid of size $s$ (see Lemma 3.1). We divide the unit distance by 4 so that two edges are at distance at least 4 from each other. Then, we divide the unit distance by $2(K-1)$.

Let $\varepsilon > 0$ and $K' = K-1$. For each line $C \in \{C_1, C_2, C_3, C_4, C_5\}$ of length $d$ in the embedding ($d$ is a multiple of $8K'$), we split the line in small parts of length $2K'$ at the two extremities, and length $4K'$ otherwise (see Figure 5.1). Each small part is either a straight line or two orthogonal straight lines. The goal is to replace each part by a unit-disk graph $\hat{C}$ such that, when aggregating all the data from $\hat{C}$, $K'$ nodes will want to transmit simultaneously at time $d$ their data to the node that connects $\hat{C}$ to the remaining of the graph.

Each straight line of length $4K'$ is replaced by $K' \times 4K'$ disks of radius $1/2$ spread across a grid of size $K' \times 4K'$, that is constrained in a rectangular area of width $4K'$, height $\varepsilon$, and centered at the middle of the initial line. This implies that the intersection graph of those disks (see Figure 5.1) is composed of $4K'$ cliques of size $K'$ that are connected by $K'$ lines of length $4K'$. This graph has the property that the distance between two nodes located at distinct extremities is either $4K'$ if

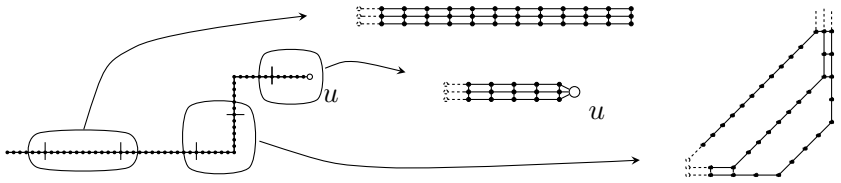



Figure 5.1 – The replacement of a line connected to node $u \in \{r_i, \bar{r}_i, l_i, \bar{l}_i, o_i\}$ by graphs containing $K' = K - 1$ lines of length $2K' - 1$ for the part connected to $u$, and $4K'$ for the parts not at an extremity.

the nodes are in the same line, or $4K'+1$ otherwise. We repeat the same process for the parts at the extremities, using only $K' \times (2K'-1)$ disks for the part connected to the remaining of the graph, and $K' \times 2K'$ disks for the part at the other extremity.

Similarly, each part composed by two lines at a right angle is replaced by $K' \times 4K'$ disks of radius 1/2 in such a way that the intersection graph contains $K'$ lines of length $4K'$ (connected with some additional edges), and has the same property as the previous intersection graph (see Figure 5.1). This implies that the intersection graph produced by all the disks of all the small parts contains $K'$ lines of length $d$ such that the distance between two nodes at distinct extremities is $d$ if they are in the same line, or $d + 1$ otherwise.

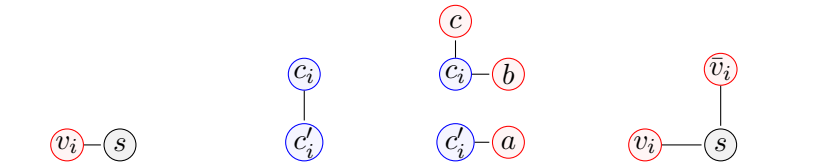
All the disks are slightly translated toward the node $u \in \{r_i, \bar{r}_i, l_i, \bar{l}_i, o_i\}$ that connects $C$ to the remaining of the graph, so that the $K'$ disks at the extremity of the line are at distance at most 1 from $u$. Also, $\varepsilon$ is chosen sufficiently small so that disks that replace Line $C$ do not intersect with other disks, located in the remaining of the graph.

Finally, as in the previous proof, we replace in the remaining of the embedding each node by a disk of radius 1/2, and every edge by a chain of $8K'l - 1$ disks of radius 1/2, centered at integer coordinates along the edge. Finally, we add a disk of radius 1/2, centered at integer coordinates, at distance 1 from $e_0$. The obtained graph is $G_{8K'l}$, where each line $C_i$ is replaced by a subgraph $\hat{C}_i$ such that, to aggregate all data from $\hat{C}_i$ without delaying the data aggregation in the whole graph, the node $u$ that connects $\hat{C}_i$ to the remaining of the graph has to aggregate the $K - 1$ neighbors in $\hat{C}_i$ simultaneously at time $d$, which equals the length of $C_i$. Then, only one other neighbor of $u$ can transmit its data to $u$ at time $d$. This simulates the constaint that no two neighbors of $u$ could transmit at time $d$ without interference in the previous setting. So, we can apply the same proof as the case $K = 1$ on the transformed graph. ■

One can observe that the problem is easy to solve in static graphs of degree at most $K + 1$. Indeed, when aggregating data along the shortest path tree, no collision occurs, except maybe at the sink node. Any schedule, for the transmissions of the neighbors of the sink, that avoid collisions is optimal.

### 5.1.2 Dynamic graphs of degree at most two

In a dynamic network we prove that, even when the maximum degree is two, the MDAT problem is NP-hard. This result is optimal since the problem is easy to




(a) configuration during $T_1$ (b) configuration during $T_2$ (c) configuration during $T_3$

Figure 5.2 – Node Configurations (clauses are blue and literals are red).

solve in a graph of degree at most one (where no collision occurs).

**Theorem 5.3**

*The MDAT problem is NP-complete even in a dynamic wireless sensor network of degree at most two.*

**Proof:** The proof is by reduction from 3-SAT. Given any 3-SAT instance $\varphi$ of $n$ variables $v_1, \ldots, v_n$ and $m$ clauses $c_1, \ldots, c_m$, we construct the dynamic grid graph $G_\varphi(V, E)$ as follow:

Nodes are composed of one sink node, literals, clauses, and copy of clauses:

$$V = \{s\} \bigcup_{1 \leq i \leq n} \{v_i\} \cup \{\bar{v}_i\} \bigcup_{1 \leq i \leq m} \{c_i\} \cup \{c'_i\}$$

Let $t_f = 3n + 2m$. We decompose the time interval $[1, t_f]$ in three periods $T_1$, $T_2$ and $T_3$ (see figure A.3):

- During $T_1 = [1, 2n]$, we have for all $i \in [1, n]$:

$$E_{2i-1} = \{(v_i, s)\}, \quad E_{2i} = \{(\bar{v}_i, s)\}$$

- During $T_2 = [2n + 1, 2n + 2m]$ we have for all $j \in [1, m]$:

$$\text{with } \{a, b, c\} = c_j, \quad E_{2n+2j-1} = \{(c_i, c'_i)\}, \; E_{2n+2j} = \{(c'_j, a), (c_j, b), (c_j, c)\}$$

- During $T_3 = [2n + 2m + 1, 2n + 2m + n]$ we have for all $i \in [1, n]$:

$$E_{2n+2m+i} = \{(v_i, s), (\bar{v}_i, s)\}$$

The remaining of the dynamic graph (after time $t_f$) can contain for instance composed only empty snapshots, or be such that the graph is periodic. This does not change the proof, but can be used to analyze the best approximation ratio for this problem (see remark 2).

During $T_3$, either a variable or its negation can send its data to the sink node $s$, but not both, so that the set of literals that send data can be seen as an interpretation of a truth-functional propositional calculus.

During $T_1$, variables that does not send their data during $T_3$ can send their data directly to the sink node.

During $T_2$, there is a link between a clause $c_i$ and its copy $c'_i$ so that either $c_i$ or $c'_i$ can send both data. Since all clauses can send their data only once to all the

literals it contains, the data is successfully sent to the sink node if and only if at least one literal it contains sends its data in $T_3$ *i.e.*, is *true*.

Thus, if the interpretation chosen in $T_3$ satisfies the formula $\varphi$, then each clause contains a literal that transmits during $T_3$, and the minimum data aggregation time is exactly $t_f$. Otherwise, some clauses must send their data before $t = 1$, and the minimum data aggregation time is greater than $t_f$.

So that the 3-SAT instance $\varphi$ is satisfiable if and only if the minimum data aggregation time ending before $t_f$ is $t_f$. ■

**Remark 2**

*Theorem 5.3 raises the question of the best approximation ratio achievable by an approximation algorithm. We observe that using time as a complexity measure is not relevant in the dynamic case. Contrarily to the static case, where good approximation ratios have been found, the problem is not approximable in the dynamic case using the duration of the solution as a measure of complexity. Indeed, the dynamic graph constructed in the proof of Theorem 5.3, with empty snapshots when time is greater than $t_f$, only contains the optimal solution. So, if an approximation algorithm finds an approximate solution, it actually finds the optimal solution, which is not possible in polynomial-time (unless $P = NP$). Even when restricted to smaller classes of graphs, such as periodic dynamic graphs, the approximation ratio (with respect to duration) can be made arbitrarily large by increasing the period. The approximation ratio can be bounded in periodic graphs, but only when the period is itself bounded (or in time-bounded recurrent connected graphs with fixed bound, as defined in Section 5.3). This remark justifies (i) our use of Foremost Convergecast Trees, defined in Section 5.2, as a complexity measure to establish upper and lower bounds, and (ii) the approximation ratio of our approximate algorithm, presented in Section 5.4, when restricted to time-bounded recurrent connected graphs.*

As in the static case, we show that when $K$ simultaneous transmissions are allowed without collisions, the problem is similar. Here, the geometric constraints do not hold, giving a more straightforward proof.

**Theorem 5.4**

*The MDAT$_K$ problem restricted to evolving graphs of degree at most $(K + 1)$ is NP-complete.*

**Proof:** In the dynamic case, there is a simpler way to apply the same trick as in the static case. From a 3-SAT instance, we create the same evolving graph as in Theorem 5.3, but with $(K - 1) \times n$ additional nodes $(r_1^1, r_2^1, \ldots, r_{K-1}^n)$, and where edges in the period $T_3$ are defined as follows: for all $i \in [1, n]$,

$$E_{2n+2m+i} = \{(v_i, s), (\bar{v}_i, s)\} \cup \bigcup_{j=1}^{K-1} \{(r_j^i, s)\}$$

So, in order for the data aggregation to terminate at time $t_f = 3n + 2m$, all nodes $r_j^i$ have to transmit during $T_3$. This implies, like in the proof of Theorem 5.3, that either a variable or its negation can transmit during $T_3$. ■

## 5.2 Upper and Lower Bounds

In this section we propose the first upper and lower bounds for the MDAT problem in dynamic networks, given in terms of foremost convergecast tree duration.

**Lemma 5.1**

*Let $G$ be a dynamic graph, we have:*

$$MDAT_{Opt}(G, s) \geq FCTD(G, s, 0)$$

**Proof:** Let $G$ be a dynamic graph, $s$ be a sink node, and $S = \{S_t\}_t$ be a data aggregation schedule to $s$ of duration $l = MDAT_{Opt}(G, s)$.

Let $x_0$ be a node different from the sink. We know that there exists $i$ such that $x_0 \in S_i$ and $x_0 \notin S_{i+1}$. Since $(G, S_i, i) \to (G, S_{i+1}, i + 1)$, there exists a node $x_1 \in S_{i+1}$ such that $(x_0, x_1) \in E_i$. We can apply the same argument to $x_1$ if it is different from the sink.

Recursively we have $x_1$, $x_2$, …, $x_p = s$ and $t_1 < t_2 \ldots < t_p < l$ such that for all $1 \leq i \leq p$, $(x_{i-1}, x_i) \in E_{t_i}$. Thus $J = \{((x_{i-1}, x_i), t_i) \mid 1 \leq i \leq p\}$ is a journey from $x_0$ to $s$ ending before $l$. We can do this with every node in $V - \{s\}$, which proves $FCTD(G, s, t_s) \leq l$. ■

In a static WSN, the same shortest path tree can be used to avoid collisions. But in a dynamic WSN, a FCT $\mathcal{T}_1$ that exists at a given time may not exist thereafter. If we delay the transmission of a node, to avoid a collision, another FCT $\mathcal{T}_2$ will be used to retry a transmission. In order to be sure that $\mathcal{T}_2$ can be used by all delayed nodes, it has to start after the end of $\mathcal{T}_1$. In this case we say that $(\mathcal{T}_1, \mathcal{T}_2)$ is a 2-time-independent FCT.

**Definition 5.1**

*A $l$-time-independent FCT of $G$ to $s$ starting at time $t_s$ is a sequence of $l$ foremost convergecast trees of $G$ to $s$ $((T_1, c_1), \ldots, (T_l, c_l))$ such that:*

- *$(T_1, c_1)$ is a FCT starting at $t_s$.*
- *for all $1 < i \leq l$, $(T_i, c_i)$ is a FCT starting at $arrival(T_{i-1}, c_{i-1})$.*

*Its duration is the sum duration of all FCT in the sequence and also equals to $arrival(T_1, c_1) - t_s$. The set of $l$-time-independent FCT of $G$ to $s$ starting at $t_s$ is denoted $FCT^l(G, s, t_s)$. The common duration of all $l$-time-independent $LTJs$ in $FCT^l(G, s, t_s)$ is denoted $LTJD^l(G, s, t_s)$.*

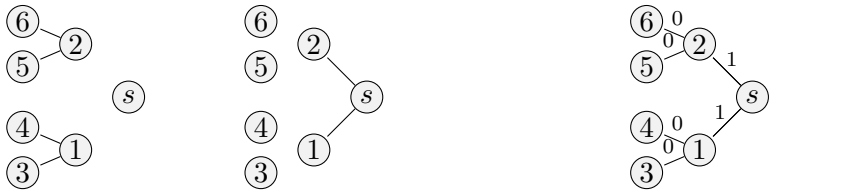



Figure 5.3 – Creation of a perfect binary FCT.

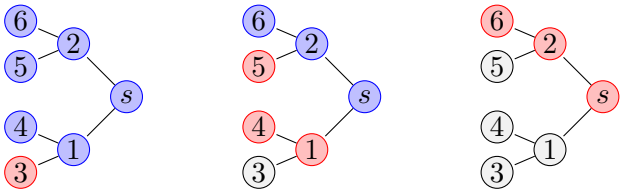


Figure 5.4 – Optimal data aggregation schedule when FCTs are complete binary trees.

This definition is used to give lower and upper bounds for the minimum data aggregation time in a dynamic graph $G$ with $n$ nodes as follow:

$$FCTD^{n-1}(G, s, 0) \geq MDAT_{Opt}(G, s) \geq FCTD(G, s, 0)$$

Where the right-hand inequality comes from lemma 5.1 and the left-hand inequality comes from the fact that a node can send its data during a FCT, so that $n-1$ foremost convergecast trees are sufficient to aggregate the data of every node.

The lower bound and the upper bound are tight in the sense that there exists a graph that reaches the lower bound (any graph of degree at most one) and a graph that reaches the upper bound (for instance a graph whose sink node is of degree $n-1$ at each time).

If we consider only graphs with a given maximum node degree $\Delta$, the lower bound is still tight, but the upper bound is no longer tight. The following lemma gives a graph with a minimum data aggregation time that lowers the upper bound, for an arbitrary maximum node degree $\Delta$. We conjecture that it also gives the worst data aggregation duration *i.e.*, that it also gives an upper bound that remains tight for an arbitrary maximum node degree.

**Lemma 5.2**

*Let $n \in \mathbb{N}^*$ and $\Delta \leq n$, there exists a dynamic graph $G$ with $n$ nodes of degree at most $\Delta$ such that:*

$$FCTD^{m}(G, s, 0) = MDAT_{Opt}(G, s) < +\infty$$

*with* $m = (\Delta - 1)\log_{\Delta}(n(\Delta - 1) + 1) - \Delta + 2$

**Proof:** Let $\Delta \geq 2$. We consider the dynamic graph $G(V, E)$. For the sake of simplicity, we suppose that there exists $h \in \mathbb{N}$ such that $|V| = n = \frac{\Delta^{h+1}-1}{\Delta-1}$. One can construct $G$ such that, there is a perfect $\Delta-$ary tree $T$ (of height $h$) such that, for all $t \equiv 0 \mod h$, $FCT(G, s, t) = \{(T, c_t)\}$ and $(T, c_t)$ is of duration $h$ (and thus with collision appearing between every node having the same parent). See for instance figure 5.3 with $\Delta = 2$ and $h = 3$. The path associated to a journey from a node $u$ to the sink is unique. So that a node has to wait to receive the data from all its children before transmitting.

Since no two children of $s$ can transmit at the same time, $s$ needs $\Delta$ FCTs to receive the data from all its children. Let $s'$ be the first direct child of $s$ that transmits, and $T(s')$ the sub-tree of $T$ rooted at $s'$. $T(s')$ is a perfect $\Delta-$ary tree of height $h-1$. When $s'$ transmits, its data contains the data of its children. Again, $\Delta$ FCTs are needed to aggregate all the data from all its children. One of these FCTs can be used to aggregate the data from $s'$ to $s$. For each layer of the tree, $\Delta - 1$ consecutive FCTs are needed.

Recursively, we need at least $(\Delta-1)h+1$ time-independent $FCTs$ to aggregate all the data from $G$. One can show that this is also sufficient (see figure 5.4). Since $h = log_\Delta(n(\Delta - 1) + 1) - 1$, the theorem is proved. ■

**Conjecture 1**

*Let $G$ be a dynamic graph with $n$ nodes of degree at most $\Delta$. Let $m = (\Delta - 1)\log_\Delta(n(\Delta-1)+1) - \Delta + 2$, we have:*

$$FCTD^m(G, s, 0) \geq MDAT_{Opt}(G, s)$$

We observe that the conjecture is proven for $\Delta = n - 1$ and is trivial if $\Delta = 1$.

## 5.3 Impossibility Results

In this section we present several classes of dynamic graphs where only a centralized algorithm that knows the future can compute optimal and "good" approximate solutions.

The two following impossibility results are for the class of periodic graphs, and naturally extend to larger classes such as recurrent connected. The first impossibility result concerns distributed algorithms. A distributed algorithm is a set of local algorithms that are each executed independently by each node. It is also assumed that nodes do *not* have knowledge about the global topology (or future global topology) of the graph; they may only have knowledge about *adjacent* edges and nodes (or their future). In general, when two nodes interact, we assume that they are allowed and able to exchange their local knowledge (for instance, about their respective local future if they are aware of it) and use this information for future interactions. For our problem, a distributed algorithm has to decide, whenever an interaction occurs, whether the node sends its data or not.

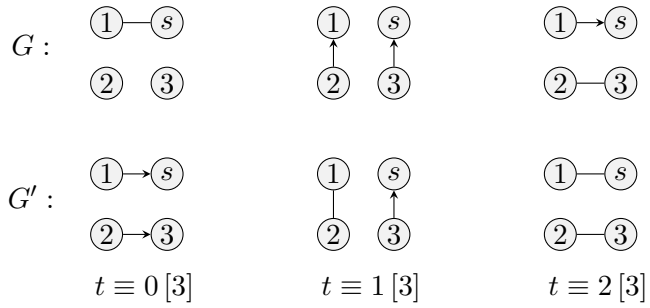



Figure 5.5 – Optimal data aggregation in graph $G$ and $G'$. In $G$, node 1 transmits at time 0 and in $G'$, node 1 does not transmit at time 0, even though node 1 has the same future in both $G$ and $G'$.

**Proposition 5.1**

*In $\mathcal{P}$, the MDAT problem does not have a distributed optimal algorithm, even if each node knows its own future.*

**Proof:** We define $G = (V, \{E_i\})$ and $G' = (V', \{E'_i\})$ as follow (see figure 5.5):

$$
\begin{array}{lll}
V = V' = [s, 1, 2, 3], \quad \forall i \in \mathbb{N}, & & \\
E'_i = \{(1, s), (2, 3)\},\ E_i = \{(1, s)\} & \text{if } i \equiv 0 & \mod 3 \\
E'_i = E_i = \{(1, 2), (3, s)\} & \text{if } i \equiv 1 & \mod 3 \\
E'_i = E_i = \{(1, s), (2, 3)\} & \text{if } i \equiv 2 & \mod 3
\end{array}
$$

At time 0 a distributed algorithm $\mathcal{A}$ does the same thing for node 1 and $s$ on $G$ and $G'$, because they have the same neighbors now and in the future. If $\mathcal{A}$ decides that node 1 transmits its data to $s$ at time 0, then $\mathcal{A}$ is not optimal on $G$ (since its faster to wait the data from node 2 at time 1 and then transmit at time 2). Otherwise, $\mathcal{A}$ is not optimal on $G'$ (since all data can be aggregated at time 1). ■

If we consider time as a measure of complexity, the following proposition shows that the best competitive ratio a distributed algorithm with knowledge of (local) future can achieve is unbounded.

**Proposition 5.2**

*In $\mathcal{P}$, the competitive ratio of a distributed algorithm is unbounded for the MDAT problem, when considering time as a measure of complexity, even if each node knows its own future.*

**Proof:** We construct two graphs, $G$ and $G'$, so that there is an arbitrary delay between the optimal solution and any other solution, resulting in unbounded competitive ratio. In more details, for all $K > 2$, we define $G_K = (V, \{E_i\})$ and $G'_K = (V', \{E'_i\})$,

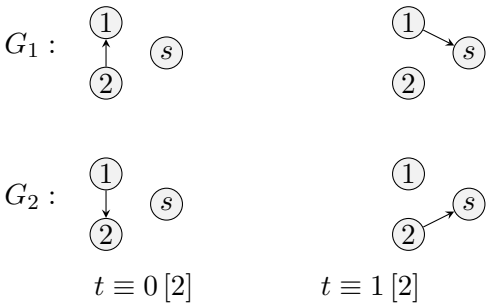



Figure 5.6 – Optimal data aggregation in graphs $G_1$ and $G_2$. In $G_1$, node 2 transmits at time 0 whereas in $G_2$, node 1 does.

periodic with period $T = 2K^2$, as follows:

$$
\begin{array}{ll}
V = V' = [s, 1, 2, 3], \quad \forall i \in \mathbb{N}, & \\
E'_i = \{(1, s), (2, 3)\},\ E_i = \{(1, s)\} & \text{if } i \equiv 0 \mod T \\
E'_i = E_i = \{(1, 2), (3, s)\} & \text{if } i \equiv 1 \mod T \\
E'_i = E_i = \{(1, s), (2, 3)\} & \text{if } i \equiv 2K \mod T \\
E'_i = E_i = \emptyset & \text{otherwise}
\end{array}
$$

So that, for an algorithm $\mathcal{A}$, either $(i)$ $\mathcal{A}$ chooses that node 1 transmits at time 0 and the time to aggregate all the data is greater than $2K^2$ compared to $2K$ for an optimal centralized algorithm (*i.e.* a ratio of $K$) or $(ii)$ $\mathcal{A}$ chooses that node 1 does not transmit at time 0 and the time to aggregate all the data is greater than $2K$ compared to 2 for an optimal centralized algorithm (*i.e.* a ratio of $K$). In both cases, the ratio $K$ between the time to aggregate the data with $\mathcal{A}$ compared to an optimal centralized algorithm can be made arbitrarily big. ■

**Proposition 5.3**

*In $\mathcal{P}$, without knowledge of the future, the MDAT problem does not allow a centralized optimal algorithm.*

**Proof :** Let $k \in \{1, 2\}$. We define $G_k$ as follow (see figure 5.6):

$$
V = [s, 1, 2], \quad \forall i \geq 0, \quad E_i = \begin{cases} (1, 2) & \text{if } i \equiv 0 \mod 2 \\ (k, s) & \text{if } i \equiv 1 \mod 2 \end{cases}
$$

Let $\mathcal{A}$ be an algorithm that does not know the future. At time 0, $\mathcal{A}$ is executed the same way on $G_1$ and $G_2$. If $\mathcal{A}$ decides that node 1 should transmit its data to node 2 at time 0, $\mathcal{A}$ cannot solve the problem on $G_1$. If $\mathcal{A}$ decides that 2 should transmit to 1 at time 0, then $\mathcal{A}$ cannot solve the problem on $G_2$. Otherwise, if $\mathcal{A}$ chooses to do nothing at time 0, the solution given by $\mathcal{A}$ is not optimal. ■

Again, if we consider the time as a measure of complexity, the following proposition shows that the best competitive ratio a centralized algorithm without the knowledge of future (called online algorithm) can achieve is unbounded.

**Proposition 5.4**

*In $\mathcal{P}$, the competitive ratio of an online algorithm (that is, a centralized algorithm without knowledge of the future) is unbounded for the dynamic MDAT problem, considering time as a measure of complexity.*

**Proof:** Let $G^{\infty}(V, E^{\infty})$ be defined as follows:

$$V = \{s, 1, 2\}, \quad \forall i \geq 0, \quad E_i^{\infty} = \begin{cases} (1,2) & \text{if } i \equiv 0 \mod 2 \\ (1,s) & \text{if } i \equiv 1 \mod 2 \end{cases}$$

Let $\mathcal{A}$ be an algorithm that does not know the future. When executing $\mathcal{A}$ on $G^{\infty}$, either *(i)* $\mathcal{A}$ decides never to transmit, and the corresponding competitive ratio of $\mathcal{A}$ is infinite, *(ii)* the first node to transmit is node 1, and the competitive ratio of $\mathcal{A}$ is infinite, or *(iii)* at some time $t$, $\mathcal{A}$ decides that node 2 transmits to node 1. Let $G^{t,T}(V, E^{t,T})$, with $T > t$, be defined as follows:

$$\forall i \geq 0, \quad E_i^{t,T} = \begin{cases} (1,2) & \text{if } i \equiv 0 \mod 2 \\ (1,s) & \text{else, if } i \equiv j \mod 3T, \text{ with } 0 < j < t \\ (2,s) & \text{otherwise} \end{cases}$$

Then, when executing $\mathcal{A}$ on $G^{t,T}$, node 2 transmits to node 1 at time $t$, and node 1 has to wait until time $3T + 1$ to transmit its data to $s$. Since the optimal offline algorithm executed on $G^{t,T}$ terminates in 2 steps if $t > 0$, and 3 steps otherwise, the competitive ratio of $\mathcal{A}$ on $G^{t,T}$ is greater than $T$. ■

## 5.4 Approximation Algorithm

In this section, we present the Greedy Data Aggregation Schedule (GDAS) approximation algorithm for the MDAT problem. The maximum duration of a solution output by this algorithm reaches the theoretical upper bound given in Section 5.2. The complexity of our algorithm is two times the number of edges of all snapshots between time 0 and time $t_f$ (where $t_f$ is the duration of the found solution).

During the first inner *while* loop, the algorithm finds the arrival time of a foremost convergecast tree of the nodes in $remainingNodes$, starting at time $t_s$. The arrival time, denoted $t_f$, becomes the starting point to find a collision-free schedule between time $t_f$ and time $t_s$ in a backward manner for the nodes in $remainingNodes$. If some nodes in $remainingNodes$ are not able to transmit between time $t_s$ and time $t_f$, we start a new iteration and compute the duration of the foremost convergecast tree consecutive to the one found in the previous iteration (its departure time is the arrival time of the previous foremost convergecast tree).

The last *for* loop converts the sequence $S$ that contains the senders over the time to a dynamic data aggregation schedule. With this method, if a node is in $S_{t_1} \cap S_{t_2}$, then only the last transmission is taken into account. The algorithm loops over all the edges, twice, for each snapshot of the dynamic graph between time 0 and the duration of the found solution.

Our algorithm uses Procedure $canTransmit(u, Senders, Receivers)$, that returns `true` if and only if Node $u$ can transmit its data to a node in $Receivers$ without interfering with other nodes in $Senders$.

**Algorithm GDAS:** Greedy Data Aggregation Schedule

**Input:** MDAT Instance $(G, s)$

```
1  S ← empty sequence
2  remainingNodes ← V \ {s}
3  t ← 0
4  do
5      t_s ← t
6      for u ∈ V do
7          data_t[u] ← {u}
8      while remainingNodes ⊄ data_t[s] do
9          data_{t+1} ← data_t
10         for (u, v) ∈ E_t do
11             data_{t+1}[v] ← data_t[u] ∪ data_t[v]
12             data_{t+1}[u] ← data_t[u] ∪ data_t[v]
13         t ← t + 1
14     t_f ← t
15     marked ← {s}
16     for t = t_f − 1, ..., t_s do
17         S_t ← ∅
18         previouslyMarked ← marked
19         for v ∈ {N_t(u) | u ∈ previouslyMarked} do
20             if canTransmit(v, S_t, marked) ∧ data_{t−1}[v] ∩ remainingNodes ≠ ∅
               then
21                 remainingNodes ← remainingNodes \ {v}
22                 marked ← marked ∪ {v}
23                 S_t ← S_t ∪ {v}
24     t ← t_f
25 while remainingNodes ≠ ∅;
26 for t = t_f − 1, ..., 0 do
27     S_t ← S_t ∪ S_{t+1}
28 return S
```

**Lemma 5.3**

*If Algorithm GDAS terminates, the sequence $\{S_i\}_{i \le t_f}$ is a valid DAS.*

**Proof:** If GDAS terminates, then the sequence $S$ satisfies $S_0 = V$, because each node that has been removed from $remainingNodes$ line 21 has been added to $S_t$ line 23. Moreover, for each time $t$, nodes in $S_t \setminus S_{t+1}$ (if not empty) have been added to $S_t$ line 23 after Function $canTransmit$ returned `true`, thus their data are successfully received by nodes in $S_{t+1}$. This implies $(G, S_t, t) \to (G, S_{t+1}, t+1)$. ■

**Theorem 5.5**

*If a graph $G$ is in $\mathcal{RC}$, algorithm GDAS finds a valid dynamic data aggregation schedule such that*

$$duration(GDAS(G,s)) \leq FCTD^{n-1}(G,s,0)$$

**Proof:** First, we observe that the inner *while* loop simulates a multicast of every node in the network and stops when the sink node receives data from the nodes in $remainingNode$. The duration $t_f - t_s$ is smaller than the duration of a foremost convergecast tree starting at $t_s$ (and it terminates in finite time since the graph is recurrent connected).

We prove that after each iteration of the main *do ... while* loop, the cardinal of the set $remainingNodes$ strictly decreases. Let $t_f^i$ be the value of $t_f$ at the end of the $i$-th iteration. Suppose we have already executed the $i-1$ iteration of the main loop. At the beginning of the $i$-th iteration, we compute in the inner *while* loop the minimum time to aggregate (maybe with collision) the data of the nodes in $remainingNodes$. Indeed, for a node $u$, $v \in data_t[u]$ implies that there exists a journey from $v$ to $u$ ending before time $t$. Since $data_{t_f^i}[s]$ contains $remainingNodes$, for each node $u \in remainingNode$, there exists a journey from $u$ to $s$ with departure greater than $t_f^{i-1} = t_s^i$ and arrival smaller that $t_f^i$.

In the *for* loop starting Line 16, we create a collision-free schedule that aggregates the data of at least one node in $remainingNodes$. Indeed, for each time $t \in [t_s^i..t_f^i]$ (in decreasing order), a node is chosen to transmit if the transmission does not create a collision (with the help of the function $canTransmit$) and the node is in a journey from a node in $remainingNodes$ to $s$ (since $data_{t-1}[v] \cap remainingNodes \neq \emptyset$). Since a node can be marked at most once, there exists a time $t$, when the only nodes (not marked) on a journey from a node in $remainingNodes$ to $s$ are themselves in $remainingNodes$. One of these nodes is chosen to transmit at time $t$ and is removed from the set $remainingNodes$.

This implies that there can be at most $n-1 = \#(V \setminus \{s\})$ iterations of the main loop. After each iteration, the duration $t_f - t_s$ is smaller than the duration of a foremost convergecast tree starting at $t_s$. Since for each iteration, the computation restart from the end of the previous one, after $i$ iterations of the main loop, $t_f^i$ is smaller than the duration of $i$ consecutive foremost convergecast trees. Since there can be at most $n-1$ iterations of the main loop, the duration of the dynamic data aggregation schedule is at most $LTJD^{n-1}(G,s,0)$. ■

If the graph is $T$-time-bounded recurrent connected, then a $FCT$ duration is smaller than $T$. Thus, we can derive an absolute bound and the following approximation factor for the dynamic MDAT problem.

**Corollary 5.1**

*Algorithm GDAS, in $\mathcal{BRC}$ with bound $T$, satisfies:*

$$duration(GDAS(G;s)) \leq T(n-1)$$

*Thus, it is an approximation of factor $T(n-1)$, for the dynamic MDAT problem.*

## 5.5 Conclusion

We studied the complexity of the minimum data aggregation time problem in wireless sensor networks. We proved that the problem is NP-complete in a static WSN of degree at most three, and NP-complete in a dynamic WSN of degree at most two. The degree constraint is crucial, as a smaller one induces a trivial solution in both cases. Then we gave tight lower and upper bounds for the minimum data aggregation time problem in dynamic networks and the first approximation scheme for the problem. Also, in a dynamic graph with $n$ nodes of degree at most $\Delta$, we conjecture a more accurate upper bound (that is tight even in the class of graphs with a given maximum node degree $\Delta$) of $l$ time-independent foremost convergecast trees (with $l = (\Delta - 1)\log_{\Delta}(n(\Delta - 1) + 1) - \Delta + 2$).

Finally we observed that only a centralized algorithm with full knowledge can compute the optimal solution of the problem. Thus, we gave a simple approximate algorithm giving a solution whose time matches the theoretical upper bound.

One can observe that allowing nodes to transmit several times their data, instead of only once, does not change the results concerning centralized algorithms that are aware of the future. Indeed, a schedule that contains multiple transmissions by node can be converted to a schedule where each node transmits only once, by keeping only the last transmission.

The results given in this chapter implies that the search for an optimal solution to aggregate data in a wireless sensor network is limited by the available computation power and knowledge. In order to perform data aggregation with a good energy-consumption/delay trade-off, one must allow a greater delay or relax the optimal energy consumption assumption. Indeed, with the latter assumption, the problem may not even be solvable when the nodes have no knowledge of the future evolution of the network, as we will see in the next chapter.

Chapter 6

# Distributed Online Data Aggregation

**Contents**



In this chapter we study the online and distributed solutions of the data aggregation problem, in dynamic networks. As we said in chapter 4, we consider here that no collision occurs and that an adversary chooses the sequence $I = (I_t)_{t\in\mathbb{N}}$ that models the pairwise interactions (or simply interactions) between nodes over the time.

## 6.1 Oblivious and Online Adaptive Adversaries

In this section we give several impossibility results when nodes have no knowledge, and then show several results depending on the amount of knowledge. We choose to limit our study to some specific knowledge, but one can be interested in studying the possible solutions for different kinds of knowledge.

### 6.1.1 Impossibility Results When Nodes Have no Knowledge

**Theorem 6.1**

*For every algorithm $A \in \mathcal{D}_{\mathsf{ODA}}$, there exists an adaptive online adversary generating a sequence of interactions $I$ such that $cost_A(I) = \infty$ (the cost function is defined in chapter 4).*

**Proof:** Let $I$ the sequence of interactions between 3 nodes $a$, $b$, and the sink $s$, defined as follow. $I_0 = \{a, b\}$. If $a$ transmits, then for every $i \in \mathbb{N}$, $I_{2i+1} = \{a, s\}$ and

$I_{2i+2} = \{a, b\}$ so that $b$ will never be able to transmit. Symmetrically if $b$ transmits the same thing happens. If no node transmits, then $I_1 = \{b, s\}$. If $b$ transmits, then $I_{2i+2} = \{a, b\}$ and $I_{2i+3} = \{b, s\}$ so that $a$ will never be able to transmit. Otherwise $I_2 = \{a, b\}$ and continue as in the first time. $A$ never terminates, and a convergecast is always possible, so that $cost_A(I) = \infty$. ■

In the case of a deterministic algorithm, the previous theorem is true even with an oblivious adversary. However, for a randomized algorithm, the problem is more complex. The following theorem states the impossibility result for oblivious randomized algorithm, leaving the case of non-oblivious randomized algorithms against oblivious adversary as an open question.

**Theorem 6.2**

*For every randomized algorithm $A \in \mathcal{D}^{\emptyset}_{\mathsf{ODA}}$, there exists an oblivious adversary generating a sequence of interactions $I$ such that $cost_A(I) = \infty$ with high probability*[2].

**Proof:** Let $V = \{s, u_0, \ldots, u_{n-2}\}$. In the sequel, indexes are modulo $n-1$ *i.e.*, $\forall i, j \geq 0$, $u_i = u_j$ with $i \equiv j \mod n-1$. Let $I^\infty$ defined by, for all $i \in \mathbb{N}$, $I^\infty_i = \{u_i, s\}$. Let $I^l$ be the finite sequence, prefix of length $l > 0$ of $I^\infty$. For every $l > 0$, the adversary can compute the probability $P_l$ that no node transmits its data when executing $A$ on $I^l$. $(P_l)_{l>0}$ is a non-increasing sequence, it converges to a limit $\mathcal{P} \geq 0$. For a given $l$, if $P_l \geq 1/n$, there is at least two nodes whose probability not to transmit when executing $A$ on $I^l$ is at least $n^{-\frac{1}{n-2}} = 1 - O\left(\frac{1}{\sqrt{n}}\right)$. To prove this, we can see the probability $P_l$ as the product of $n-1$ probabilities $p_0$, $p_1$, …, $p_{n-2}$ where $p_i$ is the probability that node $u_i$ does not transmit during $I^l$. Those events are independent since the algorithm is oblivious. Let $p_d \geq p_{d'}$ be the two greatest probabilities in $\{p_i\}_{0 \leq i \leq n-2}$, we have:

$$\left(\prod_{i=0}^{n-2} p_i \geq \frac{1}{n}\right) \Rightarrow \left(\sum_{i=0}^{n-2} \log(p_i) \geq \log\left(\frac{1}{n}\right)\right)$$
$$\Rightarrow \left((n-2)\log(p_{d'}) \geq \log\left(\frac{1}{n}\right)\right) \Rightarrow \left(p_{d'} \geq n^{-\frac{1}{n-2}}\right)$$

This implies that, if $\mathcal{P} \geq 1/n$, then $A$ does not terminate on the sequence $I^\infty$ with high probability.

Otherwise, let $l_0$ be the smallest index such that $P_{l_0} < 1/n$. So that with high probability, at least one node transmits when executing $A$ on $I^{l_0}$. Also, $P_{l_0-1} \geq 1/n$ so that the previous argument implies that there is at least two nodes $u_d$ and $u_{d'}$ whose probability to still have a data (after executing $A$ on $I^{l_0-1}$) is at least $n^{-\frac{1}{n-2}}$. If $l_0 = 0$ we can choose $\{u_d, u_{d'}\} = \{u_1, u_2\}$. We have $u_d \neq u_{l_0}$ or $u_{d'} \neq u_{l_0}$. Without loss of generality, we can suppose $u_d \neq u_{l_0}$, so that the probability that $u_d$ transmits is the same in $I^{l_0-1}$ and in $I^{l_0}$.

Now, $u_d$ is a node whose probability not to transmit when executing $A$ on $I^{l_0}$ is at least $n^{-\frac{1}{n-2}} = 1 - O\left(\frac{1}{\sqrt{n}}\right)$. Let $I'$ be the sequence of interactions defined as

2. An event $A$ occurs with high probability, when $n$ tends to infinity, if $P(A) > 1 - o\left(\frac{1}{\log(n)}\right)$.

follow:

$$\forall i \in [0, n-2] \setminus \{d-1\},\ I'_i = \{u_i, u_{i+1}\},\ I'_{d-1} = \{u_{d-1}, s\}$$

$I'$ is constructed such that $u_d$ (the node that has data with high probability) must send its data along a path that contains all the other nodes in order to reach the sink. But this path contains a node that does not have a data.

Let $I$ be the sequence of interaction starting with $I^{l_0}$ and followed by $I'$ infinitely often. We have shown that with high probability, after $l_0$ interactions, at least one node transmits its data and the node $u_d$ still has a data. The node that does not have data prevents the data owned by $u_d$ from reaching $s$. So that $A$ does not terminate, and since a convergecast is always possible, then $cost_A(I) = \infty$. ■

### 6.1.2 When Nodes Know The Underlying Graph

Let $\overline{G}$ be the underlying graph *i.e.*, $\overline{G} = (V, E)$ with

$$E = \{(u, v) \,|\, \exists t \in \mathbb{N},\ I_t = \{u, v\}\}\,.$$

The following results assume that the underlying graph is given initially to every node.

**Theorem 6.3**

*If $n \geq 4$, then there exists an online adaptive adversary that generates, for every algorithm $A \in \mathcal{D}_{\mathsf{ODA}}(\overline{G})$, a sequence of interactions $I$ such that $cost_A(I) = \infty$.*

**Proof:** $V = \{s, u_1, u_2, u_3\}$. We create a sequence of interactions with the underlying graph $\overline{G} = (V, \{(s, u_1), (u_1, u_2), (u_2, u_3), (u_3, s)\})$. We start with the following interactions:

$$(\{u_1, s\}, \{u_3, s\}, \{u_2, u_1\}, \{u_2, u_3\})\,. \tag{6.1}$$

If $u_2$ transmits to $u_1$ in $I_2$, then we repeat infinitely often the three following interactions:

$$(\{u_1, u_2\}, \{u_2, u_3\}, \{u_3, s\}, ...)\,.$$

Else, if $u_2$ transmits to $u_3$ in $I_3$, then we repeat infinitely often the three following interactions:

$$(\{u_3, u_2\}, \{u_2, u_1\}, \{u_1, s\}, ...)\,.$$

Otherwise, we repeat the four interactions (6.1), and apply the previous reasoning. Then, $A$ never terminates, and a convergecast is always possible, so that $cost_A(I) = \infty$. ■

**Theorem 6.4**

*If the interactions occurring at least once, occur infinity often, then there exists $A \in \mathcal{D}_{\mathsf{ODA}}(\overline{G})$ such that $cost_A(I) < \infty$ for every sequence of interactions $I$. However, $cost_A(I)$ is unbounded.*

**Proof:** Nodes can compute a spanning tree $T$ rooted at $s$ (they compute the same tree, using nodes identifiers). Then, each node waits to receive the data from its

children and then transmits to its parent as soon as possible. All transmissions are done in finite time because each edge of the spanning tree appears infinitely often. However, when $\overline{G}$ is not a tree, there exists another spanning tree $T'$. Let $e$ be an edge of $T$ that is not in $T'$. By repeated interactions of edges of $T'$, an arbitrary amount of convergecasts can be performed while a node is waiting for sending data to its parent through $e$ in execution of $A$. ■

**Theorem 6.5**

*If $\overline{G}$ is a tree, there exists $A \in \mathcal{D}_{\mathsf{ODA}}(\overline{G})$ that is optimal.*

**Proof:** Each node waits to receive the data from its children, then transmits to its parent as soon as possible. ■

### 6.1.3 If Nodes Know Their Own Future

For a node $u \in V$, $u.future$ denotes the future of $u$ *i.e.*, the sequence of interactions involving $u$, with their times of occurrences. In this case, according to the model, two interacting nodes exchange their future and non-oblivious nodes can store it. This may seem in contradiction with the motivation of the problem that aims to reduce the number of transmissions. However, it is possible that the data must be sent only once for reasons not related to energy (such as data that cannot be duplicated, tokens, etc.). That is why we consider this case, for the sake of completeness, even if oblivious algorithms should be favored.

**Theorem 6.6**

*There exists $A \in \mathcal{D}_{\mathsf{ODA}}(future)$ such that $cost_A(I) \leq n$ for every sequence of interactions $I$.*

**Proof:** One can show that the duration of $n-1$ successive convergecasts is sufficient to perform a broadcast from any source. So every node broadcast its future to the other nodes. After that, all the nodes are aware of the future of every nodes and can compute the optimal data aggregation schedule. So that it takes only one convergecast to aggregate the data of the whole network. In total, $n$ successive convergecasts are sufficient. ■

## 6.2 Randomized Adversary

The randomized adversary constructs the sequence of interactions by picking a couple of nodes among all possible couples, uniformly at random. Thus, the underlying graph is a complete graph of $n$ nodes (including the sink) and every interaction occurs with the same probability $p = \frac{2}{n(n-1)}$.

In this section, the complexity is computed on average (because the adversary is randomized) and no more "in the worst case" as previously. In this case, considering the number of interactions is sufficient to represent the complexity of an algorithm.

We see in Theorem 6.8 that an offline algorithm terminates in $\Theta(n\log(n))$ interactions w.h.p. This bound gives a way to convert the complexity in term of number of interactions to a cost. Indeed, if an algorithm $\mathcal{A}$ terminates in $O(n^2)$ interactions, then it performance is $O(n/\log(n))$ times worse than the offline algorithm and $cost_A(I) = O(n/\log(n))$ for a randomly generated sequence of interactions $I$. For the sake of simplicity, in the remaining of the section, we give the complexity in terms of number of interactions.

Since an interaction does not depend on previous interactions, the algorithms we propose here are oblivious i.e., they do not modify the memory of the nodes. In more details, the output of our algorithms depends only on the current interaction and on the information available in the node.

First, we introduce three oblivious DODA algorithms. For the sake of simplicity, we assume that the output is ignored if the interacting nodes do not both have data. Also, to break symmetry, we suppose the nodes that interact are given as input ordered by their identifiers. The last algorithm (Waiting Greedy) uses the $meetTime$ information defined in chapter 4.

— Waiting ($\mathcal{W} \in \mathcal{D}^{\emptyset}_{\mathsf{ODA}}$): A node transmits only when it is connected to the sink $s$:

$$\mathcal{W} : (u_1, u_2, t) = \begin{cases} u_i & \text{if } u_i.isSink \\ \perp & \text{otherwise} \end{cases}$$

— Gathering ($\mathcal{GA} \in \mathcal{D}^{\emptyset}_{\mathsf{ODA}}$): A node transmits its data when it is connected to the sink $s$ or to a node having data.:

$$\mathcal{GA} : (u_1, u_2, t) = \begin{cases} u_i & \text{if } u_i.is.Sink \\ u_1 & \text{otherwise} \end{cases}$$

— Waiting Greedy with parameter $\tau \in \mathbb{N}$ ($\mathcal{WG}_\tau \in \mathcal{D}^{\emptyset}_{\mathsf{ODA}}(meetTime)$): The node with the greatest meet time transmits, if its meet time is greater than $\tau$:

$$m_1 = u_1.meetTime(t)$$
$$m_2 = u_2.meetTime(t)$$

$$\mathcal{WG}_\tau : (u_1, u_2, t) = \begin{cases} u_1 & \text{if } m_1 < m_2 \wedge \tau < m_2 \\ u_2 & \text{if } m_1 > m_2 \wedge \tau < m_1 \\ \perp & \text{otherwise} \end{cases}$$

### 6.2.1 Lower Bounds

We show a lower bound $\Omega(n^2)$ on the number of interactions required for DODA against the randomized adversary. The lower bound holds for all algorithms (including randomized ones) that do not have knowledge about future of the evolving network. The lower bound matches the upper bound of the *Gathering* algorithm given in the next subsection. This implies that this bound is tight.

**Theorem 6.7**

*The expected number of interactions required for DODA is $\Omega(n^2)$.*

**Proof:** We show that the last data transmission requires $\Omega(n^2)$ interactions in expectation.

We consider any (randomized) algorithm $\mathcal{A}$ and its execution for DODA. Before the last transmission (from some node, say $v$, to the sink $s$), only $v$ has data except for $s$.

The probability that $v$ and $s$ interacts in the next interaction is $\frac{2}{n(n-1)}$. Thus, the expected number $EI$ of interactions required for $v$ to transmit to $s$ is:

$$EI = \frac{n(n-1)}{2}$$

So that the whole aggregation requires at least $EI = \Omega(n^2)$. ■

We also give a tight bound for algorithms that know the full sequence of interactions.

**Theorem 6.8**

*The best algorithm in $\mathcal{D}^{\emptyset}_{\mathsf{ODA}}$(full knowledge) terminates in $\Theta(n \log n)$ interactions, in expectation and with high probability.*

**Proof:** First, we show that the expected number of interactions of a broadcast algorithm is $\Theta(n \log n)$. The first data transmission occurs when the source node (say $v_0$) interacts with another node. The probability of occurrence of the first data transmission is $\frac{2(n-1)}{n(n-1)}$. After the $(i-1)$-th data transmission, $i$ nodes (say $V_{i-1} = \{v_0, v_1, \ldots, v_{i-1}\}$) have the data and the $i$-th data transmission occurs when a node in $V_{i-1}$ interacts with a node not in $V_{i-1}$. This happens with probability $\frac{2i(n-i)}{n(n-1)}$.

Thus, if $X$ is the number of interactions required to perform a broadcast, then we have:

$$\begin{aligned} E(X) &= \sum_{i=1}^{n-1} \frac{n(n-1)}{2i(n-i)} = \frac{n(n-1)}{2} \sum_{i=1}^{n-1} \frac{1}{i(n-i)} \\ &= \frac{n(n-1)}{2n} \sum_{i=1}^{n-1} (\frac{1}{i} + \frac{1}{n-i}) \\ &= (n-1) \sum_{i=1}^{n-1} \frac{1}{i} \in \Theta(n \log n). \end{aligned}$$

And the variance is

$$\begin{aligned} Var(X) &= \sum_{i=1}^{n-1} \left(1 - \frac{2i(n-i)}{n(n-1)}\right) / \left(\frac{2i(n-i)}{n(n-1)}\right)^2 \\ &= n(n-1) \sum_{i=1}^{n-1} \frac{n(n-1) - 2i(n-i)}{(2i(n-i))^2} \\ &= O\left(n^4 \sum_{i=1}^{\lfloor n/2 \rfloor - 1} \left(\frac{1}{i(n-i)}\right)^2\right) \end{aligned}$$

The last sum is obtained from the previous one by observing that it is symmetric with respect to the index $i = n/2$, and the removed elements ($i = \lfloor n/2 \rfloor$ and possibly $i = \lceil n/2 \rceil$) are negligible. We define $f : x \mapsto \frac{1}{x^2(n-x)^2}$. Since $f$ is increasing between 1 and $n/2$, we have

$$\begin{aligned} \sum_{i=1}^{\lfloor n/2 \rfloor - 1} f(i) &\leq \int_1^{n/2} f(x)dx \\ &= \frac{\frac{(n-2)n}{n-1} + 2\log(n-1)}{n^3} = O\left(\frac{1}{n^2}\right) \end{aligned}$$

So that the variance is in $O(n^2)$. Using the Chebyshev's inequality, we have

$$P(|X - E(X)| > n\log(n)) = O\left(\frac{1}{\log^2(n)}\right)$$

Therefore, a sequence of $\Theta(n\log(n))$ interactions is sufficient to perform a broadcast with high probability. By reversing the order of the interactions in the sequence of interactions, this implies that a sequence of $\Theta(n\log(n))$ interactions is also sufficient to perform a convergecast with the same probability. Aggregating data along the convergecast tree gives a valid data aggregation schedule. ■

**Corollary 6.1**

*The best algorithm in $\mathcal{D}_{\mathsf{ODA}}(future)$ terminates in $\Theta(n \log n)$ interactions, in expectation and with high probability.*

**Proof:** If each node starts with its own future, $O(n\log(n))$ interactions are sufficient to retrieve with high probability the future of the whole network. Then $O(n\log(n))$ interactions are sufficient to aggregate all the data with the full knowledge. ■

### 6.2.2 Algorithm Performance Without Knowledge

**Theorem 6.9**

*The expected number of interactions the Waiting requires to terminate is $O(n^2 \log n)$.*

*The expected number of interactions the Gathering requires to terminate is $O(n^2)$.*

**Proof:** In the *W*aiting algorithm, data is sent to the sink when a node with data is connected to the sink. We denote by $X_W$ the random variable that equals the number of interactions for the algorithm Waiting to terminate. The probability of occurrence of the first data transmission is $\frac{2(n-1)}{n(n-1)}$. The probability of occurrence of the $i$-th data transmission after the $(i-1)$-th data transmission is $\frac{2(n-i)}{n(n-1)}$. Thus, the expected number of interactions required for DODA is

$$\begin{aligned} E(X_W) &= \sum_{i=1}^{n-1} \frac{n(n-1)}{2(n-i)} \\ &= \frac{n(n-1)}{2} \sum_{i=1}^{n-1} \frac{1}{i} \in O(n^2 \log n) \end{aligned}$$

Since those events are independent, we also have that the variance of the number of interactions required for DODA is

$$\begin{aligned} Var(X_W) &= \sum_{i=1}^{n-1} \frac{n(n-1)-2i}{n(n-1)} \times \frac{(n(n-1))^2}{4i^2} \\ &= \sum_{i=1}^{n-1} \frac{n^2(n-1)^2 - 2in(n-1)}{4i^2} \\ &\sim_{+\infty} \sum_{i=1}^{n-1} \frac{n^4}{4i^2} \qquad \sim_{+\infty} \frac{n^4\pi^2}{24} \end{aligned}$$

Using the Chebyshev's inequality, we have

$$\begin{aligned} P(|X_W - E(X_W)| > n^2 \log(n)) &= O\left(\frac{n^4\pi^2}{24n^4 \log^2(n)}\right) \\ &= O\left(\frac{1}{\log^2(n)}\right) \end{aligned}$$

Therefore, algorithm Waiting terminates after $O(n^2 \log(n))$ interactions with probability greater than $1 - 1/log^2(n)$.

In the Gathering algorithm, a data is sent when a node with the data is connected to the sink or another node with data. We denote by $X_G$ the random variable that equals the number of interactions for the algorithm Gathering to terminate. Notice that the total number of data transmissions required to terminate is exactly $n-1$. The probability of occurrence of the first data transmission is $\frac{n(n-1)}{n(n-1)} = 1$. The probability of occurrence of the $i$-th data transmission after the $(i-1)$-th data transmission is $\frac{(n-i+1)(n-i)}{n(n-1)}$. Thus, the expected number of interactions required to terminate is

$$\begin{aligned} E(X_G) &= \sum_{i=1}^{n-1} \frac{n(n-1)}{(n-i+1)(n-i)} \\ &= n(n-1) \sum_{i=1}^{n-1} \frac{1}{i(i+1)} \in O(n^2) \end{aligned}$$

■

**Corollary 6.2**

*Algorithm Gathering is optimal in* $\mathcal{D}_{\mathsf{ODA}}$.

### 6.2.3 Algorithm Performance With *meetTime*

In this subsection we study the performance of our algorithm Waiting Greedy, find the optimal value of the parameter $\tau$ and prove that this is the best possible algorithm with only the *meetTime* information (even if nodes have unbounded memory). We begin by a lemma to find how many interactions are needed to have a given number of nodes interacting with the sink.

**Lemma 6.1**

*If $f$ is a function such that $f(n) = \omega(\log(n))$ and $f(n) = o(n)$ then, in $nf(n)$ interactions, $\Theta(f(n))$ nodes interact with the sink with high probability.*

**Proof:** The probability of the $i$-th interaction between the sink and a node that has data, after $i-1$ such interactions, is $\frac{2(n-i)}{n(n-1)}$. Let $X$ be the number of interactions needed for the sink to meet $f(n)$ different nodes. We have:

$$\begin{aligned} E(X) &= \sum_{i=1}^{f(n)} \frac{n(n-1)}{2(n-i)} \\ &= \frac{n(n-1)}{2}(H(n-1) - H(n-f(n))) \\ &\sim \frac{n^2}{2}\left(-\log\left(1-\frac{f(n)}{n}\right) + o(1)\right) \\ &\sim \frac{n^2}{2}\frac{f(n)}{n} \sim \frac{f(n)n}{2} \end{aligned}$$

and the variance is

$$\begin{aligned} Var(X) &= \sum_{i=1}^{f(n)} \left(1 - \frac{2(n-i)}{n(n-1)}\right) / \left(\frac{2(n-i)}{n(n-1)}\right)^2 \\ &\sim \sum_{i=1}^{f(n)} \frac{n^4}{4n^2} \sim \frac{n^2}{4} f(n) \end{aligned}$$

Using the Chebyshev's inequality, we have

$$\begin{aligned} P(|X - E(X)| > nf(n)) &= O\left(\frac{n^2 f(n)}{4n^2 f(n)^2}\right) \\ &= O\left(\frac{1}{f(n)}\right) \end{aligned}$$

So that $X = \Theta(nf(n))$ w.h.p. if $1/f(n) = o(1/\log(n))$ or equivalently, if $f(n) = \omega(\log(n))$. ■

Now we can state our theorem about the performance of Waiting Greedy depending on the parameter $\tau$.

**Theorem 6.10**

*Let $f$ be a function such that $f(n) = o(n)$ and $f(n) = \omega(\log(n))$. The algorithm Waiting Greedy with $\tau = \Theta\left(\max\left(nf(n), n^2\log(n)/f(n)\right)\right)$ terminates in $\tau$ interactions with high probability.*

**Proof:** To have an upper bound on the number of interactions needed by Waiting Greedy to terminate, we decompose the execution in two phases, one between time 0 and a time $t_1$ and the other between time $t_1$ and a time $t_2 = \tau$. In the last phase, a set of nodes $L \subset V$ interacts at least once directly with the sink. Nodes in $L$ do not transmit to anyone in the first phase by definition of the algorithm (they have a meetTime smaller than $\tau$). Nodes in $L$ help the other nodes (in $L^c = V \backslash L$) to transmit their data in the first phase. Maybe nodes in $L^c$ can transmit to $L$ in the second phase, but we do not take this into account, that is why it is an upper bound.

If a node $u$ in $L^c$ interacts with a node in $L$ in the first phase, either it transmits its data, otherwise (by definition of the algorithm) it has a meetTime smaller than $\tau$ (and smaller than $t_1$ because it is not in $L$). In every case, a node in $L^c$ that meets a node in $L$ in the first phase, transmits its data. To prove the theorem i.e., in order for the algorithm to terminate before $\tau$ with high probability, we prove two claims: (a) the number of nodes in $L$ is $f(n)$ with high probability if $t_2 - t_1 = nf(n)$ and (b) all nodes in $L^c$ interact with a node in $L$ with high probability if $t_1 = \Theta(n^2\log(n)/f(n))$. The first claim is implied by lemma 6.1. Now we prove the second claim.

Let $X$ be the number of interactions required for the nodes in $L^c$ to meet a node in $L$. The probability of the $i$-th interaction between a node in $L^c$ (with a data) and a node in $L$, after $i-1$ such interactions already occurred, is $2f(n)(n - f(n) - i)/n(n-1)$.

It follows that the expected number of interactions to aggregate all the data of $L^c$ is

$$\begin{aligned} E(X) &= \sum_{i=1}^{n-f(n)-1} \frac{n(n-1)}{2f(n)(n-f(n)-i)} \\ &= \frac{n(n-1)}{2f(n)} \sum_{i=1}^{n-f(n)-1} \frac{1}{n-f(n)-i} \\ &\sim_{+\infty} \frac{n^2}{2f(n)} \log(n - f(n)) \\ &= \frac{n^2}{2f(n)} \log(n(1 - f(n)/n)) \sim \frac{n^2\log(n)}{2f(n)} \end{aligned}$$

And the variance is

$$Var(X) = \sum_{i=1}^{n-f(n)-1} \frac{\left(1 - \frac{2f(n)(n-f(n)-i)}{n(n-1)}\right)}{\left(\frac{2f(n)(n-f(n)-i)}{n(n-1)}\right)^2}$$
$$\sim \sum_{i=1}^{n-f(n)-1} \frac{n^4}{4f(n)^2n^2} \sim \frac{n^3}{4f(n)^2}$$

Using the Chebyshev's inequality, we have

$$P\left(|X - E(X)| > \frac{n^2\log(n)}{2f(n)}\right) = O\left(\frac{1}{n\log^2(n)}\right)$$

Thus $X = O\left(\frac{n^2\log(n)}{f(n)}\right)$ with high probability. ■

**Corollary 6.3**

*The algorithm Waiting Greedy, with $\tau = \Theta(n^{3/2}\sqrt{\log(n)})$, terminates in $\tau$ interactions with high probability.*

**Proof:** In the last theorem, the bound $O\left(\max\left(nf(n), n^2\log(n)/f(n)\right)\right)$ is minimized by the function $f : n \mapsto \sqrt{n\log(n)}$. ■

**Theorem 6.11**

*Waiting Greedy with $\tau = \Theta(n^{3/2}\sqrt{\log(n)})$ is optimal in $\mathcal{D}_{\mathsf{ODA}}(meetTime)$.*

**Proof:** For the sake of contradiction, we suppose the existence of an algorithm $A \in \mathcal{D}_{\mathsf{ODA}}(meetTime)$ that terminates in $T(n)$ interactions with high probability, with $T(n) = o\left(n^{3/2}\sqrt{\log(n)}\right)$. Without loss of generality we can suppose that $A$ does nothing after $T(n)$ interactions. Indeed, the algorithm $A'$ that executes $A$ up to $T(n)$ and does nothing afterward has the same upper bound (since the bound holds with high probability).

Let $L$ be the set of nodes that interact directly with the sink during the first $T(n)$ interactions. Let $L^c$ be its complementary in $V\backslash\{s\}$. We know from lemma 6.1 that $\#L = O(T(n)/n) = o\left(\sqrt{n\log(n)}\right)$ with high probability.

We can show that $T(n)$ interactions are not sufficient for all the nodes in $L^c$ to interacts with nodes in $L$. If nodes in $L^c$ want to send their data to the sink, some data must be aggregated among nodes in $L^c$, then the remaining nodes in $L^c$ that still own data must interact with a node in $L$ before $T(n)$ interactions (this is not even sufficient to perform the DODA, but is enough to reach a contradiction).

When two nodes in $L^c$ interact, their meetTime (that are greater than $T(n)$) and the previous interactions are independent with the future interactions occurring before $T(n)$. This implies that when two nodes in $L^c$ interact, using this information to decide which node transmits is the same as choosing the sender randomly. From corollary 6.2, this implies that the optimal algorithm to aggregate data in $L^c$ is the Gathering algorithm.

Now, we show that, even after the nodes in $L^c$ use the Gathering algorithm, there is with high probability at least one node in $L^c$ that still owns data and that does not interact with any node in $L$. This node prevents the termination of the algorithm before $T(n)$ interactions with high probability, which is a contradiction.

Formally, we have the following lemmas.

**Lemma 6.2**

*Let $g(n)$ be the number of nodes in $L^c$. After using the Gathering algorithm during $T(n)$ interactions, the number of nodes in $L^c$ that still own data is in $\omega(\sqrt{n/\log(n)})$ with high probability.*

**Proof:** Let $X$ be the number of interactions needed for $R(n)$ nodes in $L^c$ to transmit their data. For the sake of contradiction, we suppose that

$$g(n) - R(n) = O(\sqrt{n/\log(n)}) = o(g(n)) \tag{6.2}$$

and show that $X$ is greater than $T(n)$ w.h.p. The probability of the $i$-th interaction between two nodes in $L^c$ that own data, after the $(i-1)$-th interaction already occurred, is $\frac{(g(n)-i)(g(n)-i-1)}{n(n-1)}$. Thus we have:

$$\begin{aligned}
E(X) &= \sum_{i=0}^{R(n)-1} \frac{n(n-1)}{(g(n)-i)(g(n)-i-1)} \\
&= n(n-1) \sum_{i=g(n)-R(n)+1}^{g(n)} \frac{1}{i(i-1)} \\
&= n(n-1)\left(\frac{1}{g(n)-R(n)+1} - \frac{1}{g(n)}\right) \\
&= n(n-1)\frac{R(n)}{g(n)(g(n)-R(n))}
\end{aligned}$$

From equation (6.2) we deduce that $g \sim R$ and we have:

$$E(X) \sim n^2 \frac{1}{g(n)-R(n)}$$

which implies

$$E(X) = \Omega\left(n^2\sqrt{\frac{\log(n)}{n}}\right) = \Omega\left(n^{3/2}\sqrt{\log(n)}\right).$$

As in the previous proofs, the expectation is reached with high probability. This contradicts the fact $T(n) = o(n^{3/2}\sqrt{\log(n)})$. ■

**Lemma 6.3**

*Let $H \subset L^c$ be the nodes in $L^c$ that still own data after the gathering. With high probability, $T(n)$ interactions are not sufficient for all the nodes in $H$ to interact with nodes in $L$.*

**Proof:** We know from the previous lemma that the number of nodes in $H$

is $h(n) = \omega(\sqrt{n/\log(n)})$. Let $X$ be the random variable that equals the number of interactions needed for the nodes in $H$ to interact with the nodes in $L$. We show that $X$ is in $\omega(n^{3/2}\sqrt{\log(n)})$ with high probability. Indeed, the probability of the $i$-th interaction between a node in $H$ that owns data, after the $(i-1)$-th interaction already occurred, is $\frac{2f(n)(h(n)-i)}{n(n-1)}$, where $f(n) = \#L$. Thus we have:

$$\begin{aligned} E(X) &= \sum_{i=0}^{h(n)-1} \frac{n(n-1)}{2f(n)(h(n)-i)} \\ &= \frac{n(n-1)}{2f(n)} \sum_{i=1}^{h(n)} \frac{1}{i} \sim \frac{n^2}{2f(n)} \log(h(n)) \end{aligned}$$

But since $f(n) = o\left(\sqrt{n\log(n)}\right)$, we have

$$\begin{aligned} E(X) &= \omega\left(\frac{n^{3/2}}{\sqrt{\log(n)}} \log(h(n))\right) \\ &= \omega\left(\frac{n^{3/2}}{\sqrt{\log(n)}} \log(n/log(n))\right) \\ &= \omega\left(n^{3/2}\sqrt{\log(n)}\right) \end{aligned}$$

Again the bound holds with high probability. This implies that, with high probability, $T(n) = o(n^{3/2}\sqrt{\log(n)})$ interactions are not sufficient for all the nodes in $H$ to interact with nodes in $L$. ■

*End of the proof of theorem 6.11.* We have shown that $T(n)$ interactions are not sufficient for the nodes in $L^c$ to transmit their data (directly or indirectly) to the nodes in $L$. Indeed, we have shown that the nodes in $L^c$ can apply the gathering algorithm so that $\omega(\sqrt{n\log(n)})$ nodes in $L^c$ still own data with high probability. But, with high probability, one of the $\omega(\sqrt{n\log(n)})$ remaining nodes does not interact with a node in $L$ in $T(n)$ interactions. This implies that, with high probability, at least one node cannot send its data to the sink in $T(n)$ interactions and an algorithm $A$ with such a bound $T$ does not exist. ■

## 6.3 Concluding Remarks

We defined and investigated the complexity of the distributed online data aggregation problem in dynamic graphs where interactions are controlled by an adversary. We obtained various tight complexity results for different adversaries and node knowledge. First we show that, when nodes have no knowledge, an oblivious adversary can generate, for a deterministic DODA algorithm or a randomized oblivious DODA algorithm, a sequence of interactions for which this algorithm has an infinite cost (it does not terminate and the optimal offline algorithm terminates on every suffix of the sequence of interactions). By giving some knowledge to the nodes, such

as the eventual underlying graph, we can create an algorithm that terminates (but with unbounded cost) or an optimal algorithm if the underlying graph is a tree. When nodes know their entire future, there exists an algorithm with a cost of $n$.

Against a randomized adversary, we present two optimal algorithms. When nodes have no knowledge, the Gathering algorithm terminates in $O(n^2)$ interactions in expectation, which we show is optimal. When each node knows the time of occurrence of their next interaction with the sink, we show that the Waiting Greedy algorithm with parameter $\tau = \Theta(n^{3/2}\sqrt{\log(n)})$ terminates in $\tau$ interactions with high probability, which we show is optimal.

The results given in this chapter implies that the search for on optimal solution that works in the worst case may is not practical. Instead, we saw that data aggregation algorithm have good performance against a randomized adversary. In more details, this means that when the network evolves randomly (or pseudo-randomly) efficient algorithm should exists.

# Part III

# Lifetime Estimation of a Wireless Sensor Network

Chapter 7

# Introduction of Part III

> Why would you want more than machine language?
>
> John von Neumann

**Contents**



In this third part of this thesis, we focus on the evaluation of the lifetime of a WSN. Estimating the battery lifetime of a device is important to reduce the manufacturing and development phase duration and ensure sensor sustainability. Indeed, sensor device applications may require long lifetime (from a day to a year), especially in the medical sector, which makes the test phase complex. Also, this long lifetime requirement impacts the efficiency of existing simulators that aim to accurately simulate all behaviors of a battery: heavy computations are expected when using low-level mode simulations (such as the electrochemical model), as those do not consider the fact that wireless sensor devices spend most of the lifetime in sleep mode.

## 7.1 Problem

The problem we are considering here is: how to evaluate accurately and efficiently the energy consumption and the lifetime of a battery-powered sensor node. Ultimately, the goal is to evaluate the lifetime of a WSN. The definition of the lifetime of a network may depend on the application (*e.g.* , it can be the duration before the first node exits the network, on the duration before a given percentage of nodes exit the network), but usually depends on the lifetime of the sensor nodes in the network.

The goal is to create two models (an energy consumption model, and a battery model) and to implement these models in a module for the WSNet simulator [FCF07]. The energy consumption model defines how a sensor node consumes the energy. For instance, one can assume that only the energy consumed by the

transmissions is considered, or only the transmissions and receptions, or assume that all the components of the node consume energy. The battery model represents the state of the battery depending on the external factors such as the instantaneous current drawn by the sensor node over the time, the characteristic of the battery, or the temperature. In Chapter 8, we present our module called WiseBat.

In the literature, when it comes to compare the energy efficiency of protocols, it is common to just count the number of transmitted messages, or to use a simple energy consumption model with a linear (ideal) battery model. With the help of our module in WSNet simulator, we can benchmark different protocols to study their performance in a realistic environment. There exists a variety of problems where our model can be used. In this thesis, we propose a simulation campaign to benchmark broadcasting protocols that aim to use varying transmission power to reduce the energy consumption of sensor nodes.

When each node can change its transmission power (and so its transmission range), the goal of an energy efficient broadcast algorithm is, for a given source $s$, to assign a transmission range to each sensor node so that the node $s$ can broadcast a message to all the other nodes (there is a path from $s$ to all the other nodes) and the overall energy consumed by all the nodes is minimized. This problem, called the *minimum energy broadcast problem*, has been defined by Cagalj *et al.* [ČHE02] as follows:

**Definition 7.1 (*The minimum energy broadcast problem*)**

*Given a graph $G = (V, E)$, where each node $i \in V$ is assigned a variable node power $p_i^v$, we assign a link cost $c_{ij} : E \to \mathbb{R}_+$ that is equal to the minimum transmission power necessary to maintain link $(i, j)$.*

*Considering a source node $v$ that wants to broadcast a message, the minimum energy broadcast problem consists of finding the power assignment vector $P = [p_1^v, p_2^v \dots p_n^v]$ such that it induces the directed graph $G = (V, E')$, where $E' = \{(i, j) \in E : c_{ij} \leq p_i^v\}$, in which there is a path from $v$ to any node of $V$ (all nodes being covered), and such that*

$$\sum_{i \in V} p_i^v \qquad \textit{is minimal}$$

In the geometric version of the problem *i.e.*, in a WSN, the cost of a link $(i, j)$, which is the minimum transmission power needed by node $i$ to reach node $j$, equals to $d_{i,j}^{\alpha}$ where $d_{i,j}$ is the distance between $i$ and $j$ and $\alpha$ is a real number between 2 and 5. As demonstrated by Cagalj *et al.* [ČHE02], the minimum energy broadcast is NP-hard. Different approaches exist to approximate this problem, all considering that the cost of a link is the square of the distance between the nodes (*i.e.*, $\alpha = 2$). In Chapter 9 we benchmark several approximate algorithms in a realistic environment using our energy consumption and battery models.

## 7.2 Related Work

### 7.2.1 Simulating a WSN

Modeling a WSN with a UDG is simple but can be far from being realistic, especially when it comes to evaluate the performance of a protocol in a real environment. If a protocol is made to avoid collisions, then a UDG is well-suited, but when a protocol assumes that collisions are handled by a MAC layer, things are different. In the latter case, to evaluate the performance of a protocol, it is common to assume that the MAC layer is ideal, which means, that if two nodes transmit simultaneously to a common neighbor, both messages are received simultaneously, ignoring the impact of the interference. However, in this case, many factors have to be considered, and it is simply not accurate to model the network with a simple UDG. Collisions are the result of interference, for which several models exist and depends in turn on the propagation model. The MAC layer has a leading effect on the results as well, especially on the energy consumption of nodes. That is why, a simulator with a complete protocol stack for each node is best suited when it comes to evaluate protocols in a WSN.

We start by presenting several wireless sensor network simulators. For a complete survey see for instance previous work by C.Singh *et al.* [SVT08], B.Musznicki *et al.* [MZ12], A.K. Dwivedi *et al.* [DV11] and H.Sundani *et al.* [SLD+11]. Then, we survey existing models to evaluate the energy consumption is WSNs simulators. Finally, we describe the related work about broadcasting protocols in WSNs.

#### 7.2.1.1 Generic Network Simulators

**NS2/3** NS2 and NS3 [ns297, HRFR06] are open-source discrete event network simulators. They are object-oriented written in C++ and easily extensible. Modules in NS2 are written in tcl. Their popularity and the big community using and improving it is an advantage. However, they are not optimized for wireless sensor networks and lack some customization about energy model, sensing hardware models, the packet format. Moreover, only a few low-energy MAC layers are already implemented.

**OPNET** OPNET [Mod03] is a commercial discrete event, object-oriented, network simulator. It is customizable (from the modeling of the hardware to the packet format). However, as NS2/3, it is not very scalable, and the number of protocols available is limited.

**OMNeT++** OMNeT++ [V+01] is a discrete event, component-based, simulator developed in C++. It is modular and comes with a simulation software to help traceability and debuggability of simulation models. It is scalable and offers many different protocols available. A useful feature of OMNeT++ is that it can be embedded in another application. This feature resulted in some simulators based on it (*e.g.* Castalia and MiXiM).

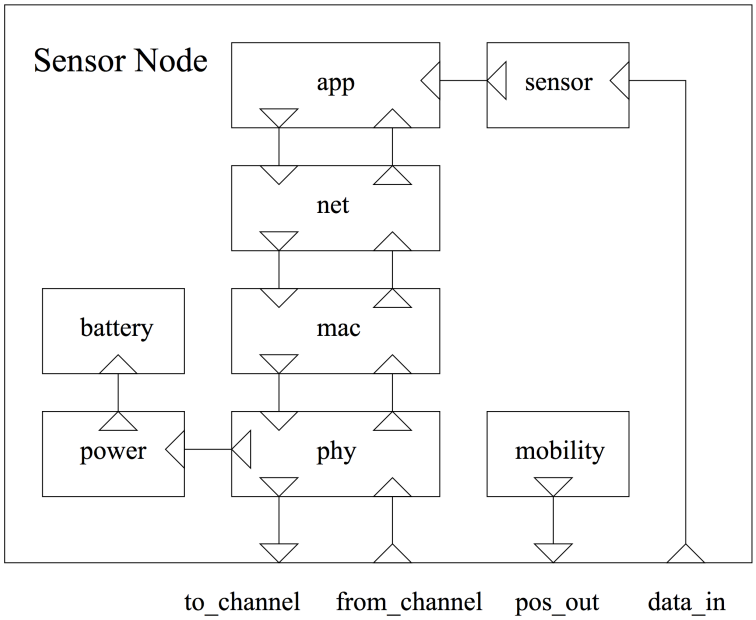



Figure 7.1 – The internal structure of a typical sensor node in SENSE simulator. From [CBP+05a]

#### 7.2.1.2 Emulators

**TOSSIM** TOSSIM [LLWC03] is an emulator as it runs actual application code. TOSSIM is designed specifically for TinyOS applications to be run on MICA Motes. However TOSSIM does not have an accurate physical model and is not easily extensible.

**Cooja** Cooja [ODE+06] is also an emulator, but designed to run ContikiOS application. It has the same drawback than TOSSIM.

#### 7.2.1.3 Wireless Sensor Oriented Simulators

**GloMoSim** GloMoSim [ZBG98] is a discrete event mobile wireless network simulator written in Parsec (an extension of C for parallel programming). Each layer has its own API to communicate with the surrounding layers and can be implemented as a module. However, GloMoSim is limited to IP networks and is not effective for low-power wireless sensor networks.

**SENSE** SENSE [CBP+05b] is a discrete event simulator developed in C++. It is component-based (see Figure 7.1), which makes it extensible and scalable. The main limitation is its small community and the small number of available protocols, especially for low-energy wireless sensor nodes.

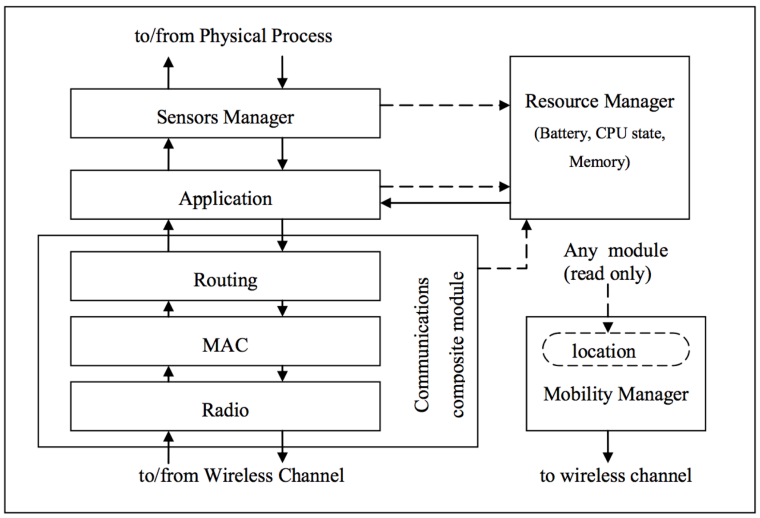



Figure 7.2 – The internal structure of a typical sensor node in Castalia simulator. From [Bou09]

**Castalia** Castalia [B+11] is a WSN and body area network (BAN) simulator built on top of OMNeT++ and proposes realistic radio and physical models, with a 802.15.4 MAC layer. The goal is to be able to test protocols in a realistic environment. However, the energy model is not easily extensible and the existing physical and MAC models are extensible but limited by the implementation of the simulator. Like SENSE simulator, we can see that the architecture of the Castalia (see Figure 7.2) is close to the architecture of a real sensor.

**MiXiM** MiXiM [MiX03] is built on top of OMNeT++ and is a combination of several OMNeT++ frameworks that models the lower layers of the protocol stack, offers detailed models of radio wave propagation and interference estimation. The main limitation is the small number of MAC layers for low power wireless sensor nodes available and the lack of documentation to implement your own modules.

**WSNet Simulator** In this thesis we use WSNet simulator [FCF07] to perform all our simulations. WSNet is an event-driven simulator for wireless networks. It simulates nodes with a full protocol stack (application, routing protocols, mac protocols, radio interface, antenna) and their mobility inside a simulated environment and radio medium (propagation models, interference models, modulation functions, etc.). WSNet is fully extensible as node, environment and radio medium blocks are developed in independent modules that can be changed. The WSNet block architecture is presented in Figure 7.3.

WSNet is particularly attractive for studying wireless sensor networks as there exist modules to provide extreme scale simulation (up to 20 million nodes [AT10b]) and study the impact of faults and attacks [AT10a]. All modules are configured

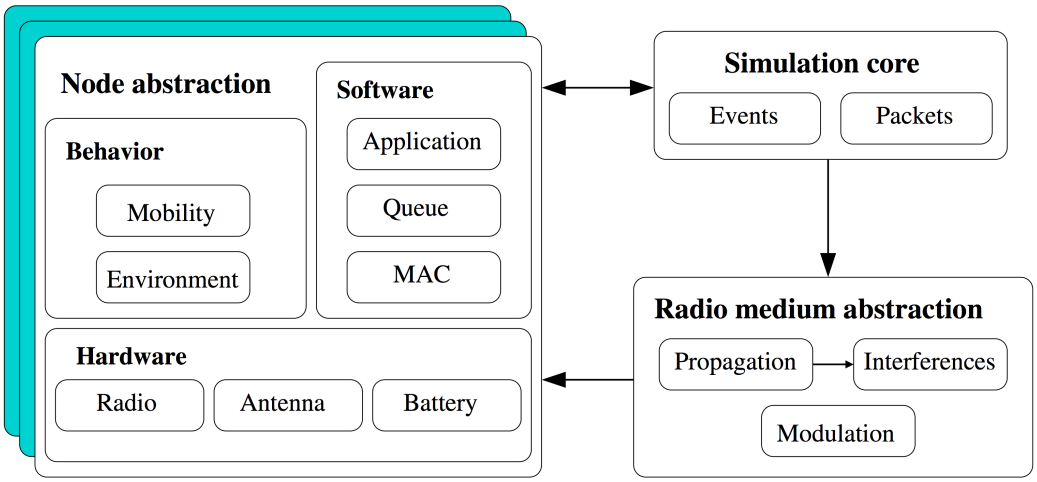



Figure 7.3 – Block architecture of WSNet. From [FCF07].

in a single XML file that represents a simulation. WSNet allows users to write their own module for the environment as well as for any node functionality. The default WSNet installation comes with several models of each type. For instance free space or log-normal shadowing among others for the radio propagation, greedy geometric or static for the routing layer. For our part, the default energy module models a linear battery that takes only the transceiver into account by default. In more details, the module receives an event at the end of a transmitted or received packet with the duration of the transmission. Then, it multiplies this duration by a number that represents the consumption of the transceiver in the TX or RX mode, and subtracts it from the capacity of the battery. A node is killed *i.e.*, leave the simulation, when the capacity of the battery is equal to or smaller than zero.

Each model can communicate with other models through the core of the simulator. A default connection exists between two consecutive modules in the protocol stack. for instance, when the application module wants to transmit a packet, it calls the function `TX` that calls the function `tx` in the lower level (usually the module handling the routing). Each module also has a generic entry point called `ioctl` that can be used by any other module, usually to perform cross-layer communications. For instance, if the application module wants to change the options of the radio module, it has to use this function. This feature is used by our module WiseBat presented in Chapter 8, to simulate the energy consumption and the battery accurately.

For this thesis, simulations are performed in the WSNet simulator [FCF07]. By default, the simulator uses the SINR model [BHCG08]. With this, we use the log-distance pathloss propagation model.

#### 7.2.1.4 Simulating Energy Consumption in WSNs

There exists a number of available simulators, designed for specific or generic networks (see Section 7.2.1). Even if network lifetime definition is very application-

dependent [DD09], it is determined using each single node's lifetime. Node's lifetime estimation requires both the node's power consumption and its battery to be modeled accurately [DDV+14]. Therefore, a variety of energy consumption and battery models have been studied [RVR03, RFG13] and implemented in existing simulators [WGMM08, FW10, PSS00] (often in the most popular ones). Despite a large choice of solutions to estimate the network lifetime, the majority of studies about wireless sensor networks does not use accurate power consumption and/or battery model. This is mostly because network lifetime is not always the first concern of these studies, and sometimes because they use a specific simulator that does not implement an accurate energy model (like WSNet simulator [CFF07]).

Usually, in the basic approach, the sensor node's consumption model is limited to an estimate of the energy consumed by each bit that has been transmitted by the radio. Together with this modeling, an ideal battery model is oftenly added (as it is the case for NS2/3 [ns297, HRFR06] or WSNet [CFF07], to cite a few). In the one hand, this model is very limited but, in the other hand, more accurate battery models can be themselves too complex and too time-consuming to be part of the simulation [RVR03]. As we demonstrate in the remainder of the paper, this approach is very limited and does not, in any real case, match the energy behavior of individual nodes.

Other solutions like SENSE [CBP+05b] or PowerTOSSIMz [PCC+08] model the power consumption of every component of the node together with a non-linear battery. Even if this approach seems to be more realistic, it does not take into account the battery's supply voltage variations (that are non-linear). This leads to another kind of inaccuracy. Indeed, we know that a node's life ends when its battery supply voltage falls under a given threshold (called the cut-off voltage), even if battery is not fully depleted. Depending on this cut-off voltage, the lifetime can be drastically reduced. That is why ignoring the voltage variation leads to an overestimation of the lifetime.

### 7.2.2 Benchmarking Energy-Centric Broadcast Protocols

We can distinguish two families of protocols that aim to be energy efficient by adjusting transmitting powers: *topology control oriented protocols*, and *broadcast oriented protocols*.

**Topology control oriented protocols.** A topology control oriented protocol assigns the transmission power for each node, independently of the source of the broadcast. The goal is to obtain a connected network with minimum total transmission power according to an energy consumption model. Once the radii are assigned, they are used for every broadcast from an arbitrary source. The problem of minimizing the total transmission power that keeps the network connected is known as *min assignment problem* and was considered by Kiroustis *et al.* [KKKP97]. Clementi *et al.* [CPS00] demonstrated that this problem is NP-hard. Most of these protocols require global knowledge of the entire graph to compute a Minimum Spanning Tree

(MST). Recently, localized protocols based on the Relative Neighborhood Graph (RNG) [Tou80], and a Local Minimum Spanning Tree (LMST) construction [LHS05] have been proposed (see [GY07] for a survey).

**Broadcast oriented protocols.** A broadcast oriented protocol has the same overall goal, but considers that the broadcast starts at a given node. Hence, the induced broadcast network does not have to be strongly connected, leading to possibly more efficient solutions. For instance, the last nodes that receive the message do not need to retransmit it *i.e.*, the algorithm may assign them a null transmission power. However, the source must be able to reach every node of the network. The problem remains difficult, as it has been proven [ČHE02] that the minimum energy broadcast problem is NP-complete.

In chapter 9 we study six broadcast oriented protocols (FLOOD, BIP, LBIP, DLBIP, RBOP and LBOP) defined in the literature [WNE00, ISR+08, CBB09, CISRS05, CISRS05].

**Flooding** Flooding is the simplest distributed protocol: when a node has a message to transmit, it transmits it with maximum power. Flooding is typically used for comparison with more elaborate energy-centric protocols, as it tries to maximize reachability without considering energy consumption.

**BIP (Broadcast Incremental Power)** [WNE00] BIP is the only centralized algorithm we consider in the paper. Although mostly Greedy-based, it is one of the most efficient algorithms, and is commonly used as a reference in the literature. It constructs a tree as follows: Initially, the broadcast tree contains only the source node; then, while there exists a node that is not in the tree, it computes the incremental minimum power needed to add a node to the tree, either by increasing the power transmission of a transmitting node in the tree, or by choosing a non-transmitting node in the tree to transmit.

**LBIP (Localized BIP)** [ISR+08] LBIP is the distributed version of BIP. Since acquiring the knowledge of the full network topology to apply BIP at every node would be to expensive (at least, energy-wise), the goal of the protocol is to discover only the 2-hop neighborhood and use the BIP algorithm on it. Each time a node receives a packet, in addition to the original message, it can contain a list of forwarder nodes. If the node is in this list, it computes the BIP tree on its 2-hop neighbors to choose the right transmission power and transmit the message with a list of nodes that must forward it, according to the BIP tree.

**DLBIP (Dynamic Localized BIP)** [CBB09] DLBIP is the energy-aware version of LBIP. It is dynamic in the sense that for the same source, two broadcast trees may be different. The broadcast is done in the same way as LBIP, but the BIP tree construct with the 2-hop neighbors take into account the remaining energy of the nodes to promote nodes with higher residual energy.

**RBOP (RNG Broadcast Oriented Protocol)** [CISRS05] RBOP is based on the RNG topology control protocol. A node that has a message sends it with

a transmission power such that all its neighbors in the RNG graph receive it. The protocol also contains some optimization to avoid transmission when a node knows that some of its neighbors in the RNG graph already received the message.

**LBOP (LMST Broadcast Oriented Protocol)** [CISRS05] LBOP applies the same scheme as RBOP but the RNG graph is replaced by the LMST one.

There exists other broadcast oriented algorithms that we do not consider in this paper. For instance the INOP (INside-Out Power adaptive approach) defined in [CBSP08], is very close to the other algorithms. It takes into account the 2-hop neighborhood, sorts them in terms of the power needed to cover them and then computes the optimal energy strategy starting from the closest neighbor to cover the next neighbor directly or indirectly. In the same paper, the authors evaluate their algorithm with a realistic network stack but used the 802.11 DCF MAC layer (which does not correspond to sensor network MAC layers), and an ideal per-packet energy consumption model.

All aforementioned works (besides INOP [CBSP08]) consider an ideal MAC layer where no interference occurs when two neighboring nodes transmit data within the same timeframe. Such an assumption is unrealistic as further discussed in Section 7.2.1. INOP [CBSP08] does consider a realistic network stack, but the stack is related to full-fledged computer networks, which is not energy-aware. Sensible network stacks for sensor nodes include ContikiMAC and 802.15.4, that we consider in the sequel.

Another oversimplification made by all previous work is even more problematic: the energy consumed by a node is supposed to be equal to the energy consumed by the radio during transmissions. In more details, the energy $E(u)$ consumed by a node $u$ for one transmission is given as a function depending on the radius $r(u)$ of the transmission (7.1) (which depends on the power transmission of the radio). The energy consumed by the protocol is then the sum $E$ of the energies consumed by all nodes (7.3). In order to compare several algorithms, we can consider the ratio $EER$ between the energy consumed by a protocol and the energy consumed by the flooding protocol (7.2-7.4), which always chooses the maximum transmission power.

$$E(u) = \begin{cases} r(u)^\alpha + c & \text{if } r(u) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.1}$$

$$E_{flooding} = n \times (R^\alpha + c) \tag{7.2}$$

$$E = \sum_{u \in V} E(u) \tag{7.3}$$

$$EER = \frac{E}{E_{flooding}} \times 100 \tag{7.4}$$

Where $\alpha, c \in \mathbb{R}^+$. Of course, real sensor nodes also consume energy when doing other tasks (reading sensor values, computing, receiving data, re-transmissions).

Also, low power batteries typically have non-linear behavior (their capacity may vary depending on the intensity of the drained current at a given time).

## 7.3 Contribution of Part III

**Energy Consumption and Battery Models for WSNet Simulator** We use an accurate existing battery model (presented in previous work [DDV+14]) that we optimized for wireless sensor networks simulation. We then link it with a power consumption model that takes all individual components' power consumption into account. The *WiSeBat* model refers to both the battery model and the power consumption model. We give a formal definition of these models. Then, we explain how to use them with user applications in the WSNet simulator. We validate our approach against a real sensor device. The collected measurements show that our model performs very well for duty-cycled scenarios, and give more realistic results than the default energy model of WSNet by several orders of magnitude.

Once validated, we used the WiSeBat approach to compare different protocol stacks in various scenarios. First, we compare X-MAC [BYAH06], ContikiMAC [Dun11], and 802.15.4 MAC [80203] beacon-enabled (simply denoted 802.15.4 in the sequel) in a single node scenario. As expected, ContikiMAC performs better than X-MAC, and 802.15.4 becomes useful if the node is in sleep mode for a long period of time. Secondly, we show that, as soon as an intermediate node is needed to forward messages from the leaf node to the coordinator, ContikiMAC outperforms 802.15.4 mac layer. Indeed, if a node using 802.15.4 needs to be able to forward a message, it has to regularly send beacons, which drains the battery faster than the short channel assessments needed by ContikiMAC. This is confirmed in the last scenario, where we compare ContikiMAC and the 802.15.4 mac performances in a dense network. WiSeBat energy model and the scenarios used in this paper are available online [wis14] and can be adapted to compare sensor lifetimes with user applications performing under user scenarios.

**Benchmarking Energy-Centric Broadcast Protocols** We consider previously proposed energy-centric broadcasting protocols for WSNs and evaluate them in realistic scenarios. We benchmark them using a simulator that includes a complete communication protocol stack, a realistic physical communication layer, and an accurate energy consumption model. We choose to perform our simulation with ContikiMAC and 802.15.4 MAC, that were both designed for small and energy constrained devices. In particular, in our settings, the MAC layer has to deal with possible collisions, and the energy consumption takes into account all components of the sensor device (including a non-linear battery behavior).

The results of our evaluation is as follows. First, we demonstrate that wireless interference significantly impacts the performance of broadcasting protocols in various ways. Indeed, the hierarchy of the broadcasting protocols (based on their performance in an ideal setting) is *not* preserved in the more realistic setting. Also,

we show that the MAC layer, also does not impact all broadcast protocols in the same way (some protocols perform better with ContikiMAC than with 802.15.4 MAC, while other do not). Quite surprisingly, it turns out that the very simple *flooding* protocol (used as a theoretical lower bound in previous work), is actually one the best distributed broadcasting protocols in our realistic environment. Our results show that considering only idealized settings in theoretical work does not give accurate performance hierarchies (including relative ones) in practical settings.

Chapter 8

# WiSeBat: Accurate Energy Benchmarking of Wireless Sensor Networks

**Contents**



In this chapter we present our model called WiSeBat, that includes an energy consumption model and a battery model, and implemented in WSNet simulator. First we describe the models used by WiSeBat and its usage in WSNet simulator. In Section 8.2 we evaluate its performance by comparing its results with measurement using a real sensor node. Finally we present in Section 8.3 several simulation done with WSNet simulator and WiSeBat to evaluate the lifetime of a WSN under different scenarios.

## 8.1 The WiSeBat Energy Model

In this section, we further describe our proposal for modeling energy-related issues in wireless sensor networks, named WiSeBat. The WiSeBat (Wireless Sensor Battery) model aims to realistically simulate the battery and the device power consumption with only a limited computation overhead. It offers the possibility to define custom components in addition to the wireless transceiver (typically sensors, LEDs, and micro-controllers) and their use by the application layer. The overview of the use of WiSeBat in a network simulator is as follows. First, the application code that is to be run by individual nodes registers all of its components during

the initialization phase and gives them a name and a consumption function. Then, the application may change the running mode of a registered component during the execution of the simulation. The WiSeBat model automatically computes the total current consumed by all registered components and updates the supply voltage and the residual capacity of the battery. A node is killed when the voltage of the battery falls bellow the cut-off voltage of the device, that is, the maximum cut-off voltage among all its components.

We begin with a formal definition of the model that uses existing techniques to simulate the behaviors of a battery (subsection 8.1.1), and then explain how it is implemented in the WSNet simulator and how to use it in a particular application (subsection 8.1.2).

### 8.1.1 Battery Modeling

With respect to the energy consumption model, the WiSeBat module "simply" takes all components into account. In this section we focus on the battery model. We progressively present the behaviors of a battery we choose to simulate, starting from the simple linear model, then the rate capacity effect, and finally the voltage variations. At the end of the section we detail an important issue of our model, which is to determine when the battery requires to be updated.

#### 8.1.1.1 Linear Battery Modeling

We start by considering an ideal battery model, but taking all the components in the device into account. The capacity of the battery is stored in the model and each time a component consumes a current $i$ during a time $t$, it subtracts $i \times t$ from the battery capacity. If a particular component $x$ (from a set of components $\mathcal{X}$) has spent $T_x(s)$ hours in a state $s$ (from the set of possible states $\mathcal{S}_x$ of component $x$), and consumes $i_x(s)$ mA in this state, then the battery capacity consumed $C$ (in mAh) by all supplied components is computed using the following equation:

$$C = \sum_{x \in \mathcal{X}} \sum_{s \in \mathcal{S}_x} i_x(s) T_x(s)$$

In this simple model, the voltage is assumed to remain constant in all cases, and there is no difference between a component that consumes 1mA during 1s and a component that consumes 10mA during 0.1s.

#### 8.1.1.2 The Rate Capacity Effect

The first behavior that is captured by our model is the rate-capacity effect. It refers to the fact that a lower discharge rate (*i.e.*, current) is more efficient than a higher one: more charge can be extracted from the battery before reaching a given cut-off voltage. In fact, the battery behaves like its capacity decreases when the discharge rate increases.

When the current is constant, the total amount of energy that can be drawn from the battery is given by a function $C_{eq}$, which is the *equivalent* capacity of the battery. The maximum capacity of the battery, called *nominal* capacity and denoted by $C_{\text{nomimal}}$, is measured with a low current draw called the *nominal current draw*. Battery data-sheets often contains $C_{\text{nomimal}}$ and few other values.

At time $t$, if a battery is subject to a current $i(t)$, the equivalent current is defined by [DDV+14]:

$$i_{eq}(i(t)) = \frac{C_{\text{nominal}}}{C_{eq}(i(t))} i(t) \tag{8.1}$$

so that, if the current is constant and equals to $i$, the battery is empty (or is not able to deliver a current) after a time $T$ such that $i \times T = C_{eq}(i)$. Using equivalent current is a simple way to approximate the rate capacity effect, but it requires the battery model to know the current draw of each component at any given time. The energy consumption cannot be distributed as in the linear model.

#### 8.1.1.3 Voltage Variation

In the case of a real battery, the supply voltage is not constant and depends on many factors. Here, our battery model takes into account two main factors: *(i)* the variation of the voltage depending on the residual capacity (when the battery is subject to the nominal current draw), and *(ii)* the internal resistance of the battery that linearly impacts the voltage depending on the current draw. Actually, the internal resistance of the battery also depends on the residual. We denote by $\text{Emf}(C)$ and $R(C)$ the voltage provided by the battery and the internal resistance when the residual capacity equals to $C$, respectively. Then, the supplied voltage $V$ of the battery with a residual capacity $C(t)$ and a current draw $i(t)$ is given by:

$$V = \text{Emf}(C(t)) - i(t)R(C(t))$$

This voltage directly affects the components in two ways. First, the current drawn by a particular component may depend on the voltage. Second, a component stops operating when the voltage is lower than its cut-off voltage. This aspect implies a cyclic relation of dependence between voltage and current draw.

To take into account the voltage, we denote by $i_x(V, s)$ the current draw of component $x$ in State $s$ with a supplied voltage $V$.

#### 8.1.1.4 WiSeBat Model

We now combine the used battery and the component's consumption models into a single module: WiSeBat. Suppose we have a set of components $\mathcal{X}$ and a function $S$ that returns the state of each component at any given time: for each component $x \in \mathcal{X}$, $S_x(t)$ denotes the state of $x$ at time $t$. We denote by $C(t)$ and $i(t)$ respectively the residual capacity of the battery in mAh and the current draw in mA at time $t$ in hours. Let $i_{eq}$ be the equivalent current draw in mA defined by equation (A.1). Let $t_1, t_2, \ldots$ be the times when the battery is updated. Our

module relies on the following relations:

$$C(t_i) = C(t_{i-1}) - i_{eq}(i(t_{i-1}))\,(t_i - t_{i-1}) \quad (8.2)$$

$$V(t_i) = E_{mf}\,(C(t_i)) - i(t_{i-1})R\,(C(t_i)) \quad (8.3)$$

$$i(t_i) = \sum_{x \in \mathcal{X}} i_x(V(t_i), S_x(t_i)) \quad (8.4)$$

We see in Equation (A.2) that the capacity decreases linearly with the time elapsed between two updates. Thus, if no update occurs in a long time, the voltage and the current draw can be obsolete. So the closer the updates are, the more accurate the approximation is.

In Equation (A.3), we update the voltage depending on the residual capacity and the previous current draw. So, if the current draw changes during its update in Equation (A.4), after a change of state of a component for instance, the voltage must be updated again.

#### 8.1.1.5 Updates Triggering

As we already mentioned, the more often the battery is updated, the more accurate its approximation is. An intuitive approach to obtain accurate results could be to trigger an update every $T$ time units. Even if this method has shown good results [DHG13], it is computationally expensive, especially for duty-cycled sensor networks where nodes are in sleep mode the vast majority of the time. Indeed, when the current draw is very low (which typically occurs when the device is in sleep mode), the residual of the battery is nearly constant and so is its voltage. In this case, close consecutive updates should be factored out.

In contrast, when the current draw is high, the residual may decrease in a short period of time and updates have to be triggered more often to keep the voltage up to date. Here, let $\varepsilon_V$ be a positive constant and $t_1$ be a time when an update occurs. Assume we want the error about the voltage to be smaller than $\varepsilon_V$. If a decrease of $\delta C$ in residual implies a change by $\varepsilon_V$ in the voltage, then another update must occur at time $t_2$ such that:

$$t_2 < t_1 + \frac{\delta C}{i_{eq}(i(t_1))}$$

Indeed, the difference between residuals at time $t_2$ and $t_1$ is smaller than $\delta C$ so that the voltage error is smaller than $\varepsilon_V$.

In our implementation, the two functions $E_{mf}$ and $R$ are piecewise linear (obtained by linear interpolation from the data-sheet) so that it is easy to compute the change in voltage depending on the change in the residual, for a given current draw.

Another important instant when an update must occur is when the running mode of a component changes. For instance, when the CPU wakes up. Indeed, such updates imply a change in the current draw. As a matter of fact, the voltage is computed with the previous current draw, and must be updated as soon as the current changes. However, as we saw, there exists a cyclic relation between voltage and

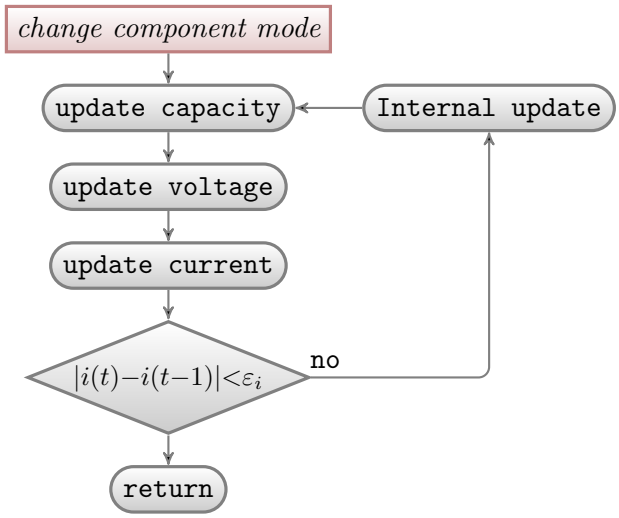



Figure 8.1 – The WiSeBat battery update scheme

current draw, and if the voltage is updated, the current draw of all the components that depend on it may change.

In our implementation, the voltage and the current draw are updated until the variation in the current draw is smaller than a given $\varepsilon_i > 0$. Since it is in theory possible that this process never converges, we delay the successive updates by a time $l > 0$ (for example the smallest possible duration available in the simulator, one nanosecond in WSNet), so that the simulation keeps going, and the residual capacity eventually decreases with every update. In practice, the convergence is really fast because on the one hand, when the voltage changes, the change in current is small, and in the other hand, because the voltage variation is linear with respect to the current draw variation.

Figure 8.1 summarizes the WiSeBat update scheme, triggered either from an internal callback or when a component's mode changes.

### 8.1.2 WiSeBat Usage

WiSeBat is used in two different places. The characteristics of the battery have to be set in the WSNet configuration file. Then, on the application code side, the consumption functions of used components must be provided, and the mode of individual components must be updated using WiSeBat control functions.

#### 8.1.2.1 WiSeBat Options

One entity of the model represents one battery. The characteristic of the battery is given to the model using the following options:

— `energy`: the nominal capacity $C_{\text{nominal}}$ of the battery (in milli Ampere Hour).

— `internal-resistance`: the internal resistance of the battery (in Ohm) depending on the residual. This parameter permits the construction of the function $R$.

— `cut-off-voltage`: the cut-off-voltage of your architecture (in Volt).

— `voltage-characteristic`: the characteristic of the voltage depending on the residual. This parameter permits the construction of the function $Emf$.

— `capacity-characteristic`: the capacity depending on the current draw. This parameter permits the construction of the function $C_{eq}$.

For the parameters that define a function (`internal-resistance`, `voltage-characteristic` and `capacity-characteristic`), the user can either give a number to define a constant function that equals this number or give a list of couples to define a piecewise linear function obtained by linear interpolation.

Other options can be given to nodes to define the level of logging. Also, the constants $\varepsilon_V$ and $\varepsilon_i$ defined in the previous section are set at the compilation time with default value $10^{-4}$.

#### 8.1.2.2 WiSeBat Consumption and Control Functions

For each component of a particular device, a consumption function has to be defined. The consumption function of a component $x$ takes as arguments the context of the battery (that contains the voltage $V$), the mode $s$ of the component, and returns the current (*i.e.*, discharge rate) used by the component, which corresponds to $i_x(V, s)$. An example of a consumption function is given in Figure 8.2. The returned values are taken from Table 8.4, and corresponds to the current drawn by the transceiver.

At WSNet initialization, each component has to be registered using Function `battery_register_component` that takes as arguments a component (*i.e.*, a name and a consumption function) and its initial running mode. Then, during the execution of the simulation, a component's mode can be changed using Function `battery_set_component_mode` that takes as arguments a registered component and its new mode. Each time this function is called, the battery is updated as explained in the previous section. Additional updates may be triggered, typically just after a change in the current draw, to take into account the change in voltage.

## 8.2 Evaluation

To evaluate the accuracy of our model, we used a real sensing device running an actual application code. The same application has been implemented in the WSNet simulator so that we can simulate the same Sense/Compute/Transmit cycles (see Figure 8.3) to compare measured and simulated lifetime (Subsection 8.2.1). We also measured the simulation overhead induced by our model, using a software profiler (Subsection 8.2.2). The same application is used for the simulations in Section 8.3.

```
double radio_consume(
            call_t* c,
            component_context_t context,
            component_mode_t mode)
{
    switch(mode){
    case MODE_TX:
        return 7.7;
    case MODE_RX:
        return 5.7;
    case MODE_IDLE:
        return 1.7;
    }
    return 0.0011;
}
```

Figure 8.2 – The transceiver consumption function that does not depend on the voltage since it is powered by a voltage regulator.

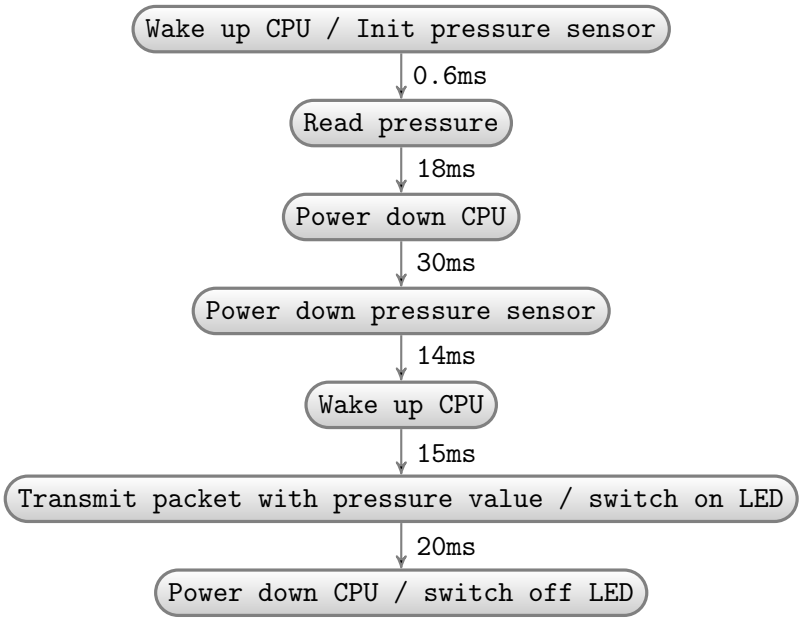



Figure 8.3 – Application layer: a single transmission cycle

| Component | Mode | Current |
|---|---|---|
| Micro controller<br>cut-off voltage: 1.65 | Run<br>Sleep | 10mA<br>2.4uA |
| Radio transceiver<br>cut-off voltage: 1.8 | Tx<br>Rx<br>Idle<br>Sleep | 7.7mA<br>5.7mA<br>1.7mA<br>1.1uA |
| Pressure sensor<br>cut-off voltage: 1.71 | Init<br>Read<br>Sleep | 42mA<br>5mA<br>0.5uA |
| Led | On<br>Off | 7.6mA<br>0mA |
| Power Manager<br>cut-off voltage: 2.4 | Efficiency: 95 - 99% | |

Figure 8.4 – Consumption characteristics of the main components of the actual device prototype

### 8.2.1 Comparison with Real life measurements

We start evaluating our WiSeBat model with a comparison between real and simulated sensor lifetime running the same application scenario. We used a prototype wireless sensor device that contains multiple sensors and has a rechargeable battery. The device runs under ContikiOS with the standard 802.15.4 MAC slotted beacon with duty cycle. The full device architecture is confidential but the main components with their electrical consumption obtained from the datasheets are listed in Table 8.4.

The cut-off voltage is the minimum voltage required for a component to operate properly. The maximum cut-off voltage among all the components becomes the cut-off voltage of the whole application. Here, the device stops working if the voltage is below 2.4 Volt, corresponding to the power manager cut-off.

The scenario consists of sensing the pressure and sending the measured value to a coordinator every second or every ten seconds, with the device in sleep mode between two transmissions. Figure 8.3 details a transmission cycle. The durations are obtained from measurements. We measured the lifetime of the real device and simulated the same scenarios with WiSeBat and with the default models of WSNet. The node lifetime simulated with each model is given in Figure 8.5 with a **log-scaled** y-axis. We observe that unlike other approaches that overestimate the lifetime of the node, WiSeBat underestimates it. This is important if the simulation guarantees are then used for actual device production: an overestimated lifetime leads to weaker batteries to be embedded on actual devices, and the network fails operating before expectations. On the contrary, an underestimated lifetime permits the real device to last at least as long as the simulation. The accuracy of the WiSeBat model

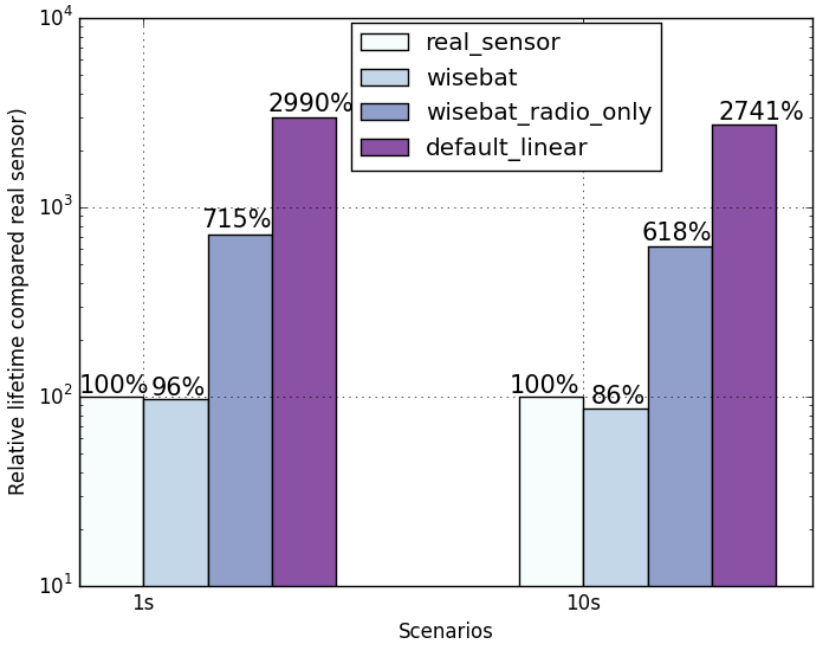



Figure 8.5 – Lifetime of a node simulated with WiSeBat models, WiSeBat battery model with only the radio transceiver consumption, and WSNet default models, relatively to a real sensor lifetime, in y-log-scale. WiSeBat underestimate the lifetime by $3 - 14\%$, and the default WSNet models overestimation is greater than 2600%.

outperforms the default models of WSNet by several orders of magnitude: WiSeBat underestimate the lifetime of the node by at most 14% (4%, respectively), while the default models of WSNet overestimate the lifetime by more than 2600% (2890%, respectively) in the 10-second scenario (in the 1 second scenario, respectively). This means that if we were using the simulation outputs to dimension batteries, WiSeBat driven batteries would last at least as expected and would be oversized by at most 14%, while "classical" WSNet driven batteries would last only $\frac{1}{27}$ of their expected lifetime, rendering the deployed sensor network practically useless.

We also simulate the scenario with the WiSeBat battery model taking only the transceiver into account, and the default energy consumption model that only takes the transceiver into account (wisebat_radio_only). We see that it improves the default battery model by a multiplicative factor of at least 5, but it remains quite far from reality, as it overestimates the lifetime by more than 500%. This demonstrates that both aspects of WiSeBat (battery and energy consumption models) play an important role in the global lifetime estimation.

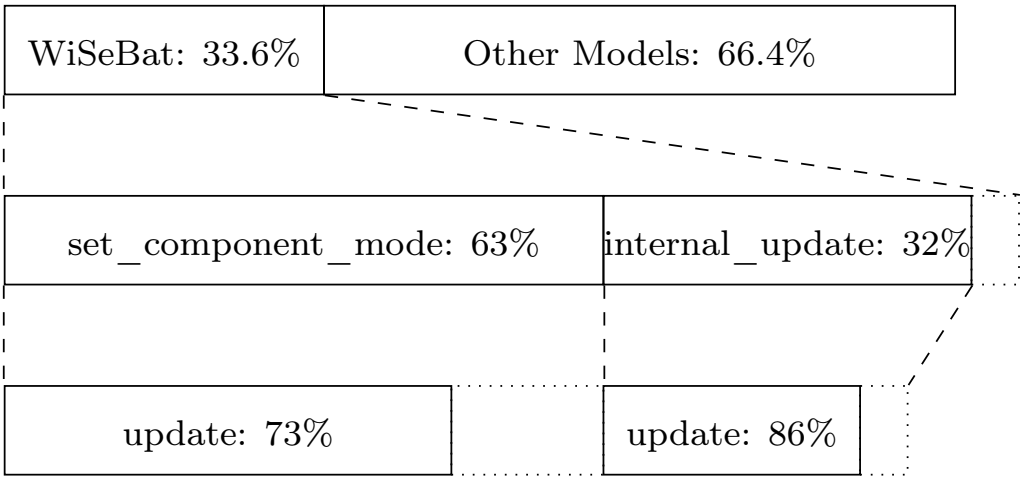



Figure 8.6 – Percentage of time spent in the WiSeBat model and on the most time consuming functions of WiSeBat, in a simulation using the ContikiMAC layer.

### 8.2.2 WiSeBat simulation overhead

The simulation time overhead of WiSeBat has been measured with the Instruments tool on Xcode. Of course, the percentage of simulation time spent on WiSeBat models depends on the overall configuration. Indeed with the heavyweight 802.15.4 MAC layer, WiSeBat-specific functions represent only 0.1% of the simulated time. In contrast, with the lightweight ContikiMAC layer, the percentage of simulation time spent in WiSeBat models is around 34% (see figure 8.6). Inside the WiSeBat models, the two functions set_component_mode and internal_update use 95% of the time. Both functions call some internal functions, in particular the main update function which corresponds to 73% of the time spent by WiSeBat and 24.5% of the whole WSNet simulation (assuming the ContikiMAC scenario).

Those results demonstrate that the second main goal of WiSeBat is attained: despite being accurate (with respect to real sensors), the computation time required to obtain accurate results remains reasonable (typically $\frac{1}{3}$ overhead, sometimes less). We expect WiSeBat to prove useful in a number of simulation scenarios, including those involving a respectable number of sensors.

## 8.3 Simulations

In this section we use WiSeBat to compare lifetime of nodes using different protocol stacks in simple and complex scenarios. We use the same application as in the previous section (see Figure 8.3). We first focus on the MAC and routing layers. We compare the X-MAC, ContikiMAC, and 802.15.4 MAC layers in a single node scenario (Subsection 8.3.1), then we examine ContikiMAC and 802.15.4 with a static or a RPL routing layer in a two-node scenario (Subsection 8.3.2).

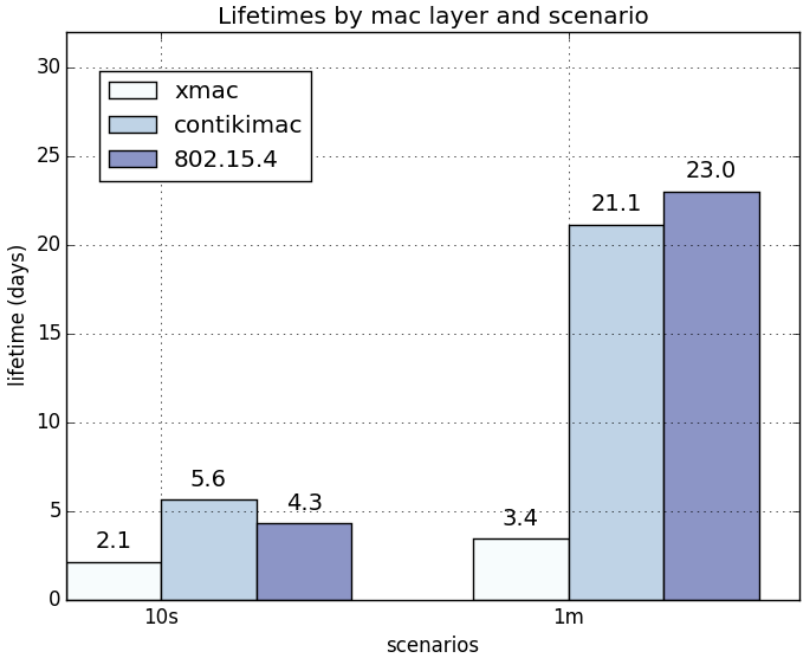



Figure 8.7 – Lifetime of a single leaf node with X-MAC, ContikiMAC and 802.15.4 MAC under two different duty-cycled scenarios. 802.15.4 mac performs better when the duty-cycle is longer.

### 8.3.1 Single Node Scenario

In this scenario, we consider a sensor node that repeatedly sends its data to a coordinator with varying time intervals (every ten seconds and every minute). The results are given in Figure 8.7. Here, there is no multi-hop communication, *i.e.*, sensor nodes do not need to forward, allowing them to return to sleep mode between two transmissions. The 802.15.4 standard allows for arbitrary long duty-cycles (for reduced functionality leaf nodes), which explains why 802.15.4 MAC layer performs better when the duty-cycle is longer. However, when nodes need to transmit every second, the near absence of control messages in ContikiMAC permits a longer lifetime than the 802.15.4 MAC layer. In both scenarios, X-MAC is outperformed by the two other mac layers.

### 8.3.2 Two-Node Scenario

In this scenario, two sensor nodes (one router node and one leaf node) and a coordinator are located in a line. The leaf node needs to send its data every minute to the intermediate router that forwards it to the coordinator. We compared the lifetime of the router and of the leaf node, with ContikiMAC or 802.15.4 mac layer with static or RPL routing layer. Figure 8.8 presents the results of this simulation. We first observe that using the RPL routing layer does not impact a lot the lifetime

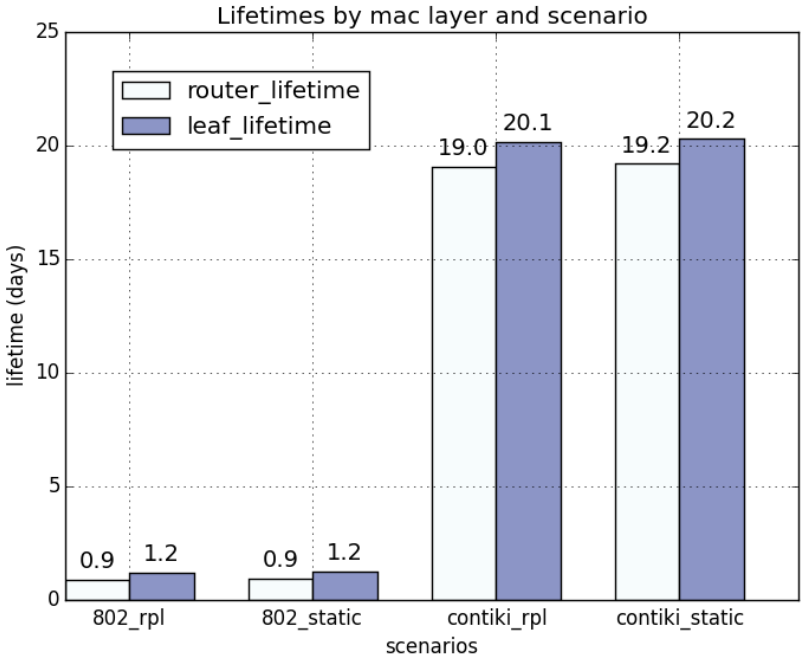



Figure 8.8 – Lifetime of a router node and a leaf node with ContikiMAC or 802.15.4 mac and RPL or static routing.

of the devices, compared to static routing. Also, we observe that ContikiMAC performs better than 802.15.4 mac layer. Indeed, with 802.15.4 mac, the fact that the router has to send beacons prevents it to last more than a day. Moreover, after the router shutdown, the leaf node uses all its remaining energy with control messages, trying to recover from the loss of its neighbor. With ContikiMAC, both router and leaf lifetimes ($19 - 20$ days) almost equal the lifetime in the single node scenario (21 days), which means that a node under ContikiMAC that forwards a message does not consume a lot more energy than a node that only sends a message.

## 8.4 Conclusion

We presented the WiSeBat model for accurate energy benchmarking of wireless sensor networks, and a reference implementation of our model in the WSNet simulator. WiSeBat includes an energy consumption model that takes all the components of the sensor node into account and a battery model with non linear effect. The battery model takes into account the rate capacity effect, the internal resistance and and voltage variations of the battery, that depends on the residual capacity of the battery. Those characteristics are computed from the data-sheet of the battery cell only.

We demonstrated that the lifetime estimation by WiSeBat is a huge improvement from the default models of WSNet as it provides $85 - 95\%$ accuracy in examined

scenarios, with a contained computational overhead (typically, $0.1 - 34\%$ of the simulation time). Our simulation results advocate that energy should be a first grade metric when evaluating efficiency of wireless sensor network protocols.

Chapter 9

# Energy Benchmarking of Broadcast Protocols

**Contents**



In this chapter we benchmark six existing energy-centric broadcast protocols that use varying transmission range: FLOOD, BIP, LBIP, DLBIP, RBOB and LBOP (see Sub-section 7.2.2 for details). These protocols were all evaluated using a specific simplified energy consumption model. Their model defines a function that maps a communication range to the amount of energy needed to transmit a message within this range. Given this energy consumption model, they were evaluated using simulations to compare their relative efficiency. Those simulations all use an ideal MAC layer, where no collision ever occurs. This leaves the open question of how those algorithms perform in a more realistic setting.

First we present the simulation setup. Then, in Section 9.2 we show and discuss the results of our simulation campaign.

## 9.1 Simulation Setup

We use WSNet simulator [FCF07] to perform our simulation campaign. We deploy 50 sensor nodes that are located uniformly at random in a square-shaped area. The size of the area varies from $300 \times 300m$ to $800 \times 800m$ to create various network densities. The protocol stack consists of a broadcast application, a broadcast protocol (whose performance is to be evaluated), a MAC layer (we consider both the 2400MHz OQPSK 802.15.4 CSMA/CA unslotted [802a] and ContikiMAC [Dun11]), a radio transceiver, and an omnidirectional antenna. For the environment, we use OQPSK modulation and the log-distance pathloss propagation model. The log-distance pathloss propagation model is more realistic than the range propagation and is simple enough to be easily predictable. This allows the nodes to choose the transmission power according to the desired range it wishes to attain.

A simulation setting consists in selecting a broadcasting protocol, a MAC layer, and the size of the area. For each setting, we run 50 simulations with various topologies. All measurements (remaining energy, number of receiving nodes per broadcast, and delay) are averaged over those simulations. Each topology is obtained by randomly deploying the nodes in the square area, and is used for all the simulation settings, so that different protocols are evaluated on the same topology. Table 9.1 summarizes the properties of the topologies we use (averaged over the 50 topologies we constructed for each size).

| Size | Density | Diameter | Vertex-Connectivity |
|---|---|---|---|
| 300 | 0.60 | 2.9 | 13.3 |
| 400 | 0.40 | 3.8 | 6.8 |
| 500 | 0.27 | 4.7 | 3.9 |
| 600 | 0.20 | 5.8 | 2.2 |
| 700 | 0.15 | 7.7 | 1.4 |
| 800 | 0.12 | 9.5 | 1.1 |

Table 9.1 – Average density, diameter, and connectivity of the topologies, depending on the area size.

For the energy model we used the WiseBat [BDBF+15] module with a real TMote Sky configuration (see Table 9.2). The current drawn by the CPU under a voltage $VCC$ is given by the formula $I[VCC] = I[3V] + 210(VCC - 3)$. We chose to power the device with a rechargeable Lithium Ion battery. Here, we used the data-sheet of the GP Battery 1015L08 model, that is designed for small devices.

Two execution scenarios are considered. In the first scenario (single source broadcast), Node 0 broadcasts a message to the other nodes every ten seconds, until the voltage is not sufficient for the node to work correctly (its cut-off voltage is 2.7V, see Table 9.2). In the second scenario (multiple source broadcast), a randomly selected node tries to broadcast its message, until there is no node working correctly and no node can initiate a broadcast.

## 9.2 Simulation Results and Discussion

We first present simulation results related to single source broadcast in Section 9.2.1, then multiple source broadcast in Section A.4.2.2. Our findings are further discussed in Section A.4.2.2

### 9.2.1 Single Source Broadcast

When the source of the broadcast does not change, it becomes the first node that stops working. This observation holds for every broadcasting protocol and every MAC layer. The reason is that the CPU of the source consumes some energy to initiate the broadcast, and our simulations show that no broadcasting protocol

| Radio Chipcon CC2420 | | CPU Texas Instruments MSP430 F1611 | |
|---|---|---|---|
| Tx 0dB | 17.4 mA | | |
| Tx -1dB | 16.5 mA | Run 8MHz 3V | |
| Tx -3dB | 15.2 mA | (with flash read) | 4mA |
| Tx -5dB | 13.9 mA | Sleep | 2.6$\mu$A |
| Tx -7dB | 12.5 mA | voltate cut-off | 2.7V |
| Tx -10dB | 11.2 mA | | |
| Tx -15dB | 9.9 mA | | |
| Tx -25dB | 8.5 mA | | |
| Rx | 19.7mA | | |
| Idle | 365$\mu$A | | |
| Sleep | 1$\mu$A | | |
| Volt. Regulator | 20$\mu$A | | |

Table 9.2 – Voltage specification of the TMote Sky Hardware

takes this fact into account when implementing their strategy. However, broadcast protocols exhibit various differences depending on the considered performance metric.

**Number of Broadcasts** The overall number of broadcasts, which directly depends on the lifetime of the source node, varies depending on the network protocol stack used. Figure 9.1 presents for each MAC layer the number of broadcasts that could be achieved depending on the size of the area, for each considered broadcasting protocol.

With 802.15.4 MAC, the number of broadcasts is roughly equivalent for all the protocols, and on average, the number of achieved broadcasts lies between 6300 and 6600, and does not depend on the size of the area. BIP has a small advantage, then FLOOD lasts a little longer than the other distributed broadcasting protocols.

With ContikiMAC, the number of broadcasts varies significantly with the considered broadcasting protocol and, to a lesser extent, with the size of the area. For BIP and LBIP protocols, the number of broadcasts is around 110,000 for dense networks. This number decreases to around 100,000 for sparse networks. In contrary, for FLOOD and DLBIP protocols, the number of broadcasts increases with the size of the area from around 80,000 to 95,000. RBOP and LBOP protocols have lower performance with less than 70,000 broadcasts.

We see that LBIP appears to be the best distributed protocol. Surprisingly (considering its energy unawareness) FLOOD exhibits very good performance, similar to DLBIP, and outperforms both RBOP and LBOP.

**Number of Receiving Nodes** Contrarily to what we expected, all the nodes do not necessarily receive every message. For some broadcasting protocols, the number of receiving nodes can vary a lot. This is due to the fact that when broadcasting

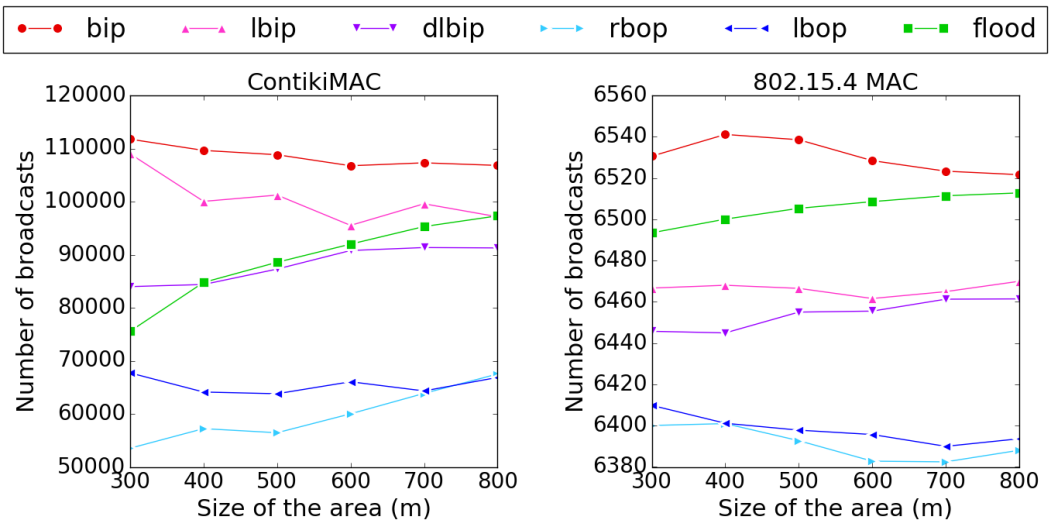



Figure 9.1 – Average number of broadcasts, depending on the size of the area, for each broadcasting protocol.

a packet, the MAC layer does not request an acknowledgment, so the packet (due to interference) may not be received by its intended destination, and this loss is never notified to the broadcasting protocol. Again, the MAC layer impacts significantly the results (see Figure 9.2). In the sequel, the *reachability* metric denotes the percentage of nodes that receive the message.

With ContikiMAC, BIP, DLBIP, and FLOOD protocols offer very good performance regardless of the size of the area. LBIP and RBOP are below, but LBIP performs better as the density of the graph decreases.

With 802.15.4, FLOOD, DLBIP exhibit results that are similar to the previous case. However, BIP has one of the worst performances, with RBOP and LBOP. Their performance increases with the size of the area but the performance of BIP with 802.15.4 is far below its performance with ContikiMAC. Again, LBIP reachability increases until 80% as the density of the networks decreases.

In both cases, the improvement observed when density decreases can be explained by the fewer number of message collisions that go unnoticed.

**Amount of Remaining Energy** In previous work, the amount of energy in the network upon simulation termination was analyzed. For our purpose, the energy that remains in the rest of the network after that the battery of the source node is depleted is not as relevant as the other metrics we considered. Indeed, for ContikiMAC, the amount of energy remaining is correlated with the two other metrics. In more details, the greater number of broadcasts and the greater number of receiving nodes, the fewer the amount of energy remaining will be. So, less remaining energy actually implies better performance of the protocol.

With 802.15.4, the amount of remaining energy is similar for all broadcasting protocols, and cannot be used to differentiate their performance.

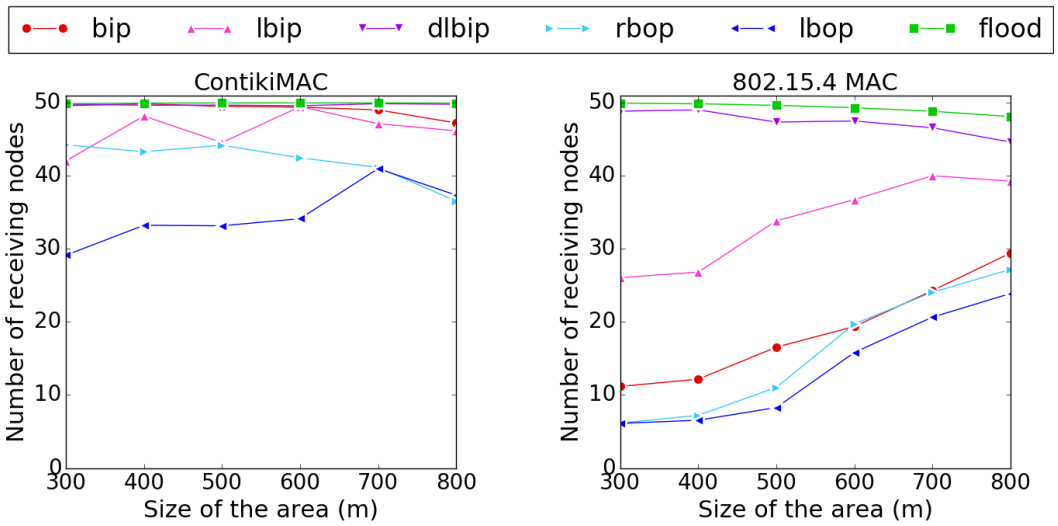



Figure 9.2 – Average number of receiving nodes, depending on the size of the area, for each broadcasting protocol.

### 9.2.2 Multiple Source Broadcast (Gossip)

In this scenario, each broadcast is initiated by a randomly chosen source. Each simulation uses the same random order to make sure the differences between two simulations do not depend on this order. The simulation terminates when no nodes are alive (hence, we are not interested either in the amount of energy remaining in the network at the end of the simulation). In this setting, it is interesting to investigate the number of receiving nodes, and the delay over time.

**Number of Receiving Nodes** Figure 9.3 (respectively, Figure 9.4) shows the number of nodes that receive the message using ContikiMAC as a MAC layer (respectively, using 802.15.4 MAC), for each considered broadcasting protocol and for various sizes. The $x$-axis represents the number of broadcasts, and it is proportional to the time (because one broadcast occurs every 10 seconds). A point of the graph with $x$-coordinate $i$ is the average number of receiving node for the $i$-th broadcast to the $(i + 100)$-th broadcast, for 50 simulations considering different topologies. Due to the sliding window used to compute the average, the graph is smoother than if we just take the average of the $i$-th broadcast over all the simulations. We observe that the number of receiving nodes at the beginning is consistent with the case of a unique source (See Section 9.2.1). Also, the number of broadcasts until a decrease starts is slightly more than in the case of a unique source, as the load is more evenly shared among sources.

With ContikiMAC, BIP is the protocol that keeps 100% reachability for the longest period of time. Then FLOOD and DLBIP are close runner-up. In dense networks, DLBIP is better because the decrease in the number of receiving nodes is slower. However, in sparse networks, FLOOD keeps 100% reachability for around

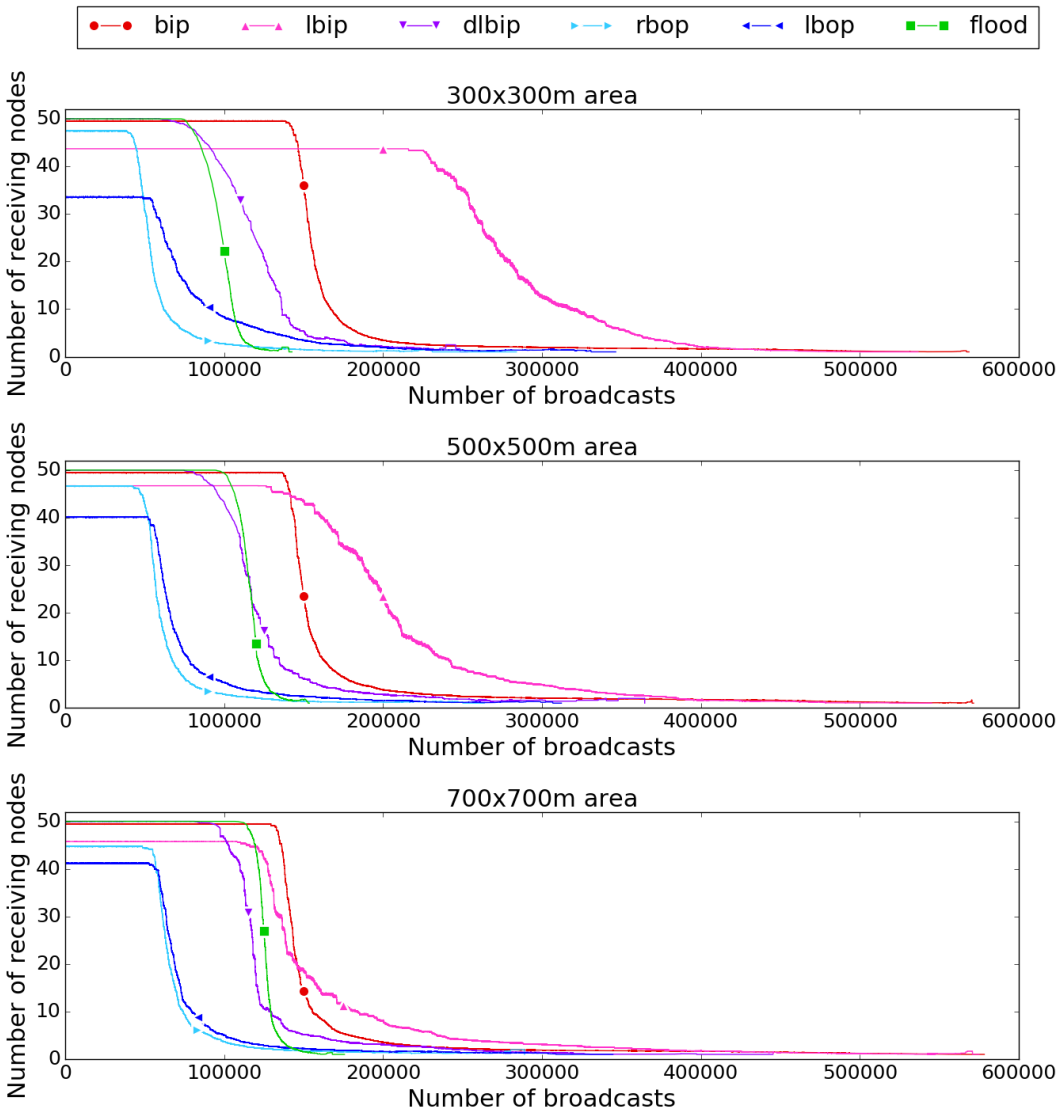



Figure 9.3 – Number of receiving nodes over time with ContikiMAC.

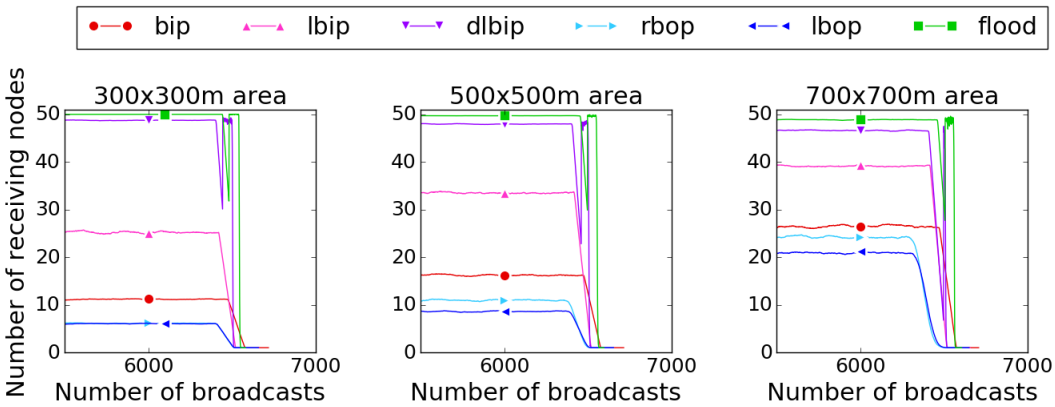



Figure 9.4 – Number of receiving nodes over time with 802.15.4 MAC.

10% more broadcasts. It is interesting to see that LBIP has around 90% reachability, but performs 50% more broadcasts compared to BIP and has a really slow decrease. Finally, RBOP and LBOP protocols are outperformed by the other protocols. We can notice that for sparse networks, the performance of BIP, LBIP, DLBIP, and FLOOD appear to converge to about 130,000 broadcasts.

With 802.15.4 MAC we observe that, for every protocol, the decrease of the number of receiving nodes is faster than with ContikiMAC. Also, all broadcasts are almost equivalent in the number of broadcasts performed. This is mainly because 802.15.4 MAC consumes the majority of the available energy, so that the other source of consumption become less significant. The FLOOD protocol is the only protocol with almost 100% reachability until the end. DLBIP performs really well with more than 90% reachability. The other protocols have bad performance. In particular, BIP is below LBIP and DLBIP in terms of number of receiving nodes, which was already observed in the case of a single source.

**End-to-End Delay** The end-to-end delay we consider is the duration between the start of the broadcast and the time of the reception of the last message for this broadcast (if not all nodes receive the broadcast message, the time of reception of the last node that receives the message is used).

With ContikiMAC (see Figure 9.5), we note that FLOOD, LBIP, and LBIP have really good performance, with a delay from 500ms for dense networks to 1s for sparse networks. Also, even if BIP has good performance in terms of number of broadcasts and reachability, it has a high delay of 2.5s, regardless of the density of the network. LBOP and RBOP have even greater delay.

With 802.15.4 (see Figure 9.6), the overall delay is better than with ContikiMAC by two orders of magnitude. BIP, LBIP, DLBIP, and FLOOD have similar performance with a delay around 10ms. RBOP and LBOP exhibit a delay that is five times greater. The results for the other size of area are not presented because they are equivalent.

In both cases, we can see that the delay decreases when the number of receiving

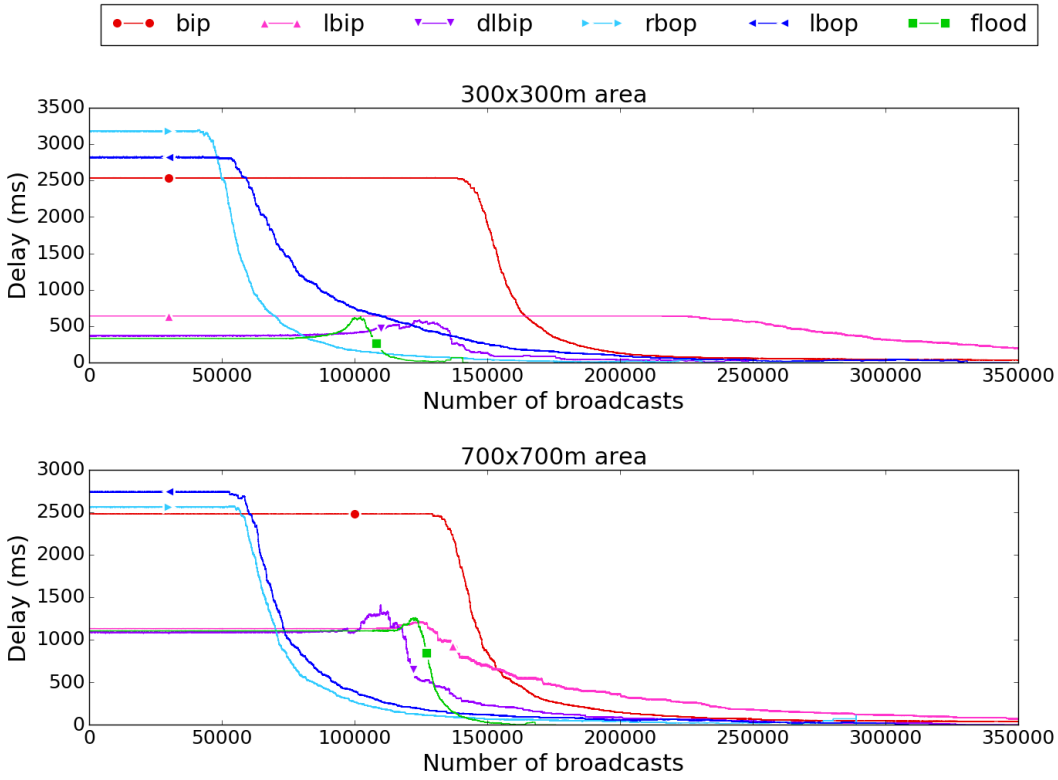



Figure 9.5 – End-to-end delay (in ms) over time with ContikiMAC.

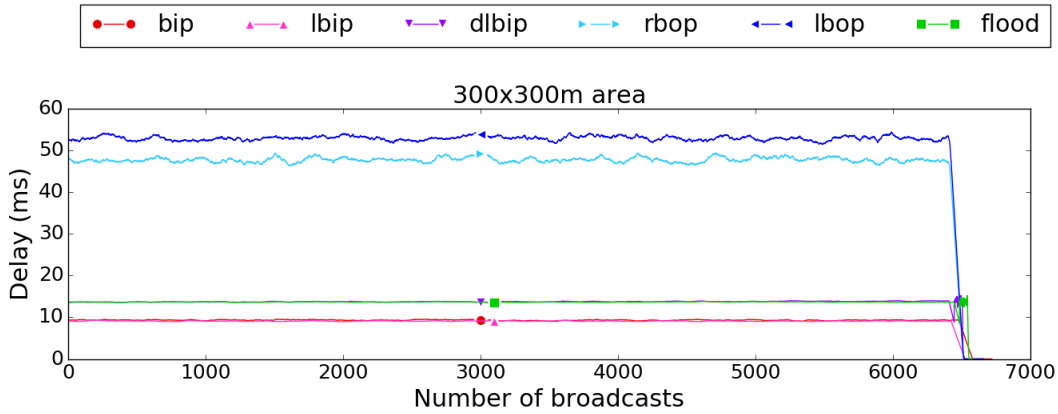



Figure 9.6 – End-to-end delay (in ms) over time with 802.15.4 MAC.

nodes decreases. However, we observe that for FLOOD and DLBIP protocols, the beginning of the decrease is preceded by a small peak, probably because after some nodes have stopped working, the network has a lower density, resulting in a greater delay.

### 9.2.3 Discussion

Our results show that the impact of the transmission collisions due to wireless interference is not uniform for each broadcasting protocol. For instance, we saw that the number of receiving nodes with RBOP is low for dense networks and increases as the density of the network decreases. This implies that interference has a huge impact on RBOP. This impact could be: *(i)* direct *i.e.*, the protocols send only few messages, so that each node often receives the message from only one neighbor, and if this message is lost, a subset of the network does not receive the message, or *(ii)* indirect *i.e.*, the nodes selected for broadcasting the message are chosen in such a way that their transmissions always collide at the receivers, particularly due to the hidden terminal problem (two nodes that are out of range from each other but that have a common destination neighbor). The first point can explain why the FLOOD protocol exhibits such good performance. Indeed, each node transmits the message, which causes more interference, but this also increases the probability that a node receives the message and is later able to retransmit it to the rest of the network.

In more details, there is a limit in the amount of interference in the network because if the MAC layer detects that another node is transmitting, it waits until the channel is clear. At some point, increasing the number of nodes that wants to transmit does not increase the number of lost packets. So that the probability that at least one node receives the message for the first time increases.

The good overall performance of FLOOD confirms the practical relevance of having redundancy when broadcasting a message, and that this redundancy *does not necessarily imply a higher overall energy cost.* This is even more stringent when the source of the broadcast remains the same. However, when the source of the broadcast is randomly chosen, there are some cases when LBIP or DLBIP may be more appropriate. For instance, with ContikiMAC, we see in Figure 9.3 that LBIP performs between two and three times more broadcasts in dense networks compared to FLOOD, albeit with less reachability. Also, DLBIP performs better than FLOOD in dense networks. So, in general, the overall best candidate is FLOOD, but some specific settings command the use of LBIP or DLBIP.

## 9.3 Conclusion

We used our model WiSeBat presented in the previous chapter to evaluate several energy-central protocols. We focused on the problem of broadcasting a message in an energy-efficient manner, in a wireless sensor network where nodes are able to change their transmission power. We studied six broadcasting protocols that are representative of the current state of the art. We answered the question left open by

the previous work: how broadcasting protocols perform with realistic (simulated) devices in a realistic (simulated) environment? We found that the energy consumption does not depend on the protocols as one could expect from the previous studies. Indeed, it is not realistic to consider only the energy consumed by the radio during the transmission. Especially, we show that the hierarchy of the broadcasting protocols (based on their performance in an ideal setting) is *not* preserved in the more realistic setting. Also, we show that the MAC layer, also does not impact all broadcast protocols in the same way (some protocols perform better with ContikiMAC than with 802.15.4 MAC, while other do not). Quite surprisingly, it turns out that the very simple *flooding* protocol (used as a theoretical lower bound in previous work), is actually one the best distributed broadcasting protocols in our realistic environment.

Our conclusion is that focusing on power transmission to improve energy-efficiency of broadcast protocols for sensor networks is not the right choice. Using ideal MAC layer is not well-suited to evaluate the efficiency of a protocol, as collisions prevent many protocols to achieve acceptable coverage, especially when the density of the network is high.

Chapter 10

# Conclusion

**Contents**



This concluding chapter surveys thesis contributions (see Section 10.1) and discusses questions raised by our work (see Section 10.2).

## 10.1 Overview of Thesis Contributions

### 10.1.1 Data Aggregation

The second part of the thesis focuses on the problem of aggregating data from all the nodes in the network to a sink node in minimum time. We consider two variants of the problem.

First, in Chapter 5, we suppose that the network is a WSN (possibly dynamic) modeled by a UDG (or a sequence of UDGs). In this case, interference occurs at a node when two neighbors transmit a message simultaneously. We investigated the complexity of finding a solution to the minimum data aggregation time problem by a centralized algorithm. We characterized the NP-completeness depending on the maximum node degree $\Delta$ of the graph. In a static WSN, the problem is trivial if $\Delta$ is 2 and is NP-complete otherwise. In a dynamic WSN, the problem is trivial if $\Delta$ is 1 and NP-complete otherwise. We presented a centralized approximate algorithm that constructs a data aggregation schedule. We showed that the duration of the schedule is smaller than the duration of $n-1$ independent foremost convergecast trees. In the class of $T$-bounded recurrent connected graphs, this implies a duration smaller than $T(n-1)$ time steps. We also conjecture that in the worst case, the data aggregation has a duration of $(\Delta-1)\log_{\Delta}(n(\Delta-1)+1)-\Delta+2$ independent foremost convergecast trees.

Second, in Chapter 6, we suppose that the network is dynamic, and nodes interact in a pairwise manner. The goal is to find an algorithm that receives the interactions one by one and has to decide whether a node transmits its data to the other or not. We study this problem depending on how the interactions occur,

and on the initial information given to the nodes. If an adversary chooses the interactions depending on the previous choices of the algorithm (an online adaptive adversary), the problem is not solvable without giving additional knowledge to the nodes. With some information the problem may be solvable. For instance, if nodes know the eventual underlying graph, there is an algorithm that terminates.

When the interactions are chosen randomly among all possible interactions (each one with probability $\frac{2}{n(n-1)}$), then we give two optimal algorithms: (a) when nodes have no knowledge, our algorithm Gathering terminates in $O(n^2)$ interactions in expectation and (b) when each node knows its next interaction with the sink, our algorithm Waiting Greedy with parameter $\tau = \Theta(n\sqrt{n\log(n)})$ terminates in $\tau$ interactions w.h.p. Our two algorithms are oblivious and optimal in the class of non-oblivious algorithms (that use the same initial knowledge).

These results were published in international conferences SSS 2015 [BT15] and IEEE ICDCS 2016 [BMT16], and the Information and Computation international journal (to appear).

### 10.1.2 Lifetime Estimation of a Wireless Sensor Network

The third part of the thesis focuses on the lifetime evaluation of WSNs. To do so, we reviewed the existing models for the energy consumption and the battery. We proposed a new model called WiSeBat (Chapter 8) that aims to be accurate while remaining simple. We evaluated its implementation in WSNet simulator against a real sensor device. The results showed that WiSeBat outperformed the default battery model of WSNet, and it provides 85 - 95% accuracy in examined scenarios, with a contained computational overhead (typically, 0.1 - 34% of the simulation time).

A lot of previous work use over-simplified energy model to benchmark energy-centric protocols. A typical example is the series of work on energy-efficient broadcast protocols in WSNs. To know if these results can be considered reliable for real applications, we used WiSeBat to evaluate six of such protocols (see Chapter 9). We showed that the hierarchy (based on their performance in an ideal setting) was not preserved in the more realistic setting. Quite surprisingly, it turned out that the very simple flooding protocol (used as a theoretical lower bound in previous work), was one the best distributed broadcasting protocols in our realistic environment. Our conclusion is that using a realistic interference model is critical to evaluate the performance of algorithms in WSNs.

These results were published in international conferences IEEE FDL 2015 [BDBF+15] and NETYS 2016 [BT16].

## 10.2 Perspectives

The contributions of this thesis concern two significant problems faced in WSNs: aggregating data from every node to the sink in an efficient way, and evaluating the lifetime of the network. We presented several models that fit each problem, and

theoretical analysis, simulations, and measurements, which lead to optimal results, impossibility results, open source tools, and engaging open questions.

**On the Practical Side** Our energy model WiSeBat has shown good performance on a specific scenario and our short term objective is to perform large scale scenarios evaluations, using different platforms and different standards (*e.g.* the 802.15.6 standard). A robust evaluation will motivate its systematic use for energy benchmarking of new protocol proposals for wireless sensor networks and body area networks. There is a lot of work proposing new protocols or comparing existing protocols that would benefit to have an accurate energy model. For instance, broadcast protocols for body area networks [BCPPB15] and network coding [RG13] currently use simple linear battery models. In a farther future, including recent advances related to energy harvesting in our methodology is a challenging open question.

After using WiSeBat to benchmark broadcasting protocols, we concluded that future protocols are bound to integrate more realistic interference and energy consumption models to be relevant in practice. A cross-layer approach with the MAC layer and the broadcast layer helping each other is a possible path for long-term research.

Another interesting open question is the impact of mobility of sensor nodes on the energy efficiency of broadcast protocols. The six broadcast protocols we considered assume a static topology. Several evaluations of broadcast protocols in WSNs have been done when the nodes are mobile [WC02, MBGW13]. However, protocol schemes and their evaluation are entirely different. In the work of S.Medetov et al. [MBGW13], the remaining energy is also considered, but with an ideal battery model and full-sized computer network stack (an 802.11 MAC layer is assumed). Energy benchmarking those mobility-aware broadcast protocols in a realistic setting such as that of this paper is a short-term research objective.

Finally, as an immediate continuation of our work on data aggregation, we plan to investigate how our distributed online data aggregation algorithms perform in real interaction graphs, obtained from human interactions or interactions between nodes on a moving person that changes postures.

**On the Theoretical Side** In the minimum data aggregation time problem with a global point of view, we can consider a similar problem by searching a data aggregation schedule that ends at given time and starts as late as possible (instead of one that start a 0). One can show that this problem is equivalent to the one we considered. In this case, one may use latest convergecast tree. Contrarily to foremost convergecast tree, a latest convergecast tree can be defined such that every journey from a leaf to the sink is a latest journey i.e., a journey that ends at a given time and starts as late as possible. Also, this problem must assume that the time domain is infinite in the negative direction (the time instants live in Z instead of N). For long-term research, we plan to focus on this problem, that leads to interesting new definitions.

For the distributed online data aggregation problem, our analysis opens several scientific challenges *(i)* in the short run:

1. Does allowing nodes to send their data a constant number of times instead of once impact our results?
2. Does there exist a randomized algorithm that terminates against any oblivious adversary?

and *(ii)* in the long run:

1. What knowledge has a real impact on the lower bounds or algorithm efficiency?
2. Can similar optimal algorithms be obtained with fixed memory or limited computational power?
3. Can randomized adversaries that use a non-uniform probabilistic distribution alter significantly the bounds presented here in the same way as in the work by Yamauchi *et al.* [YTKY12] ?

Annexe A

# Version française

## Sommaire



Cette annexe constitue un résumé des principales contributions de cette thèse. Ce résumé reste volontairement à un niveau purement intuitif car le lecteur intéressé par la présentation complète et rigoureuse d'une contribution sera renvoyé au chapitre correspondant de la thèse.

Le nombre d'appareils connectés à Internet ne cesse d'augmenter. Cette augmentation est permise par la réduction des coûts de construction et accentuée par le développement de l'*Internet des objets*, qui consiste à connecter des appareils comme des réfrigérateurs, des imprimantes, ou des lampes, directement à Internet. Leur connectivité leur permet d'être contrôlés à distance en temps réel.

Par ailleurs, l'augmentation du nombre d'appareils communicants permet aussi la création de réseaux autonomes contenant plusieurs appareils, dont un seul, appelé coordinateur, est connecté à un réseau extérieur comme Internet. Les nœuds du réseau sont autonomes et s'organisent seuls, sans l'aide d'une infrastructure centralisée. Cette architecture permet d'utiliser des nœuds possédant des capacités plus faibles, consommant moins d'énergie et utilisant des protocoles plus légers que le protocole Internet. Ces caractéristiques permettent de déployer un grand nombre d'appareils munis de capteurs (tels qu'un thermomètre, un accéléromètre ou gyroscope) afin de surveiller des environnements inaccessibles aux humains ou ne possédant pas d'infrastructures existantes. De tels réseaux sont appelés des *Réseaux de Capteur Sans Fil*.

La surveillance des signes vitaux d'un patient est une application particulièrement intéressante des réseaux de capteurs sans fil. De nombreuses maladies peuvent être détectées ou soignées avec une surveillance étroite du patient, comme les maladies cardiovasculaires, le diabète, l'hypertension, ou l'asthme. La surveillance peut être effectuée après une opération importante, de manière systématique (*e.g.* , pour prévenir la mort subite des nouveaux nés), ou pour améliorer la qualité de vie des malades (*e.g.* , une pompe administrant une dose adaptée d'insuline en fonction du taux de glucose mesuré par un capteur).

Pour contrôler l'état du patient, des capteurs physiologiques peuvent être placés sur ou dans le corps humain, formant un *Body Area Network*. Les données des capteurs sont agrégées périodiquement jusqu'au coordinateur, qui les analyse et peut envoyer une alerte à l'hôpital en cas de risque. Dans ce contexte, le réseaux se doit d'être fiable, durable et prévisible. Dans le même temps, les capteurs qui le constituent doivent être petits, utiliser des batteries, et avoir une longue durée de vie.

Les communications dans les réseaux de capteurs sans fil s'effectuent de manière multi-sauts. Les capteurs les plus éloignés du coordinateur doivent envoyer leur données de mesure à des nœuds intermédiaires qui doivent relayer les messages jusqu'au coordinateur. La stratégie utilisée pour récolter les données jusqu'au coordinateur a un impact important sur la consommation énergétique des capteurs.

## A.1 Contexte de la thèse

Les réseaux de capteurs sans fil ou *Wireless Sensor Netowks* (WSN) sont constitués de nœuds capteurs, qui communiquent à l'aide d'ondes radio et capables de récolter des données, de les analyser et de les transmettre. Ces réseaux ont des applications variées, en fonction de la zone où ils sont déployés : militaire ou de sauvetage dans des zones pouvant être inaccessibles aux humains ; sanitaire avec des capteurs déployés sur et dans le corps humain ; surveillance avec des capteurs sur les voitures d'une ville ou les arbres d'une forêt, etc. Les nœuds sont autonomes en énergie et il est primordial d'assurer leur longévité sans retarder la récolte des données.

**Étude de la Performance** La caractéristique principale des réseaux de capteurs sans fil est que le médium propageant les signaux de transmission est partagé. Les modèles les plus simples permettent de représenter de tels réseaux par un graphe de communication. Le changement de topologie du réseau (dû au mouvement des capteurs par exemple) est alors capturé par un graphe évolutif, c'est-à-dire une suite de graphes statiques représentant l'état du réseau à chaque instant. Cependant, pour étudier de manière plus précise les performances d'un protocole, il est important d'utiliser des modèles de propagation du signal qui prennent en compte les interférences.

**Consommation Énergétique** Réduire la consommation énergétique des protocoles utilisés dans les réseaux de capteurs sans fil constitue un défi majeur. Un capteur consomme de l'énergie lorsqu'il effectue des actions (a) dont il est à l'origine, telles que la mesure de l'environnement ou l'envoi d'un message, (b) venant d'autres capteurs, comme faire suivre un message au nœud suivant, ou (c) dues à un facteur externe, comme une retransmission après une transmission qui a échoué. Chaque niveau dans la pile de communication des capteurs peut avoir une influence importante sur chacun de ces aspects et donc sur la consommation du capteur. Le protocole permettant l'agrégation des données du réseau jusqu'au coordinateur étant celui déclenché le plus souvent, il est primordial de minimiser son impact énergétique.

## A.2 Modèle

### A.2.1 Modélisation de la Couche Physique

**Modélisation du Lien Radio** En l'absence d'interférence, la porté radio peut être définie simplement à l'aide d'un seuil. Lorsqu'un nœud $i$ transmet un message, un autre nœud $j$ le reçoit si le rapport entre la puissance du signal reçu (le produit de la puissance de transmission par l'atténuation) et le bruit de mesure est supérieur à un seuil donné.

Pour améliorer la précision du modèle, le seuil peut être remplacé par une probabilité d'erreur du paquet. Un paquet sera correctement reçu par le récepteur de manière probabiliste, en fonction du rapport signal à bruit, et de la modulation utilisée.

**Modélisation des Interférences** Lorsque plusieurs signaux issus de transmissions concurrentes se superposent, une interférence se produit, pouvant perturber la bonne réception des messages. Afin de prendre en compte ces interférences, le rapport signal à bruit peut être remplacé par le rapport signal à interférences plus bruit (SINR). Ainsi, lors de la réception d'un signal, la somme de tous les autres signaux reçus est considérée comme du bruit lors de la comparaison au seuil de réception.

Les travaux de P. Cardieri [Car10] et de A. Iyer *et al.* [IRK09] montrent l'impact du modèle d'interférence sur la précision des simulations. Ils concluent que l'utilisation du modèle SINR (ou plus précis) est essentiel pour une évaluation réaliste des protocoles dans les réseaux de capteurs sans fil.

### A.2.2 Le Réseau Comme un Graphe de Disque-Unité

Lorsque l'on suppose que tous les capteurs sont identiques, on peut normaliser la portée radio à 1 et représenter le réseau avec un graphe de disque-unité. Chaque capteur est un disque de diamètre 1 et deux capteurs peuvent communiquer si leur disque s'intersecte (voir figure A.1).

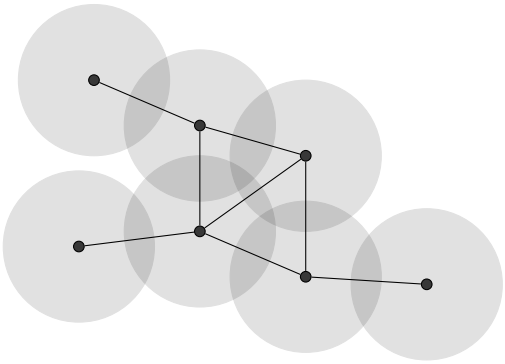


FIGURE A.1 – Un exemple de graphe de disque-unité

Cette représentation permet d'utiliser le langage de la théorie des graphes dans les réseaux de capteurs sans fil. Par exemple, la transmission d'un message entre deux capteurs se fait selon un chemin dans le graphe de communication. Une route optimale en nombre de sauts correspond alors à un plus court chemin dans le graphe.

### A.2.3 Modèle pour les Réseaux Dynamiques

Les liens de communication d'un réseau dynamique peuvent disparaitre ou apparaitre au cours du temps, sans que cela soit considéré comme une erreur. Nous choisissons de modéliser un réseau de capteurs dynamique par un graphe *évolutif*, c'est-à-dire une suite de graphes statiques. Chaque graphe de la suite représente le graphe de communication à un instant donné. De manière formelle, un graphe évolutif $G$ est un couple $(V, \{E_i\}_{i\in\mathbb{N}})$ où $E_i$ est l'ensemble des liens de communication au temps $i$. Dans le cas d'un réseau de capteurs sans fil, chaque graphe $(V, E_i)$ est un graphe d'intersection de disques. Le degré maximal d'un graphe évolutif est le degré maximal parmi tous les graphe $(V, E_i)$. Dans un graphe évolutif, une arête est un couple $(a, t)$ où $t$ est un temps et $a$ une arête dans $E_t$. Ainsi, un chemin évoluant au cours du temps $\mathcal{J}$, appelé *trajet*, est une suite d'arêtes de la forme $((a_1, t_1), (a_2, t_2), \ldots, (a_l, t_l))$ qui commence au temps $depart(\mathcal{J}) = t_1$ et se termine au temps $arrivee(\mathcal{J}) = t_l + 1$.

## A.3 Le Problème de l'Agrégation des Données

Dans la deuxième partie de cette thèse, on aborde le problème de l'agrégation de données. Dans un réseau où chaque nœud contient initialement une donnée, l'agrégation de données consiste à regrouper toutes ces données jusqu'au nœud coordinateur. Pour économiser l'énergie, on suppose que les données peuvent être agrégées, c'est-à-dire qu'il existe une fonction prenant deux données en entrée et retournant une seule donnée sensée contenir l'information importante présente dans les données d'entrée (on pourra penser par exemple aux fonctions minimum ou maximum).

l'agrégation de données est ensuite considérée comme une simple donnée. Ainsi, un nœud pourra attendre de recevoir les données de plusieurs voisins avant de décider à son tour d'envoyer en une fois l'agrégation de toutes les données reçues en direction du coordinateur.

De cette manière, $n$ transmissions sont suffisantes pour récolter les données de $n$ capteurs vers un nœud coordinateur (contre $\Omega(n^2)$ sans agrégation, dans le pire des cas). Le problème de l'agrégation des données consiste à trouver un ordonnancement des communications ou chaque nœud transmet **une seule fois**, ce qui permet d'agréger les données de tous les capteurs du réseau jusqu'à un nœud coordinateur en un minimum de temps.

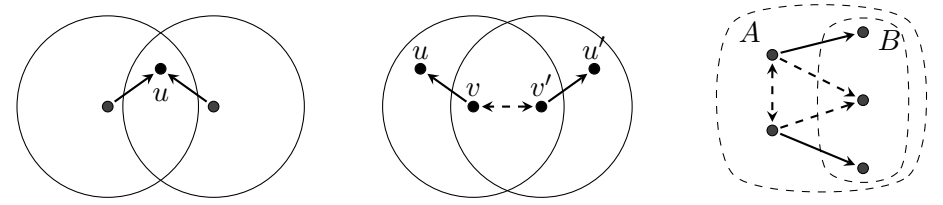
On étudie ce problème dans deux cas. Le premier est dans les réseaux de capteurs sans fil (statique ou dynamique) où les collisions dues aux interférences ne sont pas gérées par la couche MAC. Les résultats dans ce cas sont détaillés dans la section A.3.1. Le second cas s'intéresse aux solutions distribuées et en ligne du problème dans les réseaux dynamiques arbitraires. Les résultats dans ce cas sont détaillés dans la section A.3.2

**Dans les Réseaux de Capteurs Sans Fil** Afin d'économiser de l'énergie dans un réseau de capteurs sans fil, il est aussi important de prévenir les collisions due aux interférences. Pour cela on suppose que lorsque deux capteurs transmettent simultanément, les voisins communs de ces capteurs ne reçoivent aucune donnée, à cause des interférences. Dans ce contexte, l'ordonnancement qui permet l'agrégation des données doit s'assurer que les données arrivent bien jusqu'au coordinateur sans interférence.

**Agrégation en Ligne Dans un Réseau Dynamique Arbitraire** Dans le cas d'un réseau dynamique arbitraire on cherche à trouver une solution en ligne au problème de l'agrégation de données. C'est-à-dire qu'on cherche à construire un algorithme qui reçoit la séquence de graphes (qui représente l'évolution du réseau) comme un flux de données et doit prendre ces décisions au fur et à mesure. Pour simplifier on suppose que les interactions se produisent toujours entre deux nœuds. A chaque instant, lors d'une interaction entre deux nœuds, l'algorithme doit décider si un nœud transmet ou non sa donnée à l'autre. Le but étant toujours de minimiser le temps pour agréger toutes les données jusqu'à un nœud coordinateur. On remarque que dans ce modèle on suppose que les collisions sont gérées par la couche MAC.

### A.3.1 Dans les Réseaux de Capteurs Sans Fil

A chaque instant, un nœud peut transmettre ou recevoir mais ne peut pas faire les deux à la fois. Lorsque deux capteurs transmettent simultanément, leurs voisins communs ne reçoivent aucune donnée. On dit que les données sont agrégées au temps $t$ d'un ensemble de nœuds $A$ vers un ensemble de nœuds $B \subset A$ si les données des nœuds de $A \backslash B$, transmises simultanément à l'instant $t$, sont correctement reçues par des nœuds de $B$ (*i.e.*, sans interférence, voir figure A.2c). C'est-à-dire que chaque




(a) $u$ ne reçoit aucune donnée

(b) $u$ reçoit les données de $v$ et $u'$ les données de $v'$

(c) Les données sont aggrégées de $A$ vers $B$

FIGURE A.2 – Contraintes de Communication

nœud $u$ de $A \backslash B$ possède au moins un voisin $v$ dans $B$ qui reçoit ses données ($u$ est le seul voisin de $v$ qui transmet au temps $t$). Une agrégation de données de $V$ vers $s$ de durée $d$ et de départ $t_0$ est une suite décroissante d'ensembles $V = R_0 \supseteq R_1 \supseteq \ldots \supseteq R_d = \{s\}$ telle que les données sont agrégées au temps $t_0 + i$ de $R_i$ vers $R_{i+1}$, pour tout $i = 0, 1, \ldots, d-1$.

Une instance du problème de l'*Agrégation de Données en Temps Minimum* (ADTM) est un triplet $(G, s, t)$ où $G = (V, \{E_i\}_{i \in \mathbb{N}})$ est un graphe évolutif, $s \in V$ un nœud appelé *puits*, et $t$ un temps. Une solution de l'instance $ADTM(G, s, t)$ est une agrégation de données de $V$ vers $s$ de départ $t$ et de durée minimum. La durée minimum de l'agrégation est notée $ADTM_{Opt}(G, s, t)$. Dans le cas des graphes statiques, la donnée de $t$ peut être omise.

**Travaux connexes** Dans le cas statique, le problème de l'agrégation de données en temps minimum (ADTM) a été largement étudié. Une version similaire fut présentée par Anamalai *et al.* [AGS03] avec la possibilité pour les nœuds de choisir parmi un nombre fini de canaux différents pour éviter les collisions. Puis, Chen *et al.* [CHZ05] ont donné un modèle formel pour l'étude du problème ADTM, et ont montré que le problème est NP-Complet même si le graphe est de degré au plus quatre. Dans le même article, ils ont donné la borne inférieure et un majorant pour la durée minimum d'agrégation des données. Ils ont aussi proposé le premier algorithme d'approximation. Pour finir, de nombreux articles ont présenté des algorithmes d'approximations centralisés ou distribués de plus en plus performants, utilisant la nature géométrique des graphes d'intersections de disques, améliorant par la même occasion la majoration de la durée suffisante pour agréger les données du réseau. Cependant, aucune de ces études ne généralise son approche au cas des graphes évolutifs. À notre connaissance, aucune étude sur le problème de l'agrégation de données prenant en compte les collisions n'a été réalisée dans les graphes évolutifs.

**Résumé des Contributions** Notre première contribution est de caractériser exactement ce qui fait de l'agrégation de données un problème NP-Complet. Dans les graphes *statiques*, le problème est trivial pour les graphes de degré au plus deux,

et nous montrons que le problème reste NP-Complet dans les graphes de degré au plus trois. De manière similaire, le problème est trivial dans les graphes *dynamiques* de degré au plus un et nous montrons qu'il est NP-Complet dans les graphes de degré au plus deux.

Notre deuxième contribution est de donner les bornes inférieures et supérieures pour la durée de l'agrégation des données dans un graphe dynamique.

#### A.3.1.1 Complexité de l'Agrégation de Données en Temps Minimum

**Dans les Réseaux de Capteurs Statiques** Nous améliorons le résultat de Chen *et al.* [CHZ05] pour le rendre plus précis et optimal. Notre résultat s'applique aux graphes d'intersections de disques de rayon 1/2 à coordonnées entières et de degré au plus trois (le résultat de Chen *et al.* est vrai dans les grilles partielles, qui sont un sur-ensemble strict des graphes d'intersections de disques que nous considérons, de degré au plus quatre). Par ailleurs, le problème est facile à résoudre dans le cas des graphes statiques de degré au plus deux. En effet, en posant $\varepsilon$ l'excentricité du nœud puits et $n$ le nombre de nœuds total, si $n$ est impair et $\varepsilon = (n-1)/2$ (le graphe est soit un cycle, soit une ligne de centre $s$) alors $ADTM_{Opt}(G,s) = \varepsilon + 1$, sinon $ADTM_{Opt}(G,s) = \varepsilon$.

Ainsi, le résultat suivant est optimal en fonction du degré maximum du graphe.

**Theorem A.1**

*Le problème ADTM est NP-Complet dans les réseaux de capteurs statiques de degré au plus trois.*

**Dans les Réseaux de Capteurs Dynamiques** Nous utilisons une approche différente pour démontrer la NP-Complétude dans les graphes évolutifs (qui modélisent les réseaux de capteurs dynamiques) de degré au plus deux. Encore une fois, le cas des graphes de degré au plus un est trivial étant donnée l'absence de collision.

**Theorem A.2**

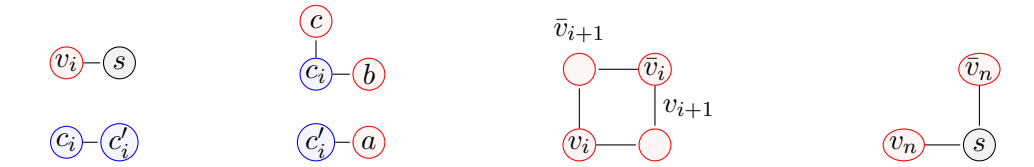
*Le problème ADTM est NP-Complet dans les réseaux de capteurs dynamiques de degré au plus deux.*

**Démonstration :** La preuve se fait par réduction depuis le problème 3−SAT. À partir d'une instance $\varphi$ du problème 3−SAT, avec $n$ variables $v_1, \ldots, v_n$ et $m$ clauses $c_1, \ldots, c_m$, on construit le graphe évolutif $G(V, \{E_i\}_{i\in\mathbb{N}})$ de la manière suivante. L'ensemble des nœuds est composé d'un nœud puits $s$, des littéraux de $\varphi$ et des clauses de $\varphi$ en deux exemplaires :

$$V = \{s\} \bigcup_{1\le i\le n} \{v_i\} \cup \{\bar{v}_i\} \bigcup_{1\le i\le m} \{c_i\} \cup \{c'_i\}$$

Soit $t_f = 3n + m$. On décompose l'intervalle de temps $[1, t_f]$ en trois périodes $T_1$, $T_2$ et $T_3$ (voir figure A.3). Durant $T_1 = [1, 2n]$, on a pour tout $i \in [1, n]$ :

$$E_{2i-1} = \{(v_i,\, s)\,,\, (c_i,\, c'_i)\}\,, \quad E_{2i} = \{(\bar{v}_i,\, s)\}$$




(a) configuration durant $T_1$ (b) configuration durant $T_2$ (c) configuration durant $T_3$ (d) dernière configuration de $T_3$

FIGURE A.3 – Configuration des nœuds (les clauses sont bleues et les littéraux sont rouges).

Durant $T_2 = [2n+1, 2n+m]$, on a pour tout $j \in [1, m]$ :

$$E_{2n+j} = \left\{ \left(c'_j,\, a\right), \left(c_j,\, b\right), \left(c_j,\, c\right) \right\} \quad \text{avec } a,\, b \text{ et } c \text{ les littéraux de la clause } c_j$$

Durant $T_3 = [2n+m+1, 2n+m+n]$, on a pour tout $i \in [1, n-1]$ :

$$E_{2n+m+i} = \left\{ \left(v_i,\, v_{i+1}\right), \left(\bar{v}_i,\, v_{i+1}\right), \left(v_i,\, \bar{v}_{i+1}\right), \left(\bar{v}_i,\, \bar{v}_{i+1}\right) \right\}$$

et

$$E_{2n+m+n} = \left\{ \left(v_i,\, s\right), \left(\bar{v}_i,\, s\right) \right\}$$

Durant $T_3$, une variable ou sa négation (de manière exclusive) peut transmettre ses données au littéral suivant jusqu'au nœud puits, de sorte que l'ensemble des littéraux qui transmettent durant cette période, en leur assignant la valeur *vrai*, est vu comme une interprétation de la formule $\varphi$. On peut montrer que cette interprétation *satisfait* la formule $\varphi$ si et seulement si la durée de l'agrégation des données de $G$ vers $s$ de départ 1 est $t_f$. ■

### A.3.1.2 Borne Inférieure et Borne Supérieure

Parmi tous les trajets qui commencent après l'instant $t$, un trajet $\mathcal{J}$ est dit *principal* s'il minimise la durée $arrivee(\mathcal{J}) - t$. Dans le cas statique, un trajet principal coïncide avec la notion de plus court chemin.

L'excentricité du nœud puits $s$ peut être définie comme la hauteur commune des arbres enracinés en $s$ de plus courts chemins. Elle est une borne inférieure dans le cas statique [CHZ05] car les arbres de plus courts chemins définissent autant de manières possibles d'agréger les données, lorsque les collisions ne sont pas prises en compte. C'est pourquoi, nous définissons un *arbre des trajets principaux* (ATP) enraciné en $s$ de départ $t$ comme étant un arbre enraciné en $s$ où chaque branche correspond à un trajet principal de la feuille vers $s$ commençant après $t$. À la manière d'un arbre de plus courts chemins dans le cas statique, un ATP peut servir de base pour définir un ordonnancement pour l'agrégation des données dans un graphe évolutif. Ainsi, la durée commune des ATP, enracinés en $s$ et de départ $t$, notée $ATPD(G, s, t)$, est une borne inférieure de $ADTM_{Opt}(G, s, t)$.

Dans le cas statique, un arbre de plus courts chemins reste toujours valide. Cela permet, pour résoudre une collision, de retarder d'une unité de temps une transmission. À chaque nœud d'un arbre de plus courts chemins, il y a au maximum $\Delta - 1$ collisions, où $\Delta$ est le degré maximum du graphe. D'où, $(\Delta - 1)\varepsilon$ est un majorant de $ADTM_{Opt}(G, s)$ [CHZ05]. Dans le cas dynamique, un ATP n'est valable que pour un temps donné, et rien ne certifie qu'une transmission peut être retardée d'une unité de temps car le lien de communication peut disparaitre. En fait, l'existence d'un ATP enraciné en $s$ ne certifie pas l'existence d'une solution au problème ADTM.

Pour trouver une condition suffisante d'existence d'une solution, on remarque qu'un ATP permet au moins à un nœud donné de transmette sa donnée jusqu'au nœud puits. Une condition suffisante pour l'existence d'une solution au problème ADTM est alors la donnée de $(n - 1)$ ATP consécutifs *i.e.*, une suite d'ATP dont la fin d'un ATP précède le début de l'ATP suivant. Cette observation permet aussi d'assurer que la durée de $(n - 1)$ ATP consécutifs de départ $t$, notée $ATPD^{n-1}(G, s, t)$ est un majorant de $ADTM_{Opt}(G, s, t)$. Ce majorant est atteint par le graphe $G(V, \{V \times V\}_{i\in\mathbb{N}})$ de degré $n - 1$ (et elle n'est pas atteinte par un graphe de degré maximum $n - 2$ ou inférieur).

**Theorem A.3**

*Soit $G$ un graphe évolutif, $s$ un nœud de $G$ et $t$ un temps, on a :*

$$ATPD^{n-1}(G, s, t) \geq ADTM_{Opt}(G, s, t) \geq ATPD(G, s, t)$$

#### A.3.1.3 Conclusion

Nous avons étudié le problème de l'agrégation de données dans les graphes évolutifs, qui modélisent les réseaux de capteurs dont la topologie évolue avec le temps. Nous avons donné les conditions optimales sur le degré maximum du graphe pour que le problème soit NP-Complet. De plus nous avons donné les premières bornes pour la durée d'agrégation des données. La borne inférieure est vraie quel que soit le degré maximum du graphe. Cependant, la borne supérieure n'est pas le plus petit des majorants pour les graphes de degré maximum $n - 2$ ou inférieur. Des travaux futurs consisteraient à trouver une borne supérieure qui resterait optimale quel que soit le degré maximum du graphe.

### A.3.2 Agrégation en Ligne Dans un Réseau Dynamique Arbitraire

Dans le chapitre 6, nous étudions le problème de l'agrégation de données distribuée dans le cas où le réseau est dynamique, *i.e.*, lorsque l'ensemble des liens permettant la transmission de données entre les nœuds varie dans le temps. Notre modèle s'appuie sur le modèle des graphes évolutifs pour représenter l'évolution du réseau. Plus précisément, l'évolution de la topologie d'un réseau peut se modéliser par une séquence de graphes statiques, où chaque graphe représente l'état du réseau à un instant donné. Dans notre modèle, où les objets interagissent deux à deux, un

adversaire, en plus de choisir la séquence de graphes statiques, choisit pour chaque graphe statique l'ordre dans lequel ces interactions se produisent. C'est pourquoi nous supposons directement que chaque graphe de la séquence ne contient qu'un seul lien, c'est-à-dire que le réseau dynamique est modélisé par une séquence d'interactions (impliquant deux nœuds) représentant les interactions entre les nœuds au cours du temps. [1]

**Travaux connexes.** On peut classer les résultats existants sur l'agrégation de données en deux parties, selon qu'ils considèrent ou non que les collisions lors des transmissions sont prises en charge par la couche MAC ou pas. Dans le cas où les collisions ne sont pas gérées par la couche MAC, le problème est NP-complet [BT15], même dans des réseaux de capteurs statiques (resp. dynamiques) de degré maximum 3 (resp. 2). Dans le cas des réseaux de capteurs statiques, de nombreux travaux ont abouti à des algorithmes d'approximation distribués performants. Cependant, le seul algorithme d'approximation existant pour les réseaux dynamiques est centralisé et suppose la connaissance complète de l'évolution future du réseau.

Dans le cas où les collisions sont gérées par la couche MAC, on retiendra particulièrement les travaux de Cornejo et al. [CGN12] où un ensemble de nœuds doit agréger des jetons. Aucun nœud ne joue le rôle de puits et le but est de minimiser le nombre de nœuds possédant au moins un jeton à la fin du temps imparti. Une solution pour laquelle un nœud particulier collecterait *tous* les jetons permettrait de résoudre le problème que nous considérons ici.

**Modèle.** Un réseau dynamique est modélisé par un couple $(V, I)$ où $V$ est l'ensemble des nœuds et $I = (I_t)_{t\in\mathbb{N}}$ est la séquence des interactions impliquant une paire de nœuds. Dans la suite, on notera $n$ le nombre de nœuds et $s \in V$ le nœud puits.

On suppose que les nœuds ont un identifiant unique, une mémoire et une capacité de calcul illimitée. Cependant, on pourra considérer des nœuds sans mémoire persistante. Initialement, chaque nœud de $V$ possède une donnée. Lors d'une interaction $I_t = \{u, v\}$, si à la fois $u$ et $v$ possèdent une donnée, l'un des nœuds peut transmettre sa donnée à l'autre, qui l'agrège avec sa propre donnée. Après avoir transmis sa donnée, un nœud ne peut plus ni recevoir de donnée, ni transmettre à nouveau.

Un algorithme distribué d'agrégation de données (DODA) est un algorithme qui, étant donnée une interaction $I_t = \{u, v\}$ et sa date d'occurrence $t$, retourne $u$, $v$, ou $\perp$. Si les deux nœuds $u$ et $v$ possèdent une donnée avant l'interaction, et si l'algorithme retourne un nœud ($u$ ou $v$), alors ce dernier agrège la donnée de l'autre nœud. Si l'algorithme retourne $\perp$, aucune donnée n'est transmise. Lors de son exécution, un DODA peut utiliser sa mémoire interne, la modifier, et utiliser des informations concernant les nœuds. Par défaut, un nœud $u$ a pour information son identifiant

1. Cette modélisation des interactions se rapproche de celle utilisée pour les protocoles de populations, mais notre modèle de nœud et d'adversaire est très différent de ceux utilisés dans le contexte de protocoles de populations.

$u.ID$ et s'il est ou non le puits $u.isSink$. Certains algorithmes nécessitent d'autres informations (comme la séquence des interactions futures qui impliquent le nœud $u.future$, ou le moment de la prochaine interaction entre $u$ et le puits $u.meetTime$). L'ensemble des DODA utilisant les informations supplémentaires $\mathfrak{i}_1, \mathfrak{i}_2, \ldots$ est noté $\mathcal{D}_{\mathsf{ODA}}(\mathfrak{i}_1, \mathfrak{i}_2, \ldots)$. Le sous ensemble de $\mathcal{D}_{\mathsf{ODA}}(\mathfrak{i}_1, \mathfrak{i}_2, \ldots)$ des algorithmes sans mémoire (*i.e.*, n'utilisant pas de mémoire persistante) est noté $\mathcal{D}^{\emptyset}_{\mathsf{ODA}}(\mathfrak{i}_1, \mathfrak{i}_2, \ldots)$

On suppose qu'un adversaire construit la séquence des interactions. On distingue plusieurs types d'adversaire : les adversaires sans mémoire (qui construisent la séquence avant le début de l'exécution), et les adversaires aléatoires (qui choisissent les interactions aléatoirement).

Par manque de place, dans ce papier nous considérons uniquement des séquences d'interactions $I = (I_t)_{t\in\mathbb{N}}$ telles que l'agrégation de données est toujours possible sur tout suffixe $(I_t)_{t>k}$ (au moins par un algorithme centralisé qui connaît le futur). De telles séquences sont dites $\mathcal{D}_{\mathsf{ODA}}$-récurrentes. Dans ce contexte, on s'intéresse aux conditions d'existence d'un DODA qui termine pour toute séquence $\mathcal{D}_{\mathsf{ODA}}$-récurrente. Certains preuves sont disponibles dans le rapport technique associé [**?**].

### A.3.2.1 Face à un Adversaire Sans Mémoire

Un adversaire sans mémoire connaît le code de l'algorithme, mais il doit construire la séquence d'interactions sans connaitre les choix (possiblement probabilistes) effectués par l'algorithme. Cela revient à générer, avant même le début de l'exécution, l'intégralité de la séquence d'interactions. Dans le cas d'un algorithme déterministe, cet adversaire est équivalent à un adversaire pouvant s'adapter aux choix faits par l'algorithme (ces choix peuvent être prédit avant le début de l'exécution). Face à un algorithme probabiliste, la séquence de nombres aléatoires n'étant pas connue de l'adversaire, les choix de l'algorithme deviennent imprévisibles. Il est facile de trouver un adversaire empêchant tout algorithme déterministe d'agréger les données du réseau en temps fini. Notre premier théorème généralise ce résultat à tout algorithme probabiliste.

**Theorem A.4**

*Pour tout algorithme probabiliste $A \in \mathcal{D}_{\mathsf{ODA}}$, il existe un adversaire sans mémoire générant une séquence d'interactions $\mathcal{D}_{\mathsf{ODA}}$-récurrente $I$ telle que l'exécution de $A$ sur $I$ ne termine pas, avec forte probabilité (avec une probabilité supérieure à $1 - o(1)$)*

**Démonstration :** Soit $V = \{s, u_0, \ldots, u_{n-2}\}$. Dans la suite, les indices des nœuds sont modulo $n-1$, *i.e.*, pour tout $i \in \mathbb{N}$ $u_i = u_j$, avec $i \equiv j\,[n-1]$. Soit $I^\infty$ défini pour tout $i \in \mathbb{N}$ par $I^\infty_i = \{u_i, s\}$. Soit $I^l$ le préfixe de longueur $l \in \mathbb{N}$ de la séquence $I^\infty$. Pour tout $l \in \mathbb{N}$, l'adversaire peut calculer la probabilité $P_l$ qu'aucun nœud ne transmette sa donnée pendant l'exécution de $A$ sur $I^l$. La suite $(P_l)_{l>0}$ est décroissante et minorée, donc converge vers une limite $\mathcal{P} \geq 0$. Pour un $l$ donné, si $P_l \geq 1/n$, alors on peut montrer qu'au moins deux nœuds ont une probabilité de ne pas transmettre supérieure à $n^{-\frac{1}{n-2}} = 1 - O(\frac{1}{n})$ (pendant l'exécution de $A$ sur $I^l$).

Donc si $\mathcal{P} \geq 1/n$, alors il existe un nœud qui ne transmet jamais, et l'exécution de $A$ sur $I^\infty$ ne termine pas, avec forte probabilité.

Sinon, soit $l_0 \in \mathbb{N}$ le plus petit indice tel que $P_{l_0} < 1/n$. Avec forte probabilité, au moins un nœud transmet pendant l'exécution de $A$ sur $I^{l_0}$. De plus, $P_{l_0-1} \geq 1/n$, donc deux nœuds $u_d$ et $u_{d'}$ n'ont pas transmis pendant l'exécution de $A$ sur $I^{l_0-1}$, avec une probabilité supérieure à $n^{-\frac{1}{n-2}}$ (si $l_0 = 0$, on peut choisir $\{u_d, u_{d'}\} = \{u_1, u_2\}$). On a $u_d \neq u_{l_0}$ ou bien $u_{d'} \neq u_{l_0}$. Sans perte de généralité, on suppose que $u_d \neq u_{l_0}$. Ainsi la probabilité que $u_d$ transmette pendant l'exécution de $A$ sur $I^{l_0-1}$ est la même que sur $I^{l_0}$.

Soit $I'$ la séquence finie d'interactions $(\{u_d, u_{d+1}\}, \{u_{d+1}, u_{d+2}\}, \ldots, \{u_{d-2}, u_{d-1}\}, \{u_{d-1}, s\})$. Cette séquence est construite de sorte que $u_d$ doit envoyer sa donnée jusqu'à $s$ en utilisant un chemin qui contient tous les autres nœuds. On construit alors la séquence d'interactions $I$ comme étant la séquence $I^{l_0}$ suivie de $I'$ répétée une infinité de fois. Ainsi, après l'exécution de $A$ sur $I^{l_0}$, avec forte probabilité, $u_d$ contient toujours sa donnée et au moins un autre nœud a déjà transmit ça donnée. Donc, lors de l'exécution de $A$ sur $I'$, la donnée de $u_d$ ne peut plus atteindre $s$ (le chemin que doit emprunter la donnée de $u_d$ contient tous les autres nœuds, dont au moins un qui a déjà transmis et ne peut plus transmettre). L'exécution de $A$ sur $I$ ne termine pas. Par ailleurs, comme la séquence $I'$ est suffisante pour agréger toutes les données de $V$, $I$ est $\mathcal{D}_{\mathsf{ODA}}$-récurrente. ■

#### A.3.2.2 Face à un Adversaire Probabiliste

On considère maintenant un adversaire qui choisit les interactions aléatoirement, uniformément parmi toutes les interactions possibles. Chaque interaction $I = \{u, v\}$, $u, v \in V$, a une probabilité $\frac{2}{n(n-1)}$ d'être choisie par l'adversaire. Comme l'apparition d'une interaction ne dépend pas des interactions précédentes, les deux algorithmes que nous proposons dans cette section fonctionnent sans mémoire, *i.e.*, la sortie de nos algorithmes dépend uniquement de l'interaction et des informations concernant les nœuds impliqués. De plus, on suppose que les nœuds proposés à l'algorithme sont ordonnés selon leur identifiant, et qu'ils contiennent tous les deux une donnée (sinon l'algorithme retourne $\perp$).

— *Gathering* ($\mathcal{GA} \in \mathcal{D}^{\emptyset}_{\mathsf{ODA}}$) : Un nœud transmet sa donnée lorsqu'il est connecté à $s$ ou à tout autre nœud possédant une donnée. $\mathcal{GA}$ associe à $(u_1, u_2, t)$ le nœud $u_i$ si $u_i.isSink$, sinon il associe $u_1$.

— *Waiting Greedy* de paramètre $\tau \in \mathbb{N}$ ($\mathcal{WG}_\tau \in \mathcal{D}^{\emptyset}_{\mathsf{ODA}}(meetTime)$) : le nœud avec le plus grand $meetTime$ transmet si son $meetTime$ est plus grand que $\tau$ :

$$\mathcal{WG}_\tau : (u_1, u_2, t) = \begin{cases} u_1 \text{ si } m_1 < m_2 \wedge \tau < m_2 \\ u_2 \text{ si } m_1 > m_2 \wedge \tau < m_1 \\ \perp \text{ sinon} \end{cases} \begin{array}{l} m_1 = u_1.meetTime(t) \\ m_2 = u_2.meetTime(t) \end{array}$$

**Bornes Inférieures.** Nous étudions la durée minimum d'agrégation des données qu'il est possible d'obtenir. Sans connaissance supplémentaire, un algorithme est

limité par la durée nécessaire pour agréger la donnée du dernier nœud (autre que $s$). En effet, lorsqu'il ne reste qu'un nœud $u \neq s$ possédant une donnée, la probabilité que $u$ puisse transmettre sa donnée est égale à la probabilité d'occurrence de l'interaction $\{u, s\}$, *i.e.*, $\frac{2}{n(n-1)}$. L'espérance d'un tel événement est $\frac{n(n-1)}{2}$, d'où le théorème suivant :

**Theorem A.5**

*Tout algorithme $A \in \mathcal{D}_{\mathsf{ODA}}()$ termine en moyenne en $\Omega(n^2)$ interactions.*

Pour les algorithmes utilisant des informations supplémentaires, nous allons minorer le nombre d'interactions par le nombre d'interactions requis par un algorithme centralisé connaissant toute la séquence d'interactions. Pour cela on peut remarquer que l'agrégation de données correspond à un broadcast en prenant le temps qui s'écoule dans le sens inverse. On peut montrer qu'un broadcast s'effectue avec forte probabilité en $\Theta(n \log(n))$ interactions, d'où le théorème suivant.

**Theorem A.6**

*Tout algorithme $A \in \mathcal{D}_{\mathsf{ODA}}(\texttt{séquence d'interactions})$ termine en $\Theta(n \log(n))$ interactions avec forte probabilité.*

**Performance des Algorithmes** Pour l'algorithme $\mathcal{GA}$, on définit la v.a. $X_i$ du nombre d'interactions entre la $(i-1)$-ème et la $i$-ème transmission. $X_i$ suit une loi géométrique de paramètre $\frac{(n-i+1)(n-i)}{n(n-1)}$. La durée moyenne d'exécution de $\mathcal{GA}$ est donc la somme $\sum_1^{n-1} E(X_i) = n(n-1)\sum_{i=1}^{n-1} \frac{1}{i(i+1)} \in O(n^2)$. $\mathcal{GA}$ est donc optimal dans $\mathcal{D}_{\mathsf{ODA}}()$.

Pour les performances de $\mathcal{WG}_\tau$ on pose $f$ une fonction telle que $f(n) = \omega(1)$ et $f(n) = o(n)$.

**Theorem A.7**

*L'algorithme $\mathcal{WG}_\tau$ avec $\tau = nf(n) + n^2 \log(n)/f(n)$ termine en $\tau$ interactions, avec forte probabilité.*

**Démonstration :** Soit $L$ l'ensemble des nœuds qui interagissent avec $s$ entre les instants $\tau' = n^2 \log(n)/f(n)$ et $\tau$, et soit $L^c$ son complémentaire (on suppose $s \in L$). Lors d'une interaction avant l'instant $\tau'$, *(i)* si les deux nœuds sont dans $L$, rien ne se produit (car leur *meetTime* est inférieur à $\tau$), *(ii)* si les deux nœuds sont dans $L^c$, l'un des nœuds transmet à l'autre, *(iii)* sinon le nœud dans $L^c$ transmet sa donnée au nœud dans $L$. Une manière possible d'agréger toutes les données est que chaque nœud dans $L^c$ interagisse au moins une fois avec un nœud dans $L$ avant l'instant $\tau'$. Ensuite les nœuds de $L$ transmettent directement au puits entre $\tau'$ et $\tau$. On peut montrer qu'avec forte probabilité, $nf(n)$ interactions sont suffisantes pour que $f(n)$ nœuds interagissent avec le puits, donc $L$ est de cardinal $f(n)$ avec forte probabilité. De plus, lorsque $L$ contient au moins $f(n)$ nœuds, alors il faut au plus $n^2 \log(n)/f(n)$ interactions pour que chaque nœud de $L^c$ interagisse au moins une fois avec un nœud de $L$. Donc en choisissant $\tau = nf(n) + n^2 \log(n)/f(n)$, l'algorithme termine avant l'instant $\tau$ avec forte probabilité.

■

Le paramètre $\tau = 2n^{3/2}\sqrt{\log(n)}$, obtenu en prenant la fonction $f : n \mapsto \sqrt{n\log(n)}$ qui minimise la valeur de $nf(n) + n^2\log(n)/f(n)$, semble donc être optimal pour l'algorithme $\mathcal{WG}_\tau$. On peut montrer que c'est le cas, et même que l'algorithme correspondant $\mathcal{WG}_\tau$, avec $\tau = 2n^{3/2}\sqrt{\log(n)}$, est asymptotiquement optimal parmi tout les algorithmes dans $\mathcal{D}_{\mathsf{ODA}}(meetTime)$ (même ceux avec mémoire).

#### A.3.2.3 Conclusion

Nous avons présenté un nouveau modèle pour l'étude du problème de l'agrégation de données distribuée dans les réseaux dynamiques. Après avoir présenté le résultat d'impossibilité dans le cas général, face à un adversaire sans mémoire, nous avons donné deux algorithmes optimaux face à un adversaire aléatoire, le premier fonctionnant sans aucune connaissance supplémentaire, le deuxième utilisant l'information $meetSink$.

## A.4 L'Estimation de la Durée de Vie des Batteries

De nombreuses applications des réseaux de capteurs, notamment dans le domaine médical, requièrent des appareils capables de fonctionner sur de longues périodes, typiquement, de plusieurs jours à plusieurs années. Cependant, la taille des capteurs étant limitée, il est crucial de simuler fidèlement la batterie et la consommation des composants pour choisir les paramètres (notamment, le ratio entre la durée d'éveil et la durée de sommeil) qui répondront aux exigences de l'application concernant la durée de vie du réseau.

Dans la troisième partie de cette thèse, nous présentons un nouveau modèle pour l'évaluation de la durée de vie des capteurs dans les réseaux de capteurs sans fil (Section A.4.1). Puis nous utilisons ce modèle pour faire une étude comparative de plusieurs protocoles de broadcast efficaces en énergie (Section A.4.2)

### A.4.1 WiSeBat : Modèle pour une Évaluation Réaliste de la Durée de Vie des Batteries

**Travaux Connexes** Même si la définition de la durée de vie d'un réseau dépend de l'application [DD09], elle est déterminée à partir de la durée de vie de chacun des capteurs. L'estimation de la durée de vie d'un appareil repose à la fois sur le modèle de batterie et le modèle de consommation [DDV+14].

De manière simplifiée, les simulateurs de réseaux populaires comme NS2/3 ou WSNET prennent en compte la consommation de la radio uniquement, couplée à un modèle de batterie linéaire. Cela revient à multiplier le nombre de bits transmis et de bits reçus par la consommation de la radio en mode TX et RX respectivement. Ce modèle très simple, utilisé par défaut, n'est cependant pas du tout réaliste, comme nous le montrons ci-après.

D'autres solutions comme SENSE ou PowerTOSSIMz prennent en compte chaque composant de l'appareil, couplé a un modèle de batterie non linéaire. Même si cette approche peut paraître plus réaliste, elle ne tient pas compte des variations de voltage de la batterie, qui elles non plus ne sont pas linéaires. Cela entraine des imprécisions, notamment pour déterminer à quel moment un nœud s'arrête de fonctionner.

**Résumé des Contributions** Nous adaptons un modèle existant de batterie pour les réseaux de capteurs sans fil. Nous le couplons avec un modèle de consommation prenant en compte chaque composant du capteur pour former le module complet que nous appelons WiSeBat, dont l'implémentation de départ est intégrée au simulateur WSNET. Enfin, nous le validons par des mesures sur des capteurs réels qui montrent que le module WiSeBat (faisant référence à la fois au modèle de batterie et de consommation) surpasse le modèle par défaut de WSNET.

#### A.4.1.1 Le Module WiSeBat

**Modèle de Consommation** Dans les réseaux de capteurs sans fil, la radio qui n'est pas constamment allumée, n'est pas nécessairement le composant qui utilise le plus la batterie. C'est pourquoi WiSeBat offre la possibilité lors de l'initialisation de la simulation d'enregistrer un ensemble de composants $\mathcal{X}$ (par exemple : les capteurs, le micro-contrôleur, les LEDs, etc). Chaque composant $x \in \mathcal{X}$ est associé à une fonction qui donne sa consommation $i_x(m)$ en mA en fonction du mode de fonctionnement $m$ dans lequel il se trouve. Lors de la simulation, la couche application peut mettre à jour le mode de fonctionnement de chaque composant enregistré (le mode de la radio étant directement modifié par la couche MAC). Si on l'associe à une batterie au comportement linéaire, la quantité d'énergie consommée en mAh par les composants est donnée par la formule suivante (où $\mathcal{M}_x$ est l'ensemble des états possibles du composant $x$ et $T_x(m)$ est le temps, en heures, passé dans l'état $m$) :

$$C = \sum_{x \in \mathcal{X}} \sum_{m \in \mathcal{M}_x} i_x(m) T_x(m)$$

**Modèle de Batterie** Le premier comportement non linéaire de la batterie pris en compte par WiSeBat est que lorsque le courant instantané augmente, la batterie se comporte comme si la capacité totale diminuait. En effet, à partir des données constructeur, on peut extraire une fonction décroissante $C_{eq} : i \mapsto C_{eq}(i)$ telle que si le courant est constant égal à $i$ alors la capacité mesurée est $C_{eq}(i)$. La capacité nominale $C_{nominal}$ de la batterie est en fait la limite en 0 de cette fonction (elle est généralement mesurée pour un courant faible appelé courant nominal). Ce comportement est capturé en utilisant la définition donnée par Dron *et al.* [DDV+14] de courant équivalent $i_{eq}(t)$ au courant total $i(t)$ fourni par la batterie au temps $t$ :

$$i_{eq}(i(t)) = \frac{C_{\text{nominal}}}{C_{eq}(i(t))} i(t) \tag{A.1}$$

En effet, si le courant consommé par les composants est constant égal à $i$, la batterie sera épuisée (*i.e.*, sera incapable de fournir du courant) après un temps $T$ tel que $i \times T = C_{eq}(i)$.

Le second comportement non linéaire de la batterie modélisé par WiSeBat est la variation de voltage au cours de sa décharge, en fonction du courant fourni. Les données du constructeur permettent de déduire les fonctions $E_{mf}$ et $R$ qui donnent la force électromotrice en Volt et la résistance interne en Ohm de la batterie en fonction de sa capacité résiduelle. Si au temps $t$ la batterie a une capacité résiduelle $C(t)$ et fournit un courant $i(t)$, alors le voltage $V(t)$ à ses bornes est donné par la formule suivante :

$$V(t) = E_{mf}(C(t)) - i(t)R(C(t))$$

Ce voltage affecte les composants d'un capteur de deux manières. Premièrement, la consommation des composants dépend du voltage à ses bornes (*i.e.*, pour $x \in \mathcal{X}$, $i_x$ dépend en fait du mode de $x$ et de la tension). Deuxièmement chaque composant nécessite un voltage minimum pour fonctionner, appelé *cut-off*. C'est-à-dire qu'un capteur va cesser de fonctionner *avant* que la batterie soit totalement déchargée.

Grâce aux équations précédentes, le module WiSeBat met à jour les valeurs de $i(t)$, $C(t)$ et $V(t)$ de manière discrète à des instants $t_1, t_2, \ldots$ en s'appuyant sur les relations suivantes (où pour un composant $x$, $m_x(t)$ est son mode à l'instant $t$ et $i_x$ est la fonction de consommation en fonction du voltage et du mode) :

$$C(t_i) = C(t_{i-1}) - i_{eq}(i(t_{i-1}))\,(t_i - t_{i-1}) \tag{A.2}$$

$$V(t_i) = E_{mf}\,(C(t_i)) - i(t_{i-1})R\,(C(t_i)) \tag{A.3}$$

$$i(t_i) = \sum_{x \in \mathcal{X}} i_x(V(t_i), m_x(t_i)) \tag{A.4}$$

Les relations sont liées de manière circulaire car le voltage dépend de l'intensité et inversement. C'est pourquoi, il faut que plusieurs mises à jours soient effectuées après chaque changement de mode, jusqu'à ce que les valeurs se stabilisent. De cette manière, l'erreur entre le modèle et les valeurs calculées de $C(t)$, $V(t)$, et $i(t)$ peut être maintenue en dessous d'une valeur arbitraire $\varepsilon > 0$.

#### A.4.1.2 Validation Expérimentale

Pour évaluer WiSeBat nous avons mesuré la durée de vie d'un capteur réel avec une application et une architecture réaliste, et nous l'avons comparé à la durée de vie simulée par WiSeBat dans le simulateur WSNET. Pour cela nous avons implémenté dans WSNET l'application et l'architecture présente dans le capteur.

| Composant | Mode | Courant |
|---|---|---|
| Micro-controller | Run | 10 mA |
| voltage *cut-off* : 1.65V | Sleep | 2.4 uA |
| Radio transceiver | Tx | 7.7 mA |
| voltage *cut-off* : 1.8V | Rx | 5.7 mA |
| | Idle | 1.7 mA |
| | Sleep | 1.1 uA |
| Capteur de pression | Init | 42 mA |
| voltage *cut-off* : 1.71V | Read | 5 mA |
| | Sleep | 0.5 uA |
| Led | On | 7.6 mA |
| Gestionnaire d'alimentation | Éfficacité : 95 - 99% | |
| voltage *cut-off* : 2.4V | | |

FIGURE A.4 – Caractéristiques des composants

Les composants du capteur ainsi que leur consommation sont présentés dans le tableau A.4. Les résultats donnés dans la figure A.5 représentent la durée de vie simulée des capteurs (fournie par WiSeBat et par le modèle par défaut de WSNET, respectivement) par rapport au capteur réel, dans deux scénarios (une transmission toutes les secondes et une transmission toutes les dix secondes). On remarque que le modèle par défaut de WSNET, utilisant une batterie au comportement linéaire et ne prenant en compte que la radio, surestime la durée de vie du capteur de plus de 2600%. Pourtant, ce modèle est très présent dans la littérature pour évaluer les performances énergétiques des protocoles réseaux. L'utilisation d'un modèle de batterie au comportement non linéaire améliore significativement les estimations, puisque la durée de vie simulée n'est plus que 7 fois supérieure à la durée de vie mesurée sur le capteur réel. Pour finir, la prise en compte de tous les composants (ce qui correspond au module complet WiSeBat) permet alors de sous-estimer la durée de vie avec une erreur comprise entre 4% et 14%. C'est un point important pour la production effective des capteurs : une sur-estimation aboutit à l'utilisation de batteries moins efficaces dans les capteurs menant à une défaillance largement prématurée du réseau (par rapport au temps estimé par la simulation). Au contraire, la sous-estimation de la durée de vie garantit que le capteur réel dure au moins le temps attendu. Lors de ces simulations, WiSeBat a représenté entre 0.1% et 30% du temps de simulation, ce qui encourage son utilisation systématique.

#### A.4.1.3 Conclusion

Nous proposons un module pour le simulateur WSNET permettant d'estimer efficacement la durée de vie des nœuds dans un réseaux de capteur sans fil avec un faible coût computationnel. Nous montrons que les résultats obtenus ont une précision pouvant aller jusqu'à 94%, contrairement au module par défaut qui surestime de plus de 2600% la durée de vie. Pour finir nous avons retrouvé des résultats connus

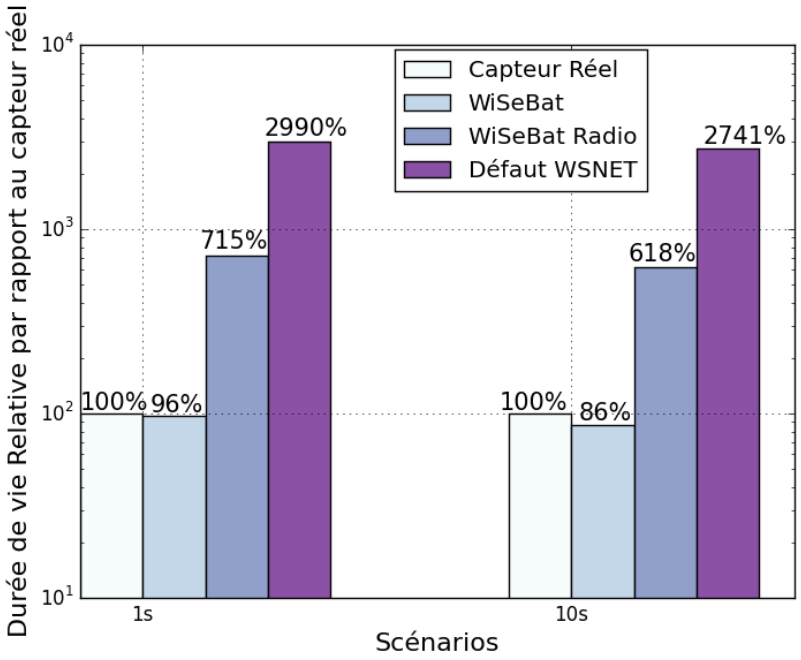



FIGURE A.5 – Précision des modèles (modèle par défaut de WSNET, WiSeBat en tenant compte uniquement de la radio, et WiSeBat prenant en compte tous composants) dans deux scénarios.

concernant l'efficacité énergétique de X-MAC, ContikiMAC et 802.15.4 ainsi que de nouveaux résultats précis concernant l'impact de la topologie sur la durée de vie des nœuds.

### A.4.2 Analyse Comparative des Protocoles de Broadcasts Efficaces en Énergie

Dans le Chapitre 9 de cette thèse, on s'intéresse à l'évaluation des performances de six protocoles de broadcast. Ces protocoles supposent que la radio est capable de faire varier sa puissance d'émission (et donc son rayon de transmission) afin d'économiser de l'énergie lorsque la destination est proche. Parmi les protocoles existant nous avons choisi les suivants :

— FLOOD : l'algorithme choisit toujours la puissance maximum.

— BIP : un algorithme centralisé offrant de très bonne performance théorique.

— LBIP : une version distribuée de l'algorithme BIP, moins performante en théorie.

— DLBIP : une version dynamique de l'algorithme LBIP, censée augmenter la durée de vie globale du réseau.

— RBOP et LBOP : deux algorithmes distribués au fonctionnement similaire, basés sur des propriétés locales du réseau utilisant RNG (graphe des voisins relatifs) et LMST (arbre couvrant local de poids minimal).

Dans les travaux précédents, les performances de ces protocoles sont données en utilisant un modèle énergétique idéal et dans un environnement idéal sans interférence. Cependant, on a vu que de telle hypothèses sont irréalistes et ne permettent pas d'évaluer efficacement les protocoles.

#### A.4.2.1 Configuration des Simulations

Nous avons réalisé des simulations en utilisant le simulateur WSNet [FCF07]. Chaque simulation contient 50 nœuds répartis aléatoirement dans un carré de côté compris entre 300 et 500 mètres. La pile réseau contient un des protocoles de test pour la couche routage, au dessus d'une couche MAC (2400MHz OQPSK 802.15.4 CSMA/CA unslotted [802a] ou ContikiMAC [Dun11]), au dessus d'une radio standard possédant une antenne omnidirectionnelle de modulation OPQSK. On suppose que l'atténuation est logarithmique. Le Tableau A.1 résume les propriétés des topologies générées pour la simulation (moyenne sur les 50 topologies générées pour chaque taille).

| Taille | Densité | Diamètre | Connectivité |
|---|---|---|---|
| 300 | 0.60 | 2.9 | 13.3 |
| 400 | 0.40 | 3.8 | 6.8 |
| 500 | 0.27 | 4.7 | 3.9 |
| 600 | 0.20 | 5.8 | 2.2 |
| 700 | 0.15 | 7.7 | 1.4 |
| 800 | 0.12 | 9.5 | 1.1 |

TABLE A.1 – Densité, diamètre, et connectivité moyens en fonction de la taille.

Le modèle énergétique utilisé est WiSeBat [BDBF+15] avec les données d'un capteur TMote Sky configuration dont les caractéristiques sont données dans le Tableau A.2).

#### A.4.2.2 Résultats et Discussion

Deux scénarios sont considérés : un broadcast est effectué toutes les 10 secondes par (a) le nœud ayant l'identifiant 0 (ne varie pas au cours de la simulation) (b) un nœud choisi aléatoirement.

**Broadcast d'une Seule Source** Parmi les résultats observés, on se concentre sur le nombre de nœuds moyen ayant effectivement les message envoyé par le nœud 0, en fonction de la taille de la zone de déploiement, présenté dans la Figure A.6. Contrairement à ce que l'on attendait, peu de protocoles offrent de bonnes performances. De plus, les performance dépendent de la couche MAC utilisée.

Avec ContikiMAC, BIP, DLBIP, et FLOOD offrent de bonnes performances quelle que soit la taille de la zone. LBIP et RBOP sont en dessous mais LBIP s'améliore lorsque la densité du graphe décroit.

| Radio Chipcon CC2420 | | CPU Texas Instruments MSP430 F1611 | |
|---|---|---|---|
| Tx 0dB | 17.4 mA | | |
| Tx -1dB | 16.5 mA | Run 8MHz 3V | |
| Tx -3dB | 15.2 mA | (with flash read) | 4mA |
| Tx -5dB | 13.9 mA | Sleep | 2.6$\mu$A |
| Tx -7dB | 12.5 mA | voltate cut-off | 2.7V |
| Tx -10dB | 11.2 mA | | |
| Tx -15dB | 9.9 mA | | |
| Tx -25dB | 8.5 mA | | |
| Rx | 19.7mA | | |
| Idle | 365$\mu$A | | |
| Sleep | 1$\mu$A | | |
| Volt. Regulator | 20$\mu$A | | |

TABLE A.2 – Caractéristique électrique de la TMote Sky

Avec 802.15.4, FLOOD et DLBIP montrent des résultats similaires. Cependant BIP est un des pires protocoles, avec RBOP et LBOP. Les performances de ces derniers s'améliorent avec des densités plus faibles mais restent bien inférieures avec ce qu'on observe en utilisant ContikiMAC.

Dans les deux cas, l'augmentation des performances avec la baisse de la densité du réseau s'explique par une diminution du nombre de collisions.

**Broadcast avec Source Multiple (Gossip)** Parmi les résultats observés, on s'intéresse au nombre de nœuds qui reçoivent effectivement le message en fonction du temps, présenté dans la Figure A.7 et Figure A.8). Dans ces figures, l'axe des abscisses représente le nombre de broadcasts effectués et l'axe des ordonnées le nombre de nœuds ayant reçu le message.

Avec ContikiMAC, BIP est le protocole qui broadcast le message à tous les nœuds pendant la plus grande durée. Ensuite FLOOD, et DLBIP offrent des performances légèrement moins bonnes mais qui restent acceptables (DLBIP est meilleur dans des graphes plus denses). Il est intéressant de voir que LBIP arrive a effectuer beaucoup plus de broadcasts dans des graphes denses, au prix d'environ 10% des nœuds qui ne reçoivent pas les messages. Finalement RBOP et LBOP sont bien en dessous des autres protocoles. On peut observer que dans des graphes creux, les performances de BIP, LBIP, DLBIP et FLOOD semblent converger vers environ 130000 broadcasts.

Avec 802.15.4 MAC on observe que, pour chaque protocole, la mort du réseau apparait brusquement. Aussi le nombre de broadcasts avec cette mort est similaire pour tous protocoles (autour de 6500 broadcasts). Cela est dû au fait que 802.15.4 MAC consomme la majorité de l'énergie, ce qui rend moins visibles les économies réalisées par nos protocoles. Le protocole FLOOD est le seul dont les broadcasts sont reçus par tous les nœuds, quelle que soit la densité du graphe. DLBIP à lui aussi de bonnes performances, et tous les autres protocoles sont loin derrière.

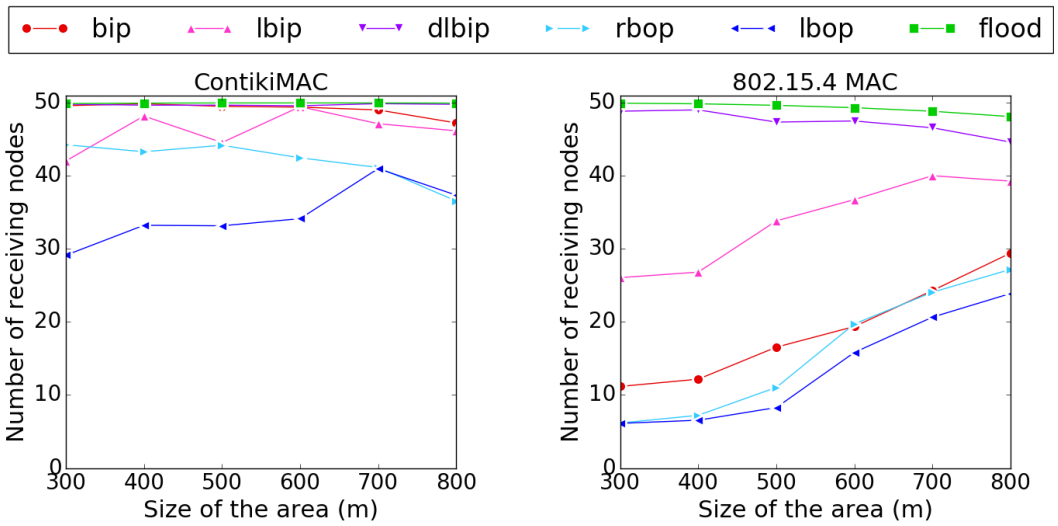



FIGURE A.6 – Nombre moyen de nœuds recevant le broadcast, en fonction de la taille de la zone

**Discussion** Nos résultats montrent que l'impact des collisions dues aux interférences est différent selon les protocoles de broadcast testés. L'impact négatif de ces collisions peut avoir plusieurs causes :

— Une cause directe *i.e.*, le protocole envoie peu de messages et chaque nœud ne possède qu'un seul voisin qui lui envoie ce message, augmentant ainsi le risque de perte en cas de collision. De plus, chaque collision risque d'isoler un sous-ensemble entier de nœuds de la source.

— Une cause indirecte *i.e.*, la manière dont le protocole sélectionne les nœuds qui retransmettrent le message peut accentuer le nombre de collisions, particulièrement à cause du problème du terminal caché.

La première cause peut expliquer les bonnes performances du protocole FLOOD. En effet, FLOOD ne limite pas le nombre de nœuds qui retransmettent le message et ainsi augmente la probabilité qu'au moins un voisin de chaque nœud transmette sans collision. De plus, l'augmentation du nombre de transmissions, et donc d'interférences, ne semble pas impacter le protocole FLOOD. On peut remarquer qu'il y a de toute façon une limite pour la quantité d'interférence dans le réseau car la couche retarde les transmission lorsque le canal est occupé. Ainsi, l'augmentation du nombre de transmissions impacte le délai de transmission mais pas forcément la proportion de message reçu avec succès. Si cette proportion se stabilise, augmenter le nombre de transmission a pour effet d'augmenter la probabilité qu'au moins un message soit reçu sans collision.

Les performances remarquables du protocole FLOOD confirment l'utilité en pratique de la redondance, et *cela n'implique pas forcément une consommation énergétique supérieure.*

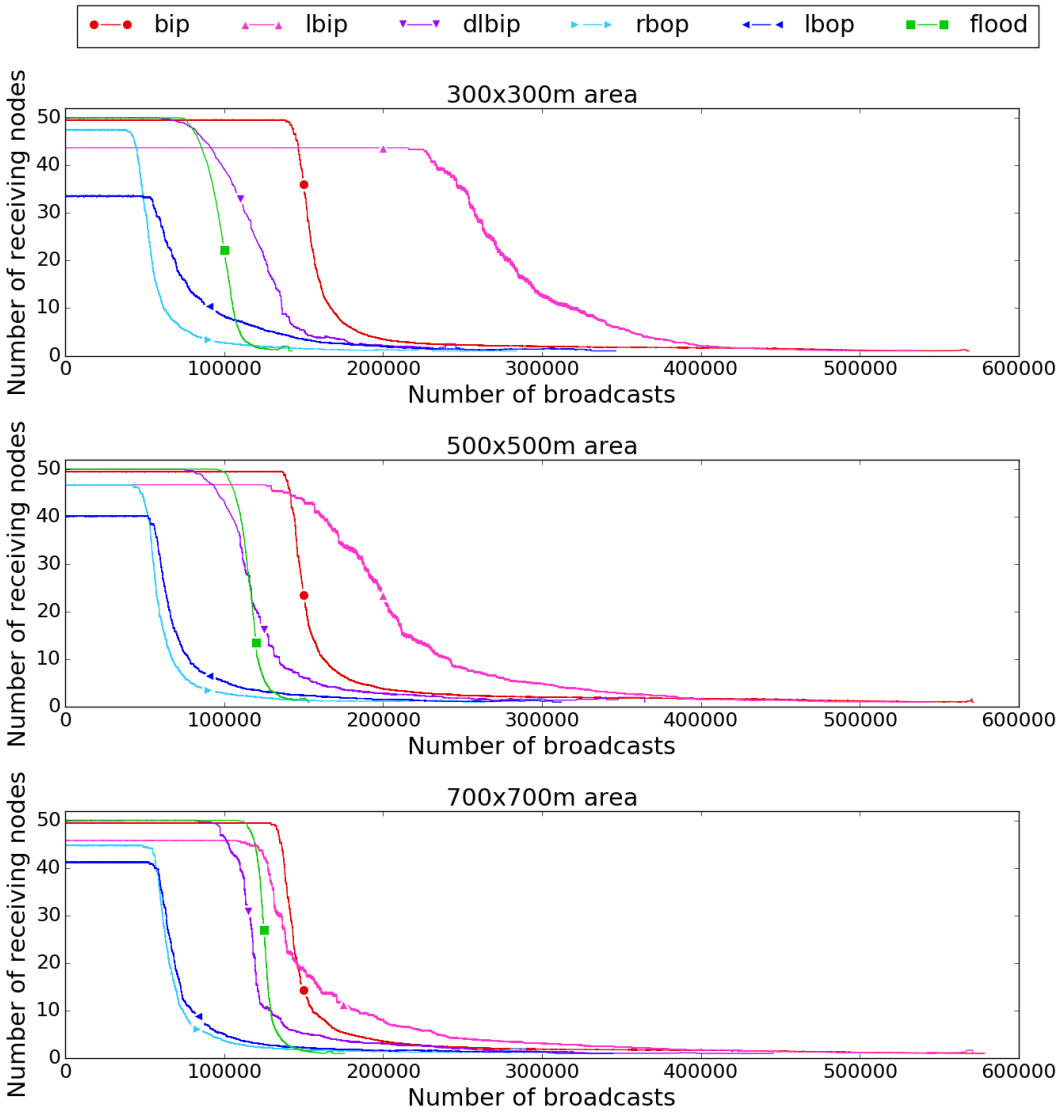



FIGURE A.7 – Nombre de nœuds recevant le message en fonction du temps, avec ContikiMAC.

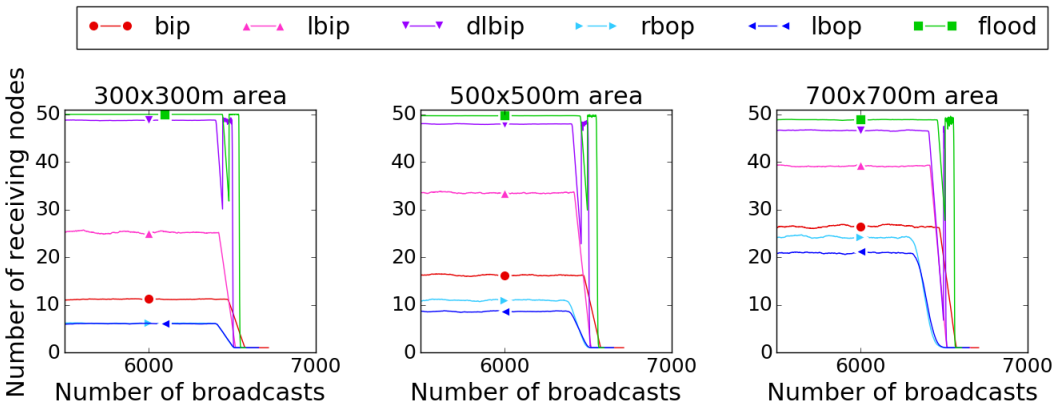



FIGURE A.8 – Nombre de nœuds recevant le message en fonction du temps, avec 802.15.4 MAC.

#### A.4.2.3 Conclusion

Nous avons utilisé notre modèle énergétique WiSeBat pour comparer les performances de six protocoles représentatifs de la littérature. Nous avons montré que les performances obtenues dans un environnement réaliste ne sont pas celle attendues. De manière surprenante, la hiérarchie entre les protocoles n'est pas respectée et varie en fonction de la couche MAC utilisée. Une observation notable concerne le modèle le plus simple FLOOD, qui était utilisé comme borne inférieure dans les études théoriques. En effet, malgré sa simplicité, il obtient de très bonnes performances dans tous les domaines.

## A.5 Perspectives

Les perspectives de cette thèse sont multiples et concernent aussi bien la partie théorique, que la partie pratique. On a vu que l'agrégation de données est un problème difficile à résoudre, même en connaissant l'évolution du futur. Cependant, les réseaux utilisés en pratiques sont bien différents des réseaux correspondant au pire. Notamment, la périodicité, ainsi que l'apparition de schémas récurrents dans l'évolution du réseau au cours du temps peut permettre l'apparition d'algorithmes spécialisés ayant de bonnes performances en pratiques. Le même constat peut être fait dans le cas de l'agrégation de données en ligne. Par exemple, il serait intéressant d'étudier les performances de notre algorithme Waiting Greedy dans des graphes géométriques aléatoires. Aussi, nous pensons que les résultats seraient similaires, même en autorisant un nombre constant de transmissions.

D'un point de vue pratique, notre modèle énergétique a montré qu'il peut être utilisé pour évaluer les performances de protocoles dans des conditions réalistes. Son coût d'utilisation limité (il ne prend que très peu de temps de simulation) devrait encourager son utilisation systématique. Les perspectives à court terme consistent à effectuer une campagne plus précise en utilisant un plus grand nombre de capteurs.

Aussi, l'utilisation de notre modèle aux côtés des couches MAC implémentées dans les réseaux de capteurs dans le corps humain (notamment 802.15.6) nous semble être une prochaine étape logique.

Cette thèse a permis de d'approfondir nos connaissances dans les réseaux de capteurs sans fil sur les difficultés à réduire la consommation d'énergie lors d'opérations simples, comme l'agrégation de données, et sur la nécessité d'effectuer des évaluations réalistes des protocoles.

# Index

# Acronyms and Abbreviations

| | |
|---|---|
| **BER** | Bit-Error-Rate. . . . 20 |
| **DODA** | Distributed Online Data Aggregation. . . . 37 |
| **ECG** | Electrocardiogram. . . . 2 |
| **EEG** | Electroencephalogram. . . . 2 |
| **EMG** | Electromyogram. . . . 2 |
| **FCT** | Foremost Convergecast Tree. . . . 27 |
| **MAC** | Media Access Control. . . . 16 |
| **MDAT** | Minimum Data Aggregation Time. . . . 35 |
| **MEG** | Markovian Evolving Graph. . . . 29 |
| **PER** | Packet-Error-Rate. . . . 20 |
| **SINR** | Signal-to-Interference-plus-Noise Ratio. . . . 21 |
| **SNR** | Signal to Noise Ratio. . . . 20 |
| **TVG** | Time-Varying Graph. . . . 25 |
| **UDG** | Unit-Disk Graph. . . . 22 |
| **w.h.p.** | With high probability. . . . 40 |
| **WSN** | Wireless Sensor Network. . . . 9 |

# List of Symbols

# Bibliography


[802a] 802.15.4 standard. https://standards.ieee.org/getieee802/download/802.15.4-2011.pdf. 105, 137

[802b] 802.15.6 standard. https://standards.ieee.org/findstds/standard/802.15.6-2012.html. 16

[80203] Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks specific requirements part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (lr-wpans). *IEEE Std 802.15.4-2003*, pages 0_1–670, 2003. 16, 88

[AGS03] Valliappan Annamalai, Sandeep KS Gupta, and Loren Schwiebert. On tree-based convergecasting in wireless sensor networks. In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, volume 3, pages 1942–1947. IEEE, 2003. 34, 40, 124

[AKL08] Chen Avin, Michal Koucký, and Zvi Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *Automata, languages and programming*, pages 121–132. Springer, 2008. 29

[AMadH14] Sebastian Abshoff and Friedhelm Meyer auf der Heide. Structural information and communication complexity: 21st international colloquium, sirocco 2014, takayama, japan, july 23-25, 2014. proceedings. pages 194–209, Cham, 2014. Springer International Publishing. 41

[And95] John GT Anderson. Pilot survey of mid-coast maine seabird colonies: an evaluation of techniques. *Bangor, ME*, 1995. 12

[AT10a] Asim Ali and Sébastien Tixeuil. Advanced faults patterns for WSN dependability benchmarking. In *Proceedings of ACM MSWiM 2010*, Bodrum, Turkey, 2010. ACM. 83

[AT10b] Asim Ali and Sébastien Tixeuil. XS-WSNet : Extreme-scale wireless sensor simulation. In *Proceedings of WOWMOM 2010*, Montreal, Canada, 2010. IEEE Computer Society. 83

[AZAFA14] Mohammed Abo-Zahhad, Osama Amin, Mohammed Farrag, and Abdelhay Ali. A survey on protocols, platforms and simulation tools for wireless sensor networks. *Int. J. Energy, Inf. Commun*, 5(6):17–34, 2014. 13

[B+11] Athanassios Boulis et al. Castalia: A simulator for wireless sensor networks and body area networks. *NICTA: National ICT Australia*, 2011. 83

[BAB+07] Chris R Baker, Kenneth Armijo, Simon Belka, Merwan Benhabib, Vikas Bhargava, Nathan Burkhart, Artin Der Minassians, Gunes Dervisoglu, Lilia Gutnik, M Brent Haick, et al. Wireless sensor networks for home health care. In *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, volume 2, pages 832–837. IEEE, 2007. 11

[Bam98] N. Bambos. Toward power-sensitive network architectures in wireless communications: concepts, issues, and design aspects. *IEEE Personal Communications*, 5:50–59, 1998. 2

[BBL+14] Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. CRAWDAD dataset roma/taxi (v. 2014-07-17). Downloaded from http://crawdad.org/roma/taxi/20140717, July 2014. 12

[BCPPB15] Wafa Badreddine, Claude Chaudet, Federico Petruzzi, and Maria Potop-Butucaru. Broadcast strategies in wireless body area networks. In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 83–90. ACM, 2015. 40, 117

[BDBF+15] Quentin Bramas, Wilfried Dron, Mariem Ben Fadhl, Khalil Hachicha, Patrick Garda, and Sébastien Tixeuil. WiSeBat: Accurate Energy Benchmarking of Wireless Sensor Networks. In *Proceedings of the Forum on specification & Design Languages (FDL 2015)*, Barcelona, Spain, September 2015. IEEE Press. 4, 106, 116, 137

[BHCG08] Elyes Ben Hamida, Guillaume Chelius, and Jean-Marie Gorce. Scalable versus Accurate Physical Layer Modeling in Wireless Network Simulations. In IEEE Computer Society, editor, *22nd ACM/IEEE/SCS workshop on Principles of Advanced and Distributed Simulation (PADS 2008)*, pages 127–134, Roma, Italy, June 2008. ACM/IEEE/SCS. 84

[BK98] Therese Biedl and Goos Kant. A better heuristic for orthogonal graph drawings. *Computational Geometry*, 9(3):159–180, 1998. 24

[BL09] Farid Benbadis and Jeremie Leguay. CRAWDAD dataset upmc/rollernet (v. 2009-02-02). Downloaded from http://crawdad.org/upmc/rollernet/20090202, February 2009. 12

[BMT16] Quentin Bramas, Toshimitsu Masuzawa, and Sébastien Tixeuil. Distributed online data aggregation in dynamic graphs. *CoRR*, abs/1602.01065, 2016. 4, 116

[Bol98] Béla Bollobás. *Random graphs*. Springer, 1998. 28

[Bou09] Athanassios Boulis. Castalia user manual. 2009. xvi, 83

[BT15] Quentin Bramas and Sébastien Tixeuil. The complexity of data aggregation in static and dynamic wireless sensor networks. In Andrzej Pelc and Alexander A. Schwarzmann, editors, *Stabilization, Safety, and*

*Security of Distributed Systems*, volume 9212 of *Lecture Notes in Computer Science*, pages 36–50. Springer International Publishing, 2015. 4, 34, 116, 128

[BT16] Quentin Bramas and Sébastien Tixeuil. *Benchmarking Energy-Efficient Broadcast Protocols in Wireless Sensor Networks*. Springer International Publishing, 2016. 4, 116

[BV05] Jean-Yves Le Boudec and Milan Vojnovic. Perfect simulation and stationarity of a class of mobility models. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 4, pages 2743–2754. IEEE, 2005. 29

[BYAH06] Michael Buettner, Gary V Yee, Eric Anderson, and Richard Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320. ACM, 2006. 16, 88

[Car10] Paulo Cardieri. Modeling interference in wireless ad hoc networks. *Communications Surveys & Tutorials, IEEE*, 12(4):551–572, 2010. 21, 121

[CBB09] Julien Champ, Anne-Elisabeth Baert, and Vincent Boudet. Dynamic localized broadcast incremental power protocol and lifetime in wireless ad hoc and sensor networks. In *Wireless and Mobile Networking*, pages 286–296. Springer, 2009. 86

[CBP+05a] Gilbert Chen, Joel Branch, Michael Pflug, Lijuan Zhu, and Boleslaw Szymanski. Sense: a wireless sensor network simulator. In *Advances in pervasive computing and networking*, pages 249–267. Springer, 2005. xvi, 82

[CBP+05b] Gilbert Chen, Joel Branch, Michael Pflug, Lijuan Zhu, and Boleslaw Szymanski. Sense: A wireless sensor network simulator. In BoleslawK. Szymanski and Bülent Yener, editors, *Advances in Pervasive Computing and Networking*, pages 249–267. Springer US, 2005. 82, 85

[CBSP08] Avinash Chiganmi, Mehmet Baysan, Kamil Sarac, and Ravi Prakash. Variable power broadcast using local information in ad hoc networks. *Ad Hoc Networks*, 6(5):675–695, 2008. 87

[CCJ90] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1):165 – 177, 1990. 22

[CFF07] Guillaume Chelius, Antoine Fraboulet, and Eric Fleury. Worldsens: A fast and accurate development framework for sensor network applications. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, SAC '07, pages 222–226. ACM, 2007. 85

[CFQS11] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. In *Ad-hoc, Mo-*

*bile, and Wireless Networks*, pages 346–359. Springer, 2011. xv, 25, 27

[CGN12] Alejandro Cornejo, Seth Gilbert, and Calvin Newport. Aggregation in dynamic networks. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, pages 195–204. ACM, 2012. 41, 42, 128

[ČHE02] Mario Čagalj, Jean-Pierre Hubaux, and Christian Enz. Minimum-energy broadcast in all-wireless networks: Np-completeness and distribution issues. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 172–182. ACM, 2002. 80, 86

[CHZ05] Xujin Chen, Xiaodong Hu, and Jianming Zhu. Minimum data aggregation time problem in wireless sensor networks. In *Mobile Ad-hoc and Sensor Networks*, pages 133–142. Springer, 2005. 34, 41, 45, 46, 124, 125, 126, 127

[CISRS05] Julien Cartigny, François Ingelrest, David Simplot-Ryl, and Ivan Stojmenović. Localized lmst and rng based minimum-energy broadcast protocols in ad hoc networks. *Ad Hoc Networks*, 3(1):1–16, 2005. 86, 87

[CMM+10] Andrea EF Clementi, Claudio Macci, Angelo Monti, Francesco Pasquale, and Riccardo Silvestri. Flooding time of edge-markovian evolving graphs. *SIAM journal on discrete mathematics*, 24(4):1694–1712, 2010. 29, 40

[CMPS09] Andrea EF Clementi, Angelo Monti, Francesco Pasquale, and Riccardo Silvestri. Information spreading in stationary markovian evolving graphs. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–12. IEEE, 2009. 40

[CMS13] Andrea Clementi, Angelo Monti, and Riccardo Silvestri. Fast flooding over manhattan. *Distributed computing*, 26(1):25–38, 2013. 40

[CPS00] Andrea EF Clementi, Paolo Penna, and Riccardo Silvestri. The power range assignment problem in radio networks on the plane. In *STACS 2000*, pages 651–660. Springer, 2000. 85

[CST15] Andrea Clementi, Riccardo Silvestri, and Luca Trevisan. Information spreading in dynamic graphs. *Distributed Computing*, 28(1):55–73, 2015. 40

[DD09] I. Dietrich and F. Dressler. On the lifetime of wireless sensor networks. *ACM Transaction on Sensor Network*, pages 1–38, 2009. 85, 132

[DDV+14] W. Dron, S. Duquennoy, T. Voigt, K. Hachicha, and P. Garda. An emulation-based method for lifetime estimation of wireless sensor networks. In *Distributed Computing in Sensor Systems (DCOSS), 2014 IEEE International Conference on*, pages 241–248, 2014. 85, 88, 93, 132, 133

[DGV04] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462, Nov 2004. xv, 14, 16

[DHG13] Wilfried Dron, Khalil Hachicha, and Patrick Garda. A fixed frequency sampling method for wireless sensors power consumption estimation. In *New Circuits and Systems Conference (NEWCAS), 2013 IEEE 11th International*, pages 1–4. IEEE, 2013. 94

[DSVA06] Adam Dunkels, Oliver Schmidt, Thiemo Voigt, and Muneeb Ali. Protothreads: simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 29–42. Acm, 2006. 14

[Dun11] Adam Dunkels. The contikimac radio duty cycling protocol. 2011. 16, 88, 105, 137

[DV11] A.K. Dwivedi and O.P. Vyas. An exploratory study of experimental tools for wireless sensor networks. *Wireless Sensor Network*, 3(7):215–240, 2011. `http://www.scirp.org/journal/PaperInformation.aspx?PaperID=6402`. 81

[ER59] P Erdős and A Rényi. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959. 28

[FCF07] Antoine Fraboulet, Guillaume Chelius, and Eric Fleury. Worldsens: development and prototyping tools for application specific wireless sensors networks. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 176–185. IEEE, 2007. xvi, 4, 79, 83, 84, 105, 137

[FRWZ07] Elena Fasolo, Michele Rossi, Jorg Widmer, and Michele Zorzi. In-network aggregation techniques for wireless sensor networks: a survey. *Wireless Communications, IEEE*, 14(2):70–87, 2007. 41

[FW10] L.M. Feeney and D. Willkomm. Energy framework: An extensible framework for simulating battery consumption in wireless networks. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, pages 20:1–20:4, 2010. 85

[GW02] ANDREA J. GOLDSMITH and STEPHEN B. WICKER. Design challenges for energy-constrained ad hoc wireless networks. *IEEE Wireless Communications*, 9:8–27, 2002. 2

[GY07] Song Guo and Oliver WW Yang. Energy-aware multicasting in wireless ad hoc networks: A survey and discussion. *Computer Communications*, 30(9):2129–2148, 2007. 86

[HAR] Communication Foundation HART. Wirelesshart overview. 16

[HCG09] Elyes Ben Hamida, Guillaume Chelius, and Jean-Marie Gorce. Impact of the physical layer modeling on the accuracy and scalability of wireless network simulation. *Simulation*, 2009. 20

[HNL08] M. Healy, T. Newe, and E. Lewis. Wireless sensor node hardware: A review. In *Sensors, 2008 IEEE*, pages 621–624, Oct 2008. 13

[HRFR06] Thomas R Henderson, Sumit Roy, Sally Floyd, and George F Riley. ns-3 project goals. In *Proceeding from the 2006 workshop on ns-2: the IP network simulator*, page 13. ACM, 2006. 81, 85

[inc] Moteiv inc. Tmote sky data sheet. xv, 15

[iNWK+15] J. i. Naganawa, K. Wangchuk, M. Kim, T. Aoyagi, and J. i. Takada. Simulation-based scenario-specific channel modeling for wban cooperative transmission schemes. *IEEE Journal of Biomedical and Health Informatics*, 19(2):559–570, March 2015. 11

[IRK09] Aravind Iyer, Catherine Rosenberg, and Aditya Karnik. What is the right model for wireless channel interference? *Wireless Communications, IEEE Transactions on*, 8(5):2662–2671, 2009. 21, 121

[ISA] Automation Standards Compliance Institute ISA. Isa100.11a technology standard. 16

[ISR+08] Fran Ingelrest, David Simplot-Ryl, et al. Localized broadcast incremental power protocol for wireless ad hoc networks. *Wireless Networks*, 14(3):309–319, 2008. 86

[JM96] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile computing*, pages 153–181. Springer, 1996. 29

[KK05] Alex Kesselman and Dariusz Kowalski. Fast distributed algorithm for convergecast in ad hoc geometric radio networks. In *Wireless On-demand Network Systems and Services, 2005. WONS 2005. Second Annual Conference on*, pages 119–124. IEEE, 2005. 34, 41

[KKKP97] Lefteris M Kirousis, Evangelos Kranakis, Danny Krizanc, and Andrzej Pelc. Power consumption in packet radio networks. In *STACS 97*, pages 363–374. Springer, 1997. 85

[LHS05] Ning Li, Jennifer C Hou, and Lui Sha. Design and analysis of an mst-based topology control algorithm. *Wireless Communications, IEEE Transactions on*, 4(3):1195–1206, 2005. 86

[Lic82] David Lichtenstein. Planar formulae and their uses. *SIAM journal on computing*, 11(2):329–343, 1982. 46, 47

[LLWC03] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137. ACM, 2003. 82

[LMP+05] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. *Ambient Intelligence*, chapter TinyOS: An Operating System for Sensor Networks, pages 115–148. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. 14

[MBGW13] Seytkamal Medetov, Mohamed Bakhouya, Jaafar Gaber, and Maxime Wack. Evaluation of an energy-efficient broadcast protocol in mobile ad hoc networks. In *Telecommunications (ICT), 2013 20th International Conference on*, pages 1–5. IEEE, 2013. 117

[MCP+02] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97. ACM, 2002. xv, 12, 13

[MiX03] MiXiM. Simulator for wireless and mobile networks using omnet++. *[Online] http://mixim.sourceforge.net/*, 2003. 83

[ML08] David Moss and Philip Levis. Box-macs: Exploiting physical and link layer boundaries in low-power networking. *Computer Systems Laboratory Stanford University*, pages 116–119, 2008. 16

[Mod03] OPNET Modeler. Opnet technologies. *Inc.[Online] http://www.opnet.com*, 2003. 81

[MZ12] Bartosz Musznicki and Piotr Zwierzykowski. Survey of simulators for wireless sensor networks. *International Journal of Grid and Distributed Computing*, 5(3):23–50, 2012. 81

[ns297] The network simulator ns-2, 1997. 81, 85

[NZC11] Thanh Dang Nguyen, Vyacheslav Zalyubovskiy, and Hyunseung Choo. Efficient time latency of data aggregation based on neighboring dominators in wsns. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6. IEEE, 2011. 41

[ODE+06] Fredrik Osterlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with cooja. In *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, pages 641–648. IEEE, 2006. 82

[PCC+08] Enrico Perla, Art Ó Catháin, Ricardo Simon Carbajo, Meriel Huggard, and Ciarán Mc Goldrick. Powertossim z: realistic energy modelling for wireless sensor network environments. In *Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pages 35–42. ACM, 2008. 85

[Pea05] Karl Pearson. The problem of the random walk. *Nature. 72*, page 294, 1905. 29

[PHC04] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd*

*international conference on Embedded networked sensor systems*, pages 95–107. ACM, 2004. 16

[PMA06] Dario Pompili, Tommaso Melodia, and Ian F. Akyildiz. Deployment analysis in underwater acoustic wireless sensor networks. In *Proceedings of the 1st ACM International Workshop on Underwater Networks*, WUWNet '06, pages 48–55, New York, NY, USA, 2006. ACM. 10

[PMS+15] Luca Petrillo, Theodoros Mavridis, Julien Sarrazin, Aziz Benlarbi-Delai, and Philippe De Doncker. Statistical On-Body Measurement Results at 60 GHz. *IEEE Transactions on Antennas and Propagation*, Volume : 63(Issue : 1):400 – 403, January 2015. 11

[PSDG09] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. CRAWDAD dataset epfl/mobility (v. 2009-02-24). Downloaded from http://crawdad.org/epfl/mobility/20090224, February 2009. 12

[PSG13] Chris Phillips, Douglas Sicker, and Dirk Grunwald. A survey of wireless path loss prediction and coverage mapping methods. *Communications Surveys & Tutorials, IEEE*, 15(1):255–270, 2013. 20

[PSS00] Sung Park, Andreas Savvides, and Mani B. Srivastava. Sensorsim: A simulation framework for sensor networks. In *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWIM '00, pages 104–111, New York, NY, USA, 2000. ACM. 85

[RFG13] Christian Rohner, LauraMarie Feeney, and Per Gunningberg. Evaluating battery models in wireless sensor networks. In Vassilis Tsaoussidis, AndreasJ. Kassler, Yevgeni Koucheryavy, and Abdelhamid Mellouk, editors, *Wired/Wireless Internet Communication*, volume 7889 of *Lecture Notes in Computer Science*, pages 29–42. 2013. 85

[RG13] R. R. Rout and S. K. Ghosh. Enhancement of lifetime using duty cycle and network coding in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 12(2):656–667, February 2013. 117

[RGL10] Meirui Ren, Longjiang Guo, and Jinbao Li. A new scheduling algorithm for reducing data aggregation latency in wireless sensor networks. *International Journal of Communications, Network & System Sciences*, 3(8), 2010. 41

[RSH+09] Injong Rhee, Minsu Shin, Seongik Hong, Kyunghan Lee, Seongjoon Kim, and Song Chong. CRAWDAD dataset ncsu/mobilitymodels (v. 2009-07-23). Downloaded from http://crawdad.org/ncsu/mobilitymodels/20090723, July 2009. 12

[RVR03] Ravishankar Rao, Sarma Vrudhula, and Daler N Rakhmatov. Battery modeling for energy aware system design. *Computer*, 36(12):77–87, 2003. 85

[SBB13] Shantanu Sharma, Rakesh K Bansal, and Sunny Bansal. Issues and challenges in wireless sensor networks. In *Machine Intelligence and Research Advancement (ICMIRA), 2013 International Conference on*, pages 58–62. IEEE, 2013. 17

[SLD+11] Harsh Sundani, Haoyue Li, Vijay K Devabhaktuni, Mansoor Alam, and Prabir Bhattacharya. Wireless sensor network simulators a survey and comparisons. *International Journal of Computer Networks*, 2(5):249–265, 2011. 81

[SML+04] Gyula Simon, Miklós Maróti, Ákos Lédeczi, György Balogh, Branislav Kusy, András Nádas, Gábor Pap, János Sallai, and Ken Frampton. Sensor network-based countersniper system. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 1–12. ACM, 2004. 12

[SVT08] Chandresh Pratap Singh, OP Vyas, and Manoj Ku Tiwari. A survey of simulation in sensor networks. In *Computational Intelligence for Modelling Control & Automation, 2008 International Conference on*, pages 867–872. IEEE, 2008. 81

[TLB+09] P. U. Tournoux, J. Leguay, F. Benbadis, V. Conan, M. Dias de Amorim, and J. Whitbeck. The Accordion phenomenon: Analysis, characterization, and impact on DTN routing. In *IEEE INFOCOM 2009 - The 28th Conference on Computer Communications*, pages 1116–1124. IEEE, April 2009. 12

[Tou80] Godfried T Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern recognition*, 12(4):261–268, 1980. 86

[TPS+05] Gilman Tolle, Joseph Polastre, Robert Szewczyk, David Culler, Neil Turner, Kevin Tu, Stephen Burgess, Todd Dawson, Phil Buonadonna, David Gay, et al. A macroscope in the redwoods. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 51–63. ACM, 2005. 11

[V+01] András Varga et al. The omnet++ discrete event simulation system. In *Proceedings of the European simulation multiconference (ESM'2001)*, volume 9, page 65. sn, 2001. 81

[WALR+06] Geoffrey Werner-Allen, Konrad Lorincz, Mario Ruiz, Omar Marcillo, Jeff Johnson, Jonathan Lees, and Matt Welsh. Deploying a wireless sensor network on an active volcano. *Internet Computing, IEEE*, 10(2):18–25, 2006. xv, 11

[WC02] Brad Williams and Tracy Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 194–205. ACM, 2002. 117

[WG03] Zhengdao Wang and Georgios B Giannakis. A simple and general parameterization quantifying performance in fading channels. *Communications, IEEE Transactions on*, 51(8):1389–1398, 2003. 20

[WGMM08] D. Weber, J. Glaser, S. A. Madani, and S. Mahlknecht. Power aware simulation framework for wireless sensor networks and nodes. *EURASIP Journal on Embedded Systems*, 2008. 85

[wis14] Wisebat model, 2014. 88

[WNE00] J.E. Wieselthier, Gam D. Nguyen, and Anthony Ephremides. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 585–594 vol.2, 2000. 86

[XLM+11] XiaoHua Xu, Mo Li, XuFei Mao, Shaojie Tang, and ShiGuang Wang. A delay-efficient algorithm for data aggregation in multihop wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 22(1):163–175, 2011. 41

[YLL09] Bo Yu, Jianzhong Li, and Yingshu Li. Distributed data aggregation scheduling in wireless sensor networks. In *INFOCOM 2009, IEEE*, pages 2159–2167. IEEE, 2009. 41

[YMG08] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292 – 2330, 2008. 10, 11, 17

[YTKY12] Yukiko Yamauchi, Sébastien Tixeuil, Shuji Kijima, and Masafumi Yamashita. Brief announcement: Probabilistic stabilization under probabilistic schedulers. In Marcos K. Aguilera, editor, *Distributed Computing - 26th International Symposium, DISC 2012, Salvador, Brazil, October 16-18, 2012. Proceedings*, volume 7611 of *Lecture Notes in Computer Science*, pages 413–414. Springer, 2012. 118

[Z+06] Alliance ZigBee et al. Zigbee specification, 2006. 16

[ZBG98] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. Glomosim: a library for parallel simulation of large-scale wireless networks. In *Parallel and Distributed Simulation, 1998. PADS 98. Proceedings. Twelfth Workshop on*, pages 154–161. IEEE, 1998. 82

[ZSLM04] Pei Zhang, Christopher M Sadler, Stephen A Lyon, and Margaret Martonosi. Hardware design experiences in zebranet. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 227–238. ACM, 2004. 12

## Réseaux de Capteurs sans Fil Efficaces en Énergie

**Résumé :** Les réseaux de capteurs sans fil sont constitués de noeuds capteurs, capables de récolter des données, de les analyser et de les transmettre. Ces réseaux ont plusieurs applications, en fonction de la zone où ils sont déployés. Application militaire ou de sauvetage dans des zones pouvant être inaccessibles aux humains ; application sanitaire avec des capteurs déployés sur et dans le corps humain ; application de surveillance avec des capteurs sur les voitures d'un ville, ou les arbres d'une forêt. Les noeuds sont autonomes en énergie et il est primordial d'assurer leur longévité sans retarder la récolte des données. La tache principale réalisée par les réseaux de capteurs sans fils consiste à effectuer des mesures et à envoyer ces données jusqu'à un noeud coordinateur. Cette tache d'agrégation est effectuée régulièrement, ce qui en fait la plus consommatrice d'énergie. L'étude approfondie de la consommation d'énergie des capteurs, qui au centre de ma thèse, peut se traduire de différentes manières.

Premièrement, nous avons étudié la complexité du problème de l'agrégation de données en utilisant un modèle simplifié pour représenter un réseau de capteurs sans fils. Dans ce modèle, nous avons montré que la recherche d'une solution optimale permettant d'agréger les données d'un réseau de capteurs sans fil quelconque n'était pas réalisable en pratique, même pour un algorithme centralisé connaissant l'évolution futur du réseaux. De plus nous avons étudié la résolution de ce problème par un algorithme distribué fonctionnant en temps réel. Nous avons montré que le problème n'avait pas de solution en général, sans connaissance supplémentaires.

Secondement, nous nous sommes concentrés sur l'estimation de cette durée de vie. Les simulateurs existant implémentent souvent des modèles trop simplistes de consommation et de batterie. Par ailleurs, les modèles plus réalistes de batterie, implémentés dans des simulateurs généralistes, sont trop complexes pour être utilisés pour les réseaux de capteurs possédant de nombreux noeuds, et dont la durée à simuler peut atteindre des mois, voire des années. WiSeBat est un modèle de batterie et de consommation d'énergie optimisé pour les réseaux de capteurs, implémenté dans le simulateur WSNET. Après validation, nous l'avons utilisé pour comparer les performances des algorithmes de broadcast efficaces en énergie.

**Mots clés :** réseau de capteur, aggrégation de données, évaluation de la durée de vie, réseau dynamique, algorithme de diffusion

## Energy-Centric Wireless Sensor Networks

**Abstract:** A wireless sensor network is an ad-hoc network connecting small devices equipped with sensors. Such networks are self-organized and independent of any infrastructure. The deployment of a WSN is possible in areas inaccessible to humans, or for applications with a long lifetime requirement. Indeed, devices in a wireless sensor network are usually battery-powered, tolerate failure, and may use their own communication protocols, allowing them to optimize the energy consumption. The main application of WSNs it to sense the environment at different locations and aggregate all the data to a specific node that logs it and can send alerts if necessary. This task of data aggregation is performed regularly, making it the most energy consuming. As reducing the energy consumed by sensor is the leading challenge to ensure sustainable applications, we tackle in this thesis the problem of aggregating efficiently the data of the network. Then, we study lifetime evaluation techniques and apply it to benchmark existing energy-centric protocols.

**Keywords:** wireless sensor network, data agregation, lifetime evaluation, dynamic network, broadcast protocol