Wireless Self-adaptive Ad hoc and Sensor Networks: Energy Efficiency and Spatial Reuse

Ichrak Amdouni

THÈSE DE DOCTORAT DE
L'UNIVERSITÉ PIERRE ET MARIE CURIE

presented by

Ichrak Amdouni

to obain the degree of

DOCTEUR DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

Discipline: Computer Science

Host Laboratory: Inria Rocquencourt

Date: 14 February 2013

Wireless Self-adaptive Ad hoc and Sensor Networks: Energy Efficiency and Spatial Reuse

Thesis Advisor: **Mrs. Pascale Minet**
Thesis Co-Advisor: **Mr. Cédric Adjih**

Jury:

Mr.	André-Luc Beylot	Reviewer	Professor, ENSEEIHT, France
Mrs.	Isabelle Guérin Lassous	Reviewer	Professor, University of Lyon I, France
Mr.	Tuan Dang	Examiner	Doctor, Engineer Research Expert, EDF, France
Mr.	Michel Misson	Examiner	Professor, University of Clermont I, France
Mr.	Guy Pujolle	Examiner	Professor, Pierre & Marie Curie University, France
Mrs.	Leila Saidane	Examiner	Professor, ENSI, Manouba University, Tunisia

Acknowledgement

At the beginning of this acknowledgement, I take this opportunity to express my gratitude to Mr. André-Luc Beylot and Mrs. Isabelle Guérin Lassous for having the willingness to review my thesis. Also, my thanks go to the members of the jury for their presence and for the interest they expressed in my research. So, I thank Mr. Tuan Dang, Mr. Michel Misson, Mr. Guy Pujolle and Mrs. Leila Saidane.

This thesis is the completion of my PhD study at Inria, within the Hipercom project team, where I benefited a lot from the inspiring and friendly atmospheres. I am very grateful to many people who contributed directly or indirectly to my thesis. It is with big emotion that I address these words to them while reliving many warm moments of my PhD.

I could not have imagined a better advisor than you Pascale! I sincerely thank you for your continuous support, orientation and immense knowledge. Your perpetual energy and enthusiasm in research had motivated me so much. I express my sincere gratitude for you for having been always available to guide and advise me. Your encouragement and personal guidance have provided a good basis for the present thesis. My goal is not to address letters through this page, but I cannot avoid expressing my admiration regarding your high responsibility and commitment towards me and towards all the tasks that you do. This gives me power and confidence even in the most tiring moments in my research thesis. Thank you for caring about me, my thesis, my publications and my future career. Thank you for "holding" me during conferences, presentations and pleasant mission trips we had together :).

I have been very privileged to get to know and to collaborate with Cédric. I am lucky; my thesis let me know a good researcher and a sympathetic person. Thank you Cédric for your valuable help and advice. Let me salute your generosity and your willingness to help people and share your immense knowledge with them. Thank you for having oriented and supported me with patience and encouragements in times of new ideas and difficulties, for your great sense of responsibility and professionalism. I admire your capacity of criticizing, giving sense and value to ideas or projects of ideas: You say things differently, and this makes difference! I owe my most sincere gratitude to you Cédric for any knowledge that I would have learned from you; I will never forget the "noeud intermédiaire", my first "lesson" :).

I am also happy for having collaborated with all of you Erwan Livolant, Ridha Soua and Saoucène Mahfoudh. Also, to all people in the OCARI project and from whom I would have learned something, I say thank you.

I thank you Christine Anocq for your valuable help. You facilitate our administrative tasks inside Inria and outside also, and this is great! I am also honored to have known many other people in Hipercom. To all of them, I wish to say thank you for the nice atmosphere inside the team: Amina Naimi, Anis Laouiti, Erwan Livolant, Nadjib Achir, Paul Muhlethaler. I also thank Dominique Fortin for the interesting discussions with him. I say thank you to Richard James, our english teacher for the interesting lessons and for having reviewed my first paper.

Special thanks to my dear friends and colleagues Ala Weslati, Hana Baccouch, Ines Khoufi, Ridha Soua, Salman Malik and Saoucène Mahdoudh. I am grateful to all of you, for your prayers, encouragements, support and for the fun moments I have shared with you. I want to address my kind regards to many friends that I have known in Inria, for their nice company and sincere friendship, thank you Ahmed Rebai, Amel Hamzaoui, Faten Nabli, Ibtihel Ben Gharbiaa, Najmeddine Attia, Olfa Mzoughi, Skander Banaouas and Younes Bouchaala.
In particular, I am in debt to you Ines Ben Jemaa for your continuous support. I made you share my thesis with all its good and bad details, and your sharing was of a great help for me! I am grateful for this and for your generous affection even if I don't speak so much and even if I have not say it as it should be :). So, under big emotions, I say thank you and very good luck!

Also, I in debt to my uncle and his family for having tried to fill the gap of my parents. So thank you uncle Houcine, aunt Souad, Noura and Hamza.

This thesis would not have been possible without the support of my parents, my brothers, and my lovely sister. Thank you for your infinite support, love and trust.

To the soul of my brother Mohamed Ali
To my parents
To my brothers Atef, Mourad and Sami
To my sister Sarra

Abstract

The need to maximize network lifetime in wireless ad hoc networks and especially in wireless sensor networks requires the use of energy efficient algorithms and protocols. Motivated by the fact that a node consumes the least energy when its radio is in sleep state, we achieve energy efficiency by scheduling nodes activity. Nodes are assigned time slots during which they can transmit and they can turn off their radio when they are neither transmitting nor receiving. Compared to classical TDMA-based medium access scheme, spatial bandwidth use is optimized: non interfering nodes are able to share the same time slots, collisions are avoided and overhearing and interferences are reduced.

In our work about time slots assignment, two cases are studied. First, when nodes require equal channel access, we use node coloring: Nodes having the same color do not interfere and a color is mapped to a time slot. In this context, we proposed the coloring algorithm OSERENA. Second, when nodes have heterogeneous traffic demands, we designed the traffic aware time slot assignment algorithm TRASA. In this algorithm, any node is assigned a number of slots proportional to its medium access time needs. We show that reducing the number of colors for OSERENA and the number of slots for TRASA increases the amount of energy saved and decreases data transmission delays.

Unlike the majority of previous works, we generalize the definition of node coloring and slot allocation problems. Indeed, we set the maximum distance between two interfering nodes as a parameter of these problems. We prove that they are NP-complete, making heuristic approaches inevitable in practice.

In addition to the energy resource scarcity, wireless ad hoc and sensor networks have many other constraints. A sensor is a tiny device with small amount of memory and storage space. Also, these networks are prone to the unreliability of wireless communications. That is why, a central directive of this thesis is to design self-adaptive solutions. This adaptivity concerns many aspects such as the mission given by the application, the heterogeneity of node traffic demands, the network density, the regularity of network topology, and the failure of wireless links. More precisely, we made the following contributions: (1) Using a cross layer with the application allows us to define two coloring modes: one adapted to general applications, and the second to data gathering applications. We prove that this adaptation reduces the coloring algorithm overhead. (2) TRASA is defined for data gathering applications while adapting to the heterogeneous numbers of packets to transmit. It allocates to each node

a medium access time proportional to its traffic demand. (3) OSERENA scales for dense wireless networks; it uses a message whose size depends neither on the density nor on the number of nodes. Furthermore, compared to an existing coloring algorithm called SERENA, OSERENA reduces the amount of data stored by sensors. (4) We propose VCM, a coloring solution for grid wireless networks. This solution profits from the regularity of this topology and performs a periodic coloring by tiling a color pattern. (5) We enhance the robustness of the known algorithm SERENA to support unreliable links in data gathering applications: after topology changes, nodes can keep their initial colors without creating interferences.

All these algorithms and protocols have been implemented and simulated on configurations of wireless networks. Furthermore, we participated in the OCARI project that targets wireless sensor networks in industrial environments. We collaborated to implement and integrate OSERENA in a real testbed of sensors coupled with a new version of the known energy efficient routing protocol EOLSR adapted to data gathering applications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to Wireless Ad hoc and Sensor Networks

1.1.1 Overview

A wireless ad hoc network is a collection of wireless nodes communicating with each other in the absence of any infrastructure. These networks ensure collaborative computing and communications between devices of varied types and sizes (including personal computers, handhelds, telephones, appliances, industrial equipments, etc), in various areas (like buildings, conference rooms, industrial plants, etc). Wireless sensor networks (WSNs) have gained worldwide attention in recent years, particularly with the proliferation of Micro-Electro-Mechanical Systems (MEMS)[1] technology which has facilitated the development of smart sensors.

WSNs consist of a possibly large number of usually small and inexpensive nodes with sensing computation and wireless communications capabilities. The role of sensor nodes is to gather data from their surroundings and transmit them to a data sink node either directly or via multihop routing. Data gathering is a typical application in these networks. Furthermore, several applications are envisioned for WSNs like (1) military applications: tracking enemy forces, information gathering in battlefields, attack detection, (2) environment monitoring: habitat monitoring, animal tracking, forest fire detection, precision farming, (3) biomedical applications: vital signs monitoring like blood glucose and blood pressure, (4) industrial applications: process control, detection of liquid/gas leakage, factory automation, preventive maintenance, workers safety monitoring.

1.1.2 Main Issues and Challenges in WSNs

WSNs are considered as a specific kind of wireless ad hoc networks. They share many characteristics like wireless communications, interferences, autonomous deployment, etc.

However, WSNs differ from wireless ad hoc networks because sensors have limited capabilities and tight resources. This makes protocols proposed for traditional wireless networks

[1] MEMS is the technology of very small devices.

not well suited to WSNs. Thus, research focusing in these networks faces many challenges:

- **Energy efficiency:** A major constraint in WSNs is that sensor nodes are battery operated. Sensors utilize their batteries for communication and sensing. In case of battery exhaustion, the sensors functionality completely halts, inevitably leading to the interruption of the application. A second constraint is that sensors are usually deployed unattended and in large numbers like in military applications or underwater monitoring applications. As a consequence, it will be difficult to change or recharge batteries of these sensors. Therefore, the key solution is to optimize the power consumption and to maximize the network lifetime.

- **Scalability and reduced overhead:** Because of the low cost of sensors and the application requirements, the number of nodes in a WSN can be very large, and in general, sensors are densely deployed in the area of interest. Dense deployment enhances the reliability and the accuracy of data, but usually implies high data and control traffic exchanges. This is problematic because of two issues. First, sensor nodes have tight memory and storage due to their miniaturization. For instance, the TMote Sky sensors based on MSP340F1611 micro-controller have 10KB of RAM and 48KB of flash [2]. Second, the network bandwidth is very limited (250Kbps in IEEE 802.15.4 for example). Thus, algorithm scalability is an important design criterion for sensor network applications: algorithms must optimize bandwidth use and the amount of control traffic stored and exchanged.

- **Adaptivity to dynamic environment**: Environments in which ad hoc or sensor nodes operate are very dynamic. This dynamicity involves: the node position (in the presence of mobile sensors), frequent topology changes (because of hardware failure, depleted batteries, intermittent radio interferences, environmental factors, or sensor mobility), the data traffic (that can be uniform or heterogeneous, prioritized or not, sent to one sink or to any destination). In such conditions, the network should self-adapt to these time-varying conditions in order to guarantee the data delivery without increasing the network overhead.

1.2 Problem Statement

Among the aforementioned challenges, we mainly address the energy and bandwidth efficiency taking into account the following resource scarcity: energy, bandwidth, memory, in addition to the reduced processing capabilities and unreliable wireless links. We focus more particularly on **the node activity scheduling problem**, where nodes alternate active and sleeping periods to save energy. Indeed, protocols and algorithms that operate on these networks must

[2] http://en.wikipedia.org/wiki/List_of_wireless_sensor_nodes.

use these resources very efficiently to be able to meet the user requirements, which differ according to the considered application (e.g data collection in a sensor network).

In addition, these protocols must be **self-adaptive** as much as possible with regard to the constraints imposed by their operating environment (type of the network topology, density, unreliable radio links) and their application (types of communications used by the application).

These objectives target both wireless ad hoc and sensor networks. However, because WSNs are more constrained, they will be the focus of our work.

1.3 Methodology and Contributions of the Thesis

Figure 1.1 illustrates the problem, the strategy and the contributions of this thesis.

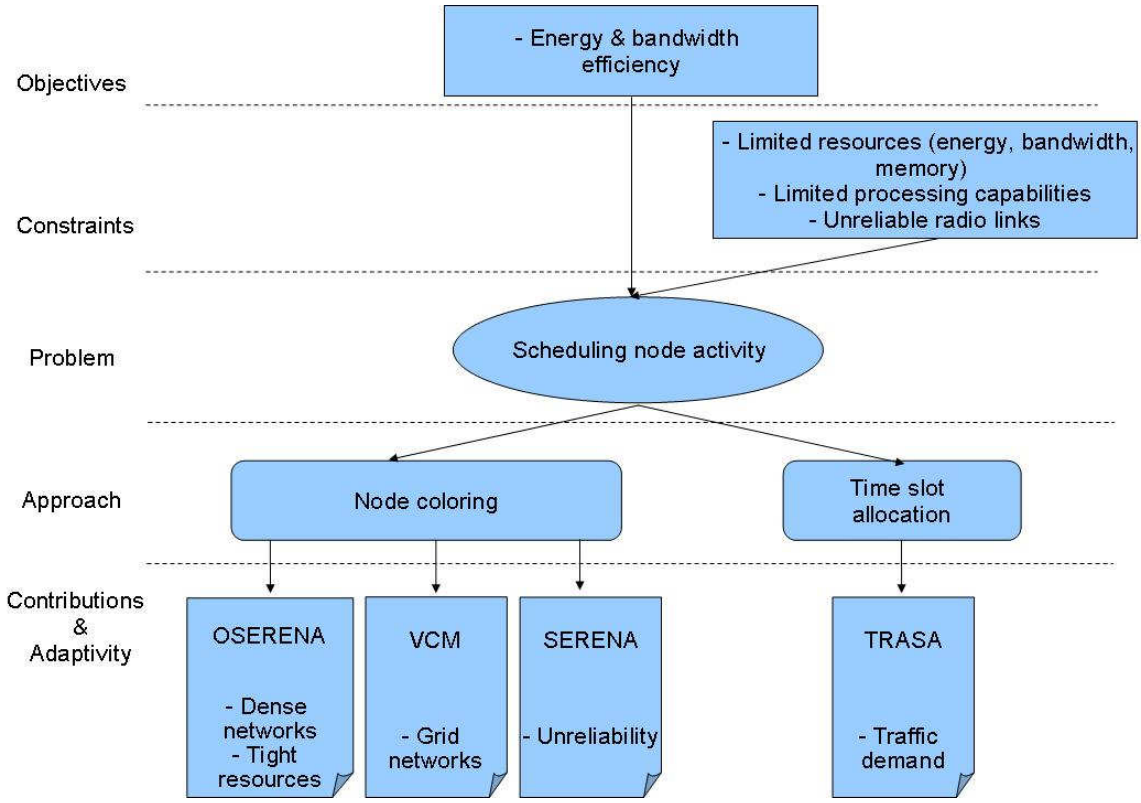


Figure 1.1: Overview about thesis problem and contributions.

To achieve energy efficiency, our approach is mainly based on node activity scheduling. This approach uses the "duty cycle". Thus, significant energy saving can be accomplished by putting nodes in the sleep state as long as possible while achieving the application mission. In this context, the solutions we propose can be implemented in the MAC layer on in an upper layer. However, they imply a slotted medium access scheme that is contention-free. Indeed, any node is allocated one or more time slots to transmit data. It is awake during the slots

allocated to its neighbors to receive data, and can sleep the remaining time to save energy.

To allocate time slots to nodes, we distinguish two cases:

1. First case: nodes require equal medium access times. Our solution is based on node coloring: We color the network and assign time slots to nodes based on these colors. The length of the activity period of the contention-free cycle is equal to one slot duration multiplied by the number of colors. That is why, our goal is to reduce this number.

   We developed two node coloring algorithms:

   (a) **OSERENA (Optimized SchEdule RoutEr Node Activity)**: OSERENA presents one adaptivity aspect of our work as it is designed for dense wireless networks and fits the small memory size of sensors. The size of the message used depends neither on the size of the network nor on the density of nodes. Also, bandwidth use is optimized: no bandwidth is lost in collisions and bandwidth sharing between non interfering nodes is guaranteed.

   (b) **VCM (Vector based Coloring Method)**: This algorithm adapts to the regularity of the grid topology by providing a periodic coloring based on the repetition of a color pattern. VCM is optimal. Furthermore, we have proved that the adaptivity to the grid regularity makes VCM more efficient than OSERENA in terms of number of colors.

   In addition, we updated the known node coloring algorithm **SERENA(SchEdule RoutEr Node Activity)**: We enhance the robustness of this algorithm to tolerate the parent change in data gathering trees. The objective is to avoid the appearance of color conflicts after any node changes its parent. So, the idea is to perform a coloring that anticipates the possible topology changes.

2. Second case: nodes require different medium access times. We proposed **TRASA (TRaffic Aware time Slot Assignment)** to assign time slots to nodes. TRASA is defined for data gathering applications. This algorithm is traffic aware: any node has a number of slots proportional to its traffic demand.

In our work, we proved the NP-completeness of both coloring and time slot allocation problems assuming that interferences are limited to h-hops where $h > 1$.

A big part of our work was done in the OCARI project that targets WSNs for industrial environments. OSERENA was integrated in wireless sensors with a modified version of the energy efficient routing protocol EOLSR. In this work, EOLSR is adapted to data gathering applications: It builds energy efficient trees rooted at the data sink. We have also specified communications between OSERENA and EOLSR: EOLSR collects topology information that is given to OSERENA and triggers the node coloring by OSERENA. Both protocols have been implemented and tested on a testbed of wireless sensors.

1.4 Manuscript Organization

This dissertation is structured as follows:

- **Chapter 1 introduces the context and the motivations of our work and describes our main contributions**.
- **Chapter 2 reviews the energy efficient techniques** used in wireless ad hoc and sensor networks and provides a classification of them.
- **Chapter 3 is centered on graph coloring for node activity scheduling.** We start by reviewing vertex and link coloring algorithms. Then, we describe the problem statement and the design choices to solve this problem. We prove that the coloring problem is NP-complete. In the second part of this chapter, we present an enhanced version of the known algorithm SERENA that (1) adapts to different types of communications in data gathering applications, and (2) is more robust against unreliable wireless communications.
- **Chapter 4 describes OSERENA**. We detail the rules and describe the properties of this algorithm. Then, we present a performance evaluation of this algorithm by simulation. The second part of this chapter deals with our contribution in the OCARI project. We describe the context and goals of this project, its architecture and its main components, in particular the routing protocol EOLSR. Some snapshots illustrating OSERENA and EOLSR running and integrated with the other OCARI protocols are given.
- **Chapter 5 focuses on grid coloring**. The aim of this chapter is to provide an optimal periodic coloring, i.e a coloring based on a tiling of a color pattern. We first give theoretical bounds on the number of colors as well as the optimal number of colors needed to color an infinite grid. Then, we describe our proposed method VCM and evaluate its performance by simulation on a wide range of grid networks.
- **Chapter 6 describes our slot assignment algorithm TRASA**. We start by formulating the slot assignment problem in wireless ad hoc and sensor networks, and proving its NP-completeness. Then, we present theoretical bounds on the optimal number of slots for line, multiline and tree networks. We finally describe TRASA and evaluate its performance regarding the total number of slots granted to nodes.
- **Chapter 7 concludes this dissertation and introduces directions for further researches**.

Chapter 2

Energy Efficiency in Wireless Ad hoc and Sensor Networks

Introduction

The energy conservation is a very challenging feature for both ad hoc and sensor networks whenever nodes are battery operated. Indeed, in such a case, these batteries are impossible or impractical to change or recharge. However, this issue is more present in WSNs because they are supposed to work unattended for longer time than ad hoc networks and can be deployed in harsh environment. Consequently, it is of major importance to develop dedicated solutions that handle application requirements in an energy-saving manner, prolonging thus the lifetime of the network. In this chapter, we address the energy conservation issue in wireless ad hoc and sensor networks, focusing on solutions adapted to WSNs. We provide the following classification of energy efficient techniques: (1) Routing, (2) Data reduction and (3) Node activity scheduling.

2.1 Energy Consumption in Wireless Sensor Networks

This section aims at studying the energy consumption in wireless sensor networks from multiple angles: in which operations the energy is consumed? Which is the most energy consuming radio state? What are the sources of energy waste?

2.1.1 Architecture of a Sensor Node

A sensor network is a wireless network of usually tiny, inexpensive, spatially distributed and radio-equipped sensors. These sensors are used for gathering information needed by smart environments and are particularly useful in unattended situations where terrain, climate and other environmental constraints make the deployment of wired/conventional networks a difficult task. Each sensor node is mainly composed of four components:

1. **Processing unit** (including the micro-controller) is in general associated with a small storage unit. It is responsible for managing and coordinating various activities of the sensor node: data storage, generating messages, data processing, and collecting sensed data, etc.

2. **Communication unit** (including the transceiver) is a radio device that can receive and transmit information.

3. **Sensing unit** measures physical data of the parameter to be monitored like temperature or pressure.

4. **The power source** supplies power to the node.

Each of the first three components consumes a specific amount of energy. This consumption will be characterized in the following section.

2.1.2 Characteristics of Energy Consumption

Due to various limitations arising from their limited size, weight, and ad hoc method of deployment, each sensor has a limited power. Hence, the power is a valuable resource that should be saved to maintain the network functioning. As far as the energy consumption is concerned, we observe that:

- A non negligible amount of energy is wasted in activities and radio states that are not useful from the application point of view. Section 2.1.2.1 details these sources.

- The communication unit consumes a much higher energy than the processing unit. Indeed, it has been shown that transmitting one bit may consume much higher than executing a few thousands of instructions [1]. Moreover, Halgamuge et al. [3] developed a model for energy consumption and performed simulations using hardware parameters like those of Mica2 Motes. They found that the communication consumes 51% of the total energy consumed by a node, while the processing consumed only 12%. For this reason, we study the energy consumed in the different radio states in Section 2.1.2.2.

2.1.2.1 Sources of Energy Waste

The major sources of energy waste are:

1. **Collision:** When a node receives more than one packet at the same time, these packets collide and should be re-transmitted. The packet retransmission causes loss of energy.

2. **Idle listening:** It happens when a node listens to an idle channel in anticipation of possible arrival of packets. When no packet is received, the node consumes energy because this idle listening.

3. **Overhearing:** It occurs when a node listens or overhears a packet thinking it may be the intended receiver, although the packet is destined to another node.

4. **Interference:** It happens when a node receives a packet and can not decode it.

5. **Control packet overhead:** The presence of too many control packets in the network such as beacons is another source of energy waste.

2.1.2.2 Energy Consumed by the Radio

The energy consumed by the communication unit varies with the radio state itself. Indeed, the radio can be in the transmit, receive, idle, or sleep state.

Table 2.1: The power consumption of two radio plateforms in wireless sensor networks[2]

Platforms	**Transmission** (mW)	**Reception/Idle** (mW)	**Sleep** (μW)
chipcon CC1000	19.8-80.1	22.2	0.6
chipcon CC2420	25.5-52.2	59.1	60

Table 2.1 presents reference values for energy consumption for two representative radio interfaces widely used in existing wireless sensor platforms: Chipcon CC1000, and Chipcon CC2420. Two observations can be drawn from this table:

- As the contrary one might think, the idle state is a non negligible energy consumption source. It consumes as much as the receive state although it is not useful from the application perspective. So, avoiding the idle state is a good solution to let sensors save energy.

- The sleep state is the least energy consuming state. Thus, a sensor should sleep as much as possible when it is not engaged in communication.

2.2 Classification of Energy Efficient Techniques

Minimizing the energy consumption has been a fertile research domain in the wireless ad hoc and sensor networks. Many techniques has been proposed to answer this question. To build our classification of these techniques, we are motivated by the following facts. First, sensor nodes have to send sensed data and route them continuously. Reducing the overhead induced by routing and by data and control message exchange is one of the techniques used to save energy. Second, in accordance with comments on Table 2.1, we notice that the sleep/awake scheduling reduces the energy wasted in the idle state by turning off radio when not in use. Third, adjusting the transmission power of nodes allows them to reduce energy consumption.

Hence, a possible classification of energy efficient techniques is based on the four following techniques:

1. **Energy efficient routing**: The focus of routing algorithms is to reduce the energy consumed in wireless communications and to avoid routing data through nodes with low residual energy. Examples of this class are [4, 10, 19].

2. **Protocol overhead reduction:** The focus of this category is to avoid data redundancy and reduce the control message overhead. Examples of this class are given by [32, 44, 47].

3. **Node activity scheduling:** Called also the **duty cycling**, is the periodic wake-up scheme that allows nodes to turn-off their radio and sleep when possible in order to save energy. Examples of this class are illustrated by [52, 53, 57].

4. **Adaptive transmission power:** This class is referred also as topology control. Indeed, nodes save energy by reducing their transmission power. Indeed, this allows nodes to have less communication links and hence to reduce the MAC contention and the overhead of computing the optimal routes to the sink. It allows also load balancing between nodes. However, the challenge is to optimally adjust the transmission power to ensure good tradeoff between energy consumption and connectivity. References for this class of energy efficiency are given by [68, 69, 70, 71].

In our work, we adopt this classification. However, we just detail the three first ones since we have integrated them in the design and implementation of our energy efficient solutions.

2.2.1 Energy Efficient Routing

Routing in WSNs is very challenging due to their specific characteristics that distinguish them from wireless ad hoc networks. Routing protocols differ in the way they take into account limited capacities of sensor nodes as well as the application and architecture requirements. For instance, routing protocols should face the challenge of reducing the energy consumed in the transmissions and to prolong the network lifetime. In this section, we provide a classification of routing protocols as follows:

1. **Data centric routing** considers data attributes to avoid data redundancy and reduce the number of transmissions.

2. **Hierarchical routing** distributes nodes into clusters which enhances scalability, favors data aggregation and avoids nodes to communicate directly with a far sink in case of data gathering applications for instance.

3. **Geographic routing** exploits the geographic information to avoid redundant transmissions and turn off unnecessary nodes.

4. **Mobility-based routing** relies on mobile nodes that walk around nodes and collect data from them. These nodes save energy because they only transmit to short range.

In the following, we detail these techniques.

2.2.1.1 Data centric routing

Data-centric routing protocols save energy by reducing the data redundancy. Indeed, data can be aggregated and nodes are queried according to their data attributes or interests. Indeed, aggregation is performed by intermediate nodes and in general does not assume a specific routing structure. Supporting the aggregation in these routing protocols implies the determination of the optimal aggregation period, that is the waiting time before forwarding the aggregated packets. A small value increases the data transmissions and a high period increases delays.

Two examples of algorithms of this category are SPIN [4] and Directed Diffusion [5] which are designed to WSNs. In SPIN, any node u advertises its data to its 1-hop neighbors. These neighbors request for these data if they do not possess them. Receiving this request, the node u sends the data to the interested neighbors. This scheme of data negotiation prevents the transmission of redundant packets unlike the classic flooding where each node repeats the packet it received to all its neighbors except the sender one. Further, in SPIN-2, nodes with a low energy threshold send neither requests nor data advertisements. However, this scheme does not ensure end-to-end data delivery. In Directed Diffusion [5], the sink requests data by broadcasting interests for named data. Nodes that have recorded some events matching the requested interest reply by sending gradient messages. Thus, routes toward the sink are established. Notice that since the routing is based on the data attributes, only good paths are selected. Moreover, there is no need for maintaining global network topology and intermediate nodes perform data aggregation. Consequently, energy is saved. However, this algorithm is only suitable for data-driven applications. Another example of data-centric routing protocols is the Rumor routing [6] that avoids the overhead induced by query flooding in Directed Diffusion. In Rumor routing, as soon as an event occurs in a network region, sensing nodes create some agents as event agents and propagate them along the network to build a list of known events. When a query is generated, it can be sent on a random walk until it finds a node that knows the path to the requested event instead of flooding the whole network. Other data-centric routing protocols are COUGAR [7], CADR [8] and ACQUIRE [9].

2.2.1.2 Hierarchical routing

Hierarchical routing is based on a hierarchical routing structure of the network in which nodes do not directly communicate with the sink but rather with other hierarchical nodes like the cluster-heads.

In hierarchical routing, energy is mainly achieved by: (1) Data reduction thanks to data aggregation performed by the hierarchical nodes, and (2) Short range communications. In

addition to energy saving, hierarchical routing enhances scalability. Based on the type of the network topology, we distinguish the following classes for the hierarchical routing.

1. **Cluster-based:** In this case, nodes are classified into clusters. Each cluster has a cluster-head which is selected among cluster members. Cluster-heads aggregate data from their cluster members and transmit them towards the sink. This allows nodes to save energy as they only communicate with their cluster-heads. However, if they are far from their cluster-head, they may expend excessive energy in communication. The cluster-heads can communicate with the sink directly via long range transmission or multi-hop transmission through other cluster-heads.
   In LEACH [10], any node selects the cluster-head from which it receives the strongest signal strength. LEACH is energy efficient because: (1) a randomized rotation of cluster-heads is used in order to spread energy usage over multiple nodes (2) cluster-heads aggregate data, (3) nodes within a cluster are activated successively: if they are not transmitting they turn off their radios, (4) few energy is consumed in radio communication because nodes communicate just with their cluster-heads, and only these cluster-heads send data to the base station. An optimization of LEACH is given by Two-Level LEACH (TL-LEACH) [11], where cluster-heads form two levels of data aggregators: the cluster heads from the first level aggregate data from cluster-heads of the second level. A similar strategy is considered in [33] where a hierarchy of clusters is defined: aggregators of level i aggregate data of aggregators from level $i-1$. Compared to a single level of clusters, this hierarchical clustering consumes less energy since the number of nodes that need to communicate with the base-station is reduced. TEEN [14] also relies on two levels of clusters. TEEN is cluster-based and data centric routing that reduces the data transmissions based on the sensed data attributes. Indeed, nodes are allowed to transmit sensed data if the sensed value is beyond a hard threshold HT, or if the change in this value is greater than a soft threshold ST. Consequently, TEEN is not suitable for applications requiring periodic data transmissions. APTEEN [15] is an extension of TEEN that adapts the periodicity or threshold values according to user needs and application types. EECS [12], like LEACH, forms clusters where a cluster-head is the node with the highest residual energy. Other examples of this class are given by TSC [16] and [17].

2. **Tree-based:** In tree-based routing, sensor nodes are organized into a tree where any node aggregates data sent by its children in this tree. The main issue is how to build an energy-efficient tree. Kuo et al. [34] prove that constructing the minimum energy consuming tree is NP-complete, and that Energy Cost(SPT) $< 2 *$ Optimal Energy Cost, which means that the Shortest Path Tree heuristic, SPT, is a 2-approximation of the problem. However, the drawback is that nodes close to the sink will be selected by many nodes and hence consume their energy quickly. To avoid this problem, other heuristics are considered like both the distance to the sink and the residual energy as in [35, 36]. Eskandari et al. [37] consider

another parameter which is the sum of the residual energy of nodes along a path divided by the path length. A second parameter is the distance to the sink. Further, each node has a predetermined maximum number of children to balance energy consumption between network nodes. Another example of this category is [38] where nodes are classified into clusters, and cluster-heads are organized in a tree topology. Further, authors propose a scheme where the sink dynamically adapts the data aggregation period according to the past aggregation quality (number of packets that could be aggregated).

3. **Chain-based:** The key idea behind the chain-based aggregation is to organize the nodes into a chain. Nodes in the chain aggregate packets and transmit them to a close neighbor in the chain. An example of this category is PEGASIS [13]. PEGASIS avoids the overhead of cluster construction and organizes nodes in a chain where any node receives from and transmits to the closest neighbor. This results in small transmission distances for most of the nodes and reduced power consumption for transmission. Further, nodes alternate to be the head of the chain and transmit aggregated data to the sink. However, PEGASIS introduces high delays for nodes at the end of the chain. CCPAR [39] divides the network into clusters where nodes inside the cluster form a chain and transmit data along the chain to their cluster-head. Cluster-heads are also organized in a chain, so they avoid communicating directly to a far base station. The issue with chain-based clustering is the high data transmission delays for nodes at the end of the chain.

4. **Grid-based:** Grid-based clustering has attracted a lot of attention because of its simplicity and scalability. In grid-based clustering nodes are organized into grids and communicate directly with a cluster-head. This cluster head is responsible for aggregating data and routing them. An example of this category is GAF [19] which is also a geographic routing. GAF is proposed for mobile ad hoc networks and can be used for WSNs too. GAF divides the network into grids and conserves energy by turning off unnecessary nodes in each grid cell. Indeed, only the node having the highest energy level in the grid cell is active for a certain period and ensures routing. The protocols [40] and GROUP [41] belong to this category.

2.2.1.3 Geographic Routing

Geographic or location-based routing protocols optimize the routing process using geographic information. For instance, GEAR [18] propagates queries to the target region instead of flooding the whole network like Directed Diffusion. Besides, intermediate nodes are chosen based on their remaining energy and their distance to the target region. Other examples of location-based protocols are DREAM [20], SPAN [21] and SPEED [22].

2.2.1.4 Mobility-based Routing

The key idea of these protocols is that nodes wait the passage of a mobile node and route their messages toward it. Low-power nodes can save power since they have only to transmit to a short range node. Besides, link errors are reduced. The main drawback of these protocols is that they induce high delays. For instance, in [23], nodes transmit their data to a mobile sink via a multi-hop communication. Consequently, there is a change in the set of nodes close to the sink which are more likely subject to battery depletion. Thus, energy consumption is balanced. Similarly, the work in [24] considers a mobile sink and evaluates the energy dissipation for different mobility patterns. Some other approaches like [25] use mobile relays, called the data MULES (Mobile Ubiquitous LAN Extension) which travel among nodes, pick up data from them, buffer them, and deliver the data collected to access points.

2.2.1.5 Energy-aware Route Selection

In this category, route selection is based on energy criteria like the residual energy as in REAR [26], [28] and [27]. However, the main drawback is that some intermediate nodes are overused and thus may fail. Other protocols try to select the route that consumes the minimum energy as in [29]. This technique may enhance the total power consumption of the overall network but fails to prolong the lifetime of the nodes on the path of minimum energy. That is why other schemes like EOLSR [30] and [31] take into account both criteria and select routes that consume the lowest energy while using nodes with high residual energy.

2.2.1.6 Concluding Remarks about Energy Efficient Routing Protocols

Table 2.2 provides examples of routing protocols and explains how they achieve energy efficiency.

From this table we notice that many routing protocols achieve energy efficiency by combining many techniques. For instance, LEACH combines data aggregation with the sleep mode. This strategy enhances the energy saving. Cumulating the benefits provided by several techniques enhances the performance of the routing protocol. However, the issue is how to keep low the overhead of the protocol. Data aggregation is a very suitable technique in data gathering applications. Also, it is a natural strategy in hierarchical routing for instance where clusters aggregate data from their cluster members. Increasing the data aggregation period allows nodes to reduce the number of packets transmitted and hence save energy and bandwidth, but this may result in higher data delivery delays. Hence, a tradeoff must be found between the data delivery delays and the resources (bandwidth and energy) consumed. Reducing the number of transmissions is highly related to data aggregation as it is evidently its consequence. That is why, protocols that perform data fusion reduce the number of transmissions. Moreover, reducing the number of transmissions can also be achieved either by reducing the number of transmitting nodes like in the geographic routing and in the optimized flooding, or by reducing the number of data transmitted by avoiding data redundancy

Table 2.2: Examples of energy efficient routing algorithms

Routing algorithms		**How energy efficiency is achieved?**			
Class	**Examples**	**Data aggregation**	**Number of transmissions reduced**	**Nodes energy level awareness**	**Sleep mode**
Data Centric	SPIN [4]	x	x		
	SPIN-2 [4]	x	x	x	
	Directed Diffusion [5]	x	x		
	Rumor [6]	x	x		
Hierarchical	LEACH [10]	x	x	x	x
	PEGASIS [13]	x	x		
	TEEN [14]	x	x		
	APTEEN [15]	x	x		
Geographic	GEAR [18]		x	x	
	GAF [19]		x	x	x
Mobility-based	[23]		x		
	[25]		x		
Energy-aware route selection	REAR[26]			x	
	EOLSR [30]		x	x	

for instance. This energy efficient feature is required whenever the allocated bandwidth is small (which is usually the case in wireless ad hoc and sensor networks).

In our work, we extend the known energy efficient routing protocol EOLSR by adapting it to data gathering applications. This routing considers energy level of intermediate nodes and the cost of routes is energy-aware. Besides, we combine this routing with a node activity scheduling scheme.

2.2.2 Protocol Overhead Reduction

In wireless ad hoc and sensor networks, the wireless communications consume much more energy than local computation [32]. That is why many researches work at optimizing the overhead of the protocols destined to these networks. And generally, this is achieved by optimizing the overhead of the control message exchange (mainly the flooding process), and by tuning the periodicity of the messages.

1. **Flooding optimization:** In classic flooding, called also "blind" flooding, any node should retransmit a packet if it receives it for the first time. This scheme is not energy efficient because of: (1) the redundant transmissions, indeed it is not obligatory that all nodes re-transmit a packet to ensure the delivery to all nodes, (2) the presence of collisions. That is why, many protocols rely on a subset of re-transmitting nodes selected according to specific criteria.

   Basically, two techniques are used to optimize flooding in wireless ad hoc networks. First, a connected dominating set of nodes is built as in [42]. A dominating set DS of a graph is a subset of nodes such that any node is either in DS or has a neighbor in DS. Further, only nodes in the dominating set forward the broadcast message if it is received for the first time. Notice that finding the minimum connected dominating set is NP-complete [43]. Second, other solutions rely on the construction of the Multi-Point Relays, MPR, as used in OLSR [44]. Each node builds its minimum set of MPRs that allows it to reach all its 2-hop neighbors. The MPR flooding means that a node forwards a message if it receives it for the first time from a node that selected it as MPR.

   The drawback of the above mentioned solutions is the need to store 2-hop neighborhood information at each node. The paper [45] compares different variants of flooding applied to RPL (IPv6 Routing Protocol for Low power and Lossy Networks) routing protocol for WSNs. One of the results of the comparison is that the MPR-based broadcast incurs the highest delivery ratio and the lowest delays.

   In the literature, we can find other solutions that do not require knowledge of the entire network topology. For instance, in counter-based solutions [47], a node does not re-transmit a packet if it overhears the same message from its neighbors for more than a prefixed number of times. In a distance-based scheme [47], a node discards its retransmission if it overhears a neighbor within a predetermined distance retransmitting the same message. Other methods are based on gossiping [48] where a node forwards a packet with a specific probability.

2. **Adaptive control message periodicity:** This category aims at avoiding the unnecessary transmissions by tuning the periodicity of the control messages. Two famous techniques are illustrated by the Trickle and the Fisheye algorithms. The trickle algorithm [49] is designed for code propagation and maintenance in WSNs. It dynamically adjusts the transmission rate depending on the freshness of the code the sensors have. Indeed, a node does not send a packet if it recently overhears a similar information. The principle of the Fisheye technique [50] is simple: a node exchanges routing information more frequently with its nearer nodes, and less frequently with farther nodes. In [51], authors extend the OLSR protocol with the Fisheye method to enable it to support large and dense networks.

2.2.3 Duty Cycle and Node Activity Scheduling

As the energy consumed in sleeping state is smaller than the energy consumed in any other state, keeping sensor nodes in the sleep state is a good way to save energy. However, to prevent network partition and message loss when some nodes are in the sleep state, node activity scheduling is needed. We speak about **the duty cycling** which is presented in the literature as the periodic wake-up scheme. **The duty cycle** means also the ratio of the listening period length to the wake-up period length.
Depending on their vision of the network and the stack layer they focus on, these methods approach the scheduling problem at high or low granularity:

1. **High granularity:** The scheduling problem is approached at a high level granularity: the network is seen as a set of nodes cooperating to monitor a given area like in a surveillance application for instance. Among these nodes, there are nodes that are qualified redundant from an application point of view. These nodes are turned off while non redundant nodes are needed to fulfil the application requirements. Consequently, these protocols do not obligatory determine the schedule of each sensor itself but rather determine the set of active nodes sufficient to perform application tasks. This set will be active for a specific period that can be determined based on the residual energy of the nodes for instance. The set of active nodes can be re-evaluated each activity period, depending on the application requirements. This class of algorithms does not depend on the medium access method and is implemented at an upper layer.

2. **Low granularity:** These solutions depend on the MAC layer and are in general integrated in this layer. They deal with each non redundant node individually to schedule its transmissions and receptions and allow it to sleep if it is neither transmitting nor receiving. The activity of each node is re-evaluated at each time slot. In other words, these methods determine for each time slot the set of nodes having to be active, and the set of nodes that can sleep.

Notice that the first and second class of solutions can be used jointly: after having selected a set of active nodes, these nodes are scheduled at the time slot level.

In the following, we detail these two classes. The second one is classified according to the medium access: CSMA/CA, TDMA or hybrid.

2.2.3.1 High Granularity

The goal of these protocols is to build a set of active nodes, such that only nodes belonging to this set must be active, all other nodes can sleep to keep their energy. In this type of solutions, the number of deployed nodes is supposed to be higher than the minimum number required to perform the monitoring activity (i.e nodes are redundant). Active node selection

is determined based on the problem requirements (e.g. area monitoring, network connectivity, power efficiency).

In [52], Cardei and Du introduce a new energy saving method where the sensor nodes are organized into a maximum number of disjoint sets. The sets are activated successively to monitor the whole region while nodes in the other sets are in a low-energy sleep mode. The problem of maximizing the number of these sets is NP-complete. However, this centralized protocol assumes that all nodes are synchronized and does not take into account the real node energy consumption. This work has been extended in [53], where a node can belong to several sets to improve network lifetime. In [54], a distributed and localized solution is proposed. It consists in selecting a connected dominating set of sensor nodes. Only the nodes of this set are active. Each node has a priority that is given by its residual energy. A node whose function is ensured by the dominating set can sleep if and only if: 1) the dominating set is connected, 2) all its neighbors have at least one neighbor in this set, 3) all nodes in the dominating set have a higher priority than itself. GAF [19] is an anergy-aware geographical routing protocol. GAF distributes nodes into a virtual grid such that nodes in one cell are equivalent from routing point of view. Indeed, based on its location provided by a GPS, each node determines the cell it belongs to. Each node broadcasts how long it will be in the active state. Its neighbors from the same cell use this information to schedule their sleep/awake periods such that they sleep as long as the existence of an active node per cell in the virtual grid is achieved. Notice that this routing scheme considers that all nodes belonging to the same cell are equivalent. It is not often the case from the application point of view: the final destination may be in the sleep state when another node wants to transmit data to it. Another example is given by the PEAS [55] algorithm. Unlike [52], PEAS selects a set of active nodes which stay in working state until they deplete their energy. Consequently, this scheme allows the appearance of holes until another set of active nodes is constructed. In [56], Cho et al. proposed ESP, a distributed node scheduling where each node switches from the sleep to the active state if there is no active neighbors in vicinity, or if its sensing area is greater than a predefined threshold. Further, an active node computes its working time based on its remaining energy.

2.2.3.2 Low Granularity

The principles of these solutions is to allow a node to sleep, whenever it is neither transmitting nor receiving. These solutions can be classified in three classes depending on the medium access:

- **CSMA/CA**:

Many energy efficient CSMA/CA solutions use the RTS/CTS exchange preceding any unicast data transmission. The goal is to enable the neighbors of the sender and the neighbors of the receiver to sleep during the communication to avoid any wasteful idle listening

and overhearing problems. However, the RTS/CTS mechanism increases the overhead and reduces the protocol efficiency. Hence, it is not adequate in case of short messages which is usually the case in WSNs. Examples of solutions based on CSMA/CA are: S-MAC [57], T-MAC [58], D-MAC [59], and OMAC [60].

For instance, the main goal of S-MAC is to reduce the energy consumption of WSNs, while supporting good scalability and collision avoidance. S-MAC is based on sleep-listen scheme. Each node prepares its activity schedule (the time of its next sleep) and exchanges it with its 1-hop neighbors. After receipt of such schedule from its 1-hop neighbors, each node deduces when these neighbors will be awake, it updates its schedule such that it wakes up and listens to them in order to receive data destined to it. However, it is possible that a node can not adapt its schedule to one of its neighbors schedule because of a schedule loss for instance. In this case, if this node wants to send data to one receiver, it must wait until this receiver wakes up, which increases the network latency. To reduce energy consumption, nodes store the duration needed for communications they are not involved in, and sleep for these durations. These mechanisms require a phase of synchronization between nodes. Many other variations of S-MAC have arisen such as T-MAC [58] with an adaptive duration of the active period, D-MAC [59] that reduces network latency, O-MAC [60] that improves the throughput.

• **TDMA**:

TDMA based protocols allocate time slots to each node and schedule medium access based on these slots. In general, energy efficiency is achieved by allowing nodes to sleep when they are neither receiving nor transmitting and by avoiding collisions.

In TRAMA [61], only nodes having data to send contend for a slot. The node with the highest priority in its neighborhood up to 2-hop wins the right to transmit in the slot considered. Each node declares in advance its next schedule containing the list of its slots and for each slot its receiver(s). The cost that TRAMA pays to support the adaptivity to the traffic is its complexity. FLAMA [62] is an adaptation of TRAMA to WSNs supporting data gathering applications. Its supports only the communications of a node with its parent and its children in the data gathering tree.

FlexiTP [63] is a TDMA-based protocol in which a slot is assigned to one transmitter and one receiver. All other nodes can sleep during this slot. Slots are assigned such that no nodes that are 1-hop or 2-hop away transmit in the same slot. In this protocol, nodes build a tree rooted at the data aggregation sink and run a neighbor discovery phase. The slot assignment order is given by a deep-first search of the tree. A node selects the first available slot in its neighborhood up to 2 hops and advertises its schedule. The exchange of this schedule allows any node to determine the conflicting slots and select slot(s) to transmit, slots to receive data from each child, and slots to receive from the parent. Medium access in the neighborhood discovery and slot assignment phases is done according to CSMA/CA. An additional slot is used for fault tolerance purposes. Indeed, FlexiTP supports link failure and node appearance in the network by updating the slot schedule locally and disseminating the updated schedule

to nodes in interference range. Note that a node does not aggregate data from its children before sending them to its parent. Which means that the transition between idle, transmit, receive activities are frequent and increase with the network density. This may impact the data gathering delays and the energy consumed by a node. This solution does not support immediate acknowledgement. We speak about **immediate acknowledgement** when the receiver uses the slot of the sender to transmit its acknowledgement.

Among the TDMA-based algorithms, there are algorithms that rely on **the graph coloring theory** to achieve an energy efficient scheduling. Indeed, the network is modeled as a graph and the nodes or edges of this graph are colored. At the MAC layer, this coloring is exploited as follows: a color is mapped to a time slot during which all nodes having this color can transmit. Examples of these algorithms are TDMA-ASAP [64] and SERENA [67].
For instance, TDMA-ASAP is designed for data gathering applications but does not support neither communications with broadcast nor immediate acknowledgement. It aims at providing spatial reuse, saving energy and decreasing the end-to-end delays. TDMA-ASAP colors the data gathering tree and allows each node to sleep if it is neither transmitting nor receiving data. It consists in a centralized *level-by-level node coloring* algorithm and *slot stealing algorithm*. It performs level by level coloring starting from the leaves such that any child is scheduled before its parent. Authors presented also an enhancement to this coloring which is the L-level coloring. Indeed, after each color c of each level l is assigned, the leaf nodes from L lower levels (less deep levels than l) are assigned the same color c if they do not have conflicts with any of the nodes in level l. The L-level coloring improves the end-to-end delays but results in higher complexity scheduling. The slot stealing is used in case of light traffic load in order to avoid empty slots to be unused and to adapt to various traffic conditions. Any node listens for unused slots owned by any of its one-hop neighbors that has the same parent. It can then steal any slot if the owner of this slot is not using it. Contention is used to prevent collision between multiple stealers.

SERENA [67] (SchEdule RoutEr Node Activity) is a localized and decentralized node activity scheduling based on node coloring. SERENA assigns colors to nodes in such a way that the number of colors used is small and any two nodes having the same color can transmit simultaneously without interfering. A node has to wake up in the slot of its color and the slots of its 1-hop neighbors to receive data sent to it if any, and sleeps the remaining time. In the following chapter, we are going to detail the principles and the properties of this algorithm and enhance its performances regarding the application types and the wireless communications failures.

• **Hybrid**:

Hybrid protocols use CSMA/CA as a baseline MAC scheme in low contention, and use TDMA schedule to enhance contention resolution under heavy contention.
For instance, the goal of Z-MAC [65] is to optimize the bandwidth efficiency of the MAC protocol. It is based on DRAND [66] which is a distributed randomized time slot scheduling

that ensures that no two nodes that are 1 or 2-hop away are assigned the same slot. After DRAND slot assignment is achieved, ZMAC, unlike TDMA, allows any node to own additional slots that are not used by any node within the 2-hop neighborhood of this node. Furthermore, depending on the contention level in the network, any node performs a carrier sense and transmits when the channel is clear. An owner of the slot has a higher priority than a non-owner one. Any node must stay awake during the slots assigned to its neighbors in order to be able to receive the message sent by one of them. From the energy point of view, nodes are not allowed to sleep during the activity period because nodes are allowed to send during any time slot. We can notice that Z-MAC, like TDMA-ASAP, does not allow an immediate acknowledgement of unicast messages, while this acknowledgement is important in wireless communication to confirm the correct reception of the packet.

To summarize, Table 2.3 presents a classification of the aforementioned scheduling algorithms. To determine the classification criteria, we look for the properties of the methods presented (distributed/centralized, probabilistic/deterministic, vertex/edge scheduling), types of communication supported (unicast, broadcast, tree) and application supported (general, data gathering). **General application** is an application where any node can communicate with any other node.

Table 2.3: Classification of scheduling algorithms

	central. distrib.	**determinist. probabilist.**	**vertex edge**	**communication unicast/tree/ broadcast**	**application supported**
TRAMA	distrib.	determin.	edge	unicast+broadcast	general
FLAMA	distrib.	determin.	edge	unicast in a tree	data gathering
ZMAC-DRAND	distrib.	random.	vertex	unicast+broadcast	general
TDMA-ASAP	central.*	determin.	vertex	unicast in a tree	data gathering
FlexiTP	distrib.	determin.	edge	unicast+broadcast	data gathering
SERENA	distrib.	determin.	vertex	unicast*+broadcast	data gathering

Legend: unicast*: unicast with immediate acknowledgement:
central*: only the centralized version is described in [64].

From this table, we notice that the algorithms vary depending on the type of the application and communication supported. These parameters define the ability of the solution to adapt to different types of applications. Besides, being aware of the application supported is also of a major benefit. However, in the studied algorithms, there is no awareness of the network topology characteristics: the density of nodes, the type of the topology, the memory constraints of nodes, etc.

In our work, we take into account these features by proposing a scheduling method adapted to both general and data gathering applications. The type of the topology (e.g grids), the

density of nodes and their memory constraints are also taken into account in our work (see Chapters 4 and 5).

2.3 Conclusion

Energy efficiency has always been a major concern in wireless ad hoc networks and especially in WSNs. Standards dedicated to these networks deal with this issue like IEEE 802.15.4 for instance. In this chapter, we reviewed the energy conservation strategies and classified them. The main focus of the remaining of this work is one of the techniques used to save energy which is the scheduling of nodes activities. To apply it, we start by using the graph coloring algorithm which is the subject of the next chapter.

Chapter 3

Graph Coloring and Node Activity Scheduling in Wireless Ad hoc and Sensor Networks

Introduction

Graph coloring is a famous optimization problem. It is a special kind of graph labeling and consists in assigning colors to vertices or edges of a graph subject to some constraints. An example of these constraints is the distance that prevents two adjacent vertices to share the same color for instance. Graph coloring was proved to be particularly useful to a large number of diverse fields, mainly networking. For instance, it is used in cellular networks to allocate frequencies. Further, typically, vertex coloring is used for node scheduling and edge coloring is used for link scheduling. In our work, we use the coloring to allocate time slots to nodes and to schedule their activities. As we will prove, the advantages of the coloring are multiple: It optimizes the network resources usage (energy, bandwidth, time,...).

This chapter contains 4 parts:

1. In Section 3.1, we present a state of the art about vertex and edge coloring and describe two applications of the graph coloring in networking: frequency allocation and time slot assignment.

2. In Section 3.2, we explain how the graph coloring is used to schedule the nodes activities. Moreover, in the literature, it is usually assumed that the coloring should ensure that a color of a node can only be used beyond its 2-hop neighborhood. In our study, we generalize this assumption and define a h-hop coloring where h is a positive integer such that at nodes sharing the same color are at least at $h + 1$ hops. We present also two coloring modes depending on the application type: the general mode and the strategic mode. We end this section by proving the NP-completeness of the coloring problem in its two modes.

3. In Section 3.3, we discuss the main issues of applying the graph coloring in real wireless environment.

4. In Section 3.4, we focus on the known coloring algorithm SERENA. We describe its rules and then, we present an optimization of this algorithm by adapting it to the application requirements and the wireless communications failures.

3.1 State of the Art: Graph Coloring

Graph coloring was firstly mentioned in 1852 by August De Morgan while he tried to color a map with 4 colors such that no two adjacent countries have the same color. Nowadays, applications of graph coloring are multiple and cover many domains. Depending on the elements of the graph that are colored, we distinguish two types of coloring: vertex or node coloring and edge coloring. In both types of coloring, the performance of the algorithm is usually measured by **(1)** the number of colors used [64, 67] and **(2)** the running time [102]. In this section, we present a state of the art about vertex and edge coloring. We then detail two of the applications of graph coloring in networking: frequency allocation and time slot assignment.

For simplicity and clarity reasons, we adopt the following notations to present the vertex and edge coloring methods.

- Let G be a graph with n vertices, Δ is the degree of the graph[1].
- Two vertices are said interfering if they are adjacent, that is neighbors.
- Two edges are said interfering if they are incident at the same vertex.
- Two vertices (respectively edges) are conflicting, if they have the same color although they are interfering.
- A color conflict occurs if there are two conflicting vertices or edges.
- A coloring is said valid, if it does not contain any color conflict.

3.1.1 Vertex Coloring

Vertex (node) coloring consists in coloring each vertex of the graph such that two adjacent vertices, that is vertices linked via an edge, have not the same color while minimizing the number of colors used. Coloring a graph G with k colors is equivalent to partitioning the vertices into k subsets where the nodes in one set are the nodes able to share the same color. The smallest number of colors that can be used to color a graph G is the **chromatic**

[1]The degree of a graph is the maximum degree of its vertices. The degree of a vertex in the graph is the number of edges incident to this vertex.

number of G, denoted χ. Finding the chromatic number of a graph has been shown NP-complete in [43] for the general case, whereas graphs with maximum vertex degree less than 4, and bipartite graphs[2] can be colored in a polynomial time.

3.1.1.1 Bounds on the Chromatic Number

Two trivial bounds for the chromatic number are given by $1 \leq \chi \leq n$ for any graph. Indeed, assigning distinct colors to distinct vertices always yields a valid coloring. Obviously, for a complete graph[3] $\chi = n$. For a bipartite graph, only 2 colors are needed to color the graph. Another bound is given by $\chi \geq \frac{|V|}{\alpha}$ where α is the independence number[4] of G. Indeed, let G be a graph colored with k colors and V_i is the set of nodes having the color i. Let $|V| = |V_1| + |V_2| + \ldots + |V_k|$. Hence, we have $n = |V| \leq \alpha\chi$, and so $\chi \geq \frac{|V|}{\alpha}$. Consider the biggest clique [5] of size ω, then the vertices of the clique should have different colors as they are all adjacent, so $\omega \leq \chi$. Also, we have $\chi \leq \Delta + 1$ [72].

3.1.1.2 Overview about Existing Works

Vertex coloring has received a lot of attention from researchers. The first methods were **centralized** like for instance [73, 74, 75]. Some methods are deterministic whereas others are **randomized** where nodes select colors randomly. For example, in [73], an uncolored vertex chooses randomly a color not already used by any neighbor. However, since two neighbors can select the same color at the same time, a color conflict may occur and should be resolved. Consequently, if any node discovers that any of its neighbors has chosen the same color, it selects another one. This algorithm is simple and fast and runs in $O(\log n)$. However, it does not use any mechanism to reduce the number of colors used so it produces $\Delta + 1$ colors. Consequently more sophisticated mechanisms are required to optimize the number of colors used.

As the coloring problem is NP-complete, **heuristics and approximation algorithms** are applied. In [76], Brélaz presents Dsatur which is a centralized greedy algorithm. Dsatur colors first the vertex with the highest number of already colored neighbor vertices. Similarly, in DLF algorithm (Distributed Largest First) [77], each node is assigned a priority that determines its coloring order. The priority of a vertex is equal to its degree, because it is known that it is much better to color the vertices in a largest degree first order. Authors show that the complexity of their algorithm is $O(\Delta^2 \log n)$. A classic general heuristic is the FirstFit [81] coloring algorithm that assigns the first available color to the uncolored node with the highest priority.

[2] A bipartite graph is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V; that is, U and V are independent sets.

[3] A complete graph is a graph in which every pair of distinct vertices is connected by an edge.

[4] An independence number of a graph is the cardinality of the largest (vertex) independent set.

[5] A clique of a graph G is a subset of its vertices such that every two vertices in the subset are connected by an edge.

Other algorithms use **local search** approaches [93, 94]. Local search heuristics operate in a search space of solutions denoted S. For every solution $s \in S$, a neighborhood $N(s) \subset S$ is defined. A local search method starts at an initial solution, and then moves repeatedly from the current solution to a neighbor solution in order to find better solutions, measured by an appropriate objective function. The passage from one solution to the next solution is called a move. The challenge for these methods is to determine the optimal neighborhood of a solution and the optimal move. The most studied local search strategy is tabu search called TabuCOL which was originally proposed by Glover [78]. It starts by generating an initial random solution which contains typically color conflicts. Then, the heuristic iteratively modifies the color of a vertex to decrease progressively the number of conflicts. Moreover, at each iteration, the move that maximizes the cost function is performed. A specific number of last moves are stored in order to avoid loops. Although these pure local search algorithms produce remarkable results for small graphs, it is no longer the case with large random graphs [94]. By constraint, an alternative that was used is the extraction of large independent sets (called also stable sets)[6] from the graph to obtain a smaller residual graph easier to color (see for instance $EXTRACOL$ that needs 2.5 hours to color 1000 nodes with a density of 5 in [94]).

A whole class of results expresses properties related to the **worst-case performance of approximation algorithms**: they typically prove that, for any input graph of a given family, the coloring obtained by a given algorithm uses at most α times the optimal number of colors. Such an algorithm is denoted an α-approximation algorithm. **The approximation ratio of coloring algorithms** that is defined as the ratio of the number of colors obtained by the algorithm to the optimal number is of crucial interest. The reader can refer to [79] for approximation ratio of coloring algorithms in grids, triangular lattices and hexagonal graphs. Various approximation ratio are established depending on the priority, graph type and class of algorithms [80].

3.1.2 Edge Coloring

Edge coloring of a graph is an assignment of colors to the edges of the graph such that edges incident on the same vertex have different colors. Edge-coloring can be regarded as vertex-coloring of line graph. Coloring the edges of a graph G with k colors is equivalent to partitioning these edges into k subsets. The minimum number of colors for the edges of a given graph is called **the chromatic index** of the graph or **edge chromatic number** denoted χ'. Finding the edge chromatic number is not less complex than finding the chromatic number. Edge coloring is also NP-complete [85].

3.1.2.1 Bounds on the Edge Chromatic Number

For any bipartite graphs, Konig [84] proved that Δ colors are needed. If G is complete, then if n is even then Δ colors are needed, otherwise $\chi = \Delta + 1$. For a general graph, it is obvious

[6] An independent set or stable set is a set of vertices in a graph, no two of which are adjacent.

that all edges incident to the same vertex must be assigned different colors, we have $\chi' \geq \Delta$, where Δ is the degree of the graph. Another trivial bound is $\chi' \leq 2\Delta - 1$ because each edge has at most $2\Delta - 2$ adjacent edges. Vizing's theorem [82] states that the chromatic index verifies: $\Delta \leq \chi' \leq \Delta + 1$. Shanon [83] proved that every graph can be edge-colored with at most $3\Delta/2$ colors.

3.1.2.2 Overview about Existing Works

Durand et al [85, 86] apply the edge coloring to schedule the data transfer between servers and clients in bipartite graphs. The approach of color selection is distributed and randomized, and is similar to the one proposed in vertex coloring in [73]: Each node randomly selects a color for each of its edges and keeps this color if it is not used by any conflicting edge. If it is the case, the node with the lowest degree has to change the color of this edge. Marathe et al [87] propose a simple distributed algorithm where each edge has a palette of $(1 + \epsilon) \times \Delta$ colors. Each edge picks a color from this palette and checks if it does not conflict with the colors of the neighboring edges. If it is the case, the color is kept and removed from the palette of available colors. Otherwise, the edge performs a new attempt in the next round. As the edge coloring problem is NP-complete, heuristics are used, especially for large graphs. Authors of [90] compare the performances of different heuristics for edge coloring over standard benchmarks (taken from a list of 119 graphs given at CP2002) for small graphs (< 500 nodes).

In [92], Huang and Tzeng propose another algorithm that colors the edges of planar networks with $\Delta + 4$ colors, where $\Delta \geq 5$. Their idea is to give each edge a priority and let the higher-priority edges choose colors first.

There are algorithms that target graphs with unidirectional links. For instance, Herman et al [88] use [87] to propose a randomized oriented edge coloring of a tree. The algorithm includes the edge coloring of [87] as a first step, and then recolors edges randomly with probability p, and then assigns orientation to all edges to obtain a tree. Compared to [87], Herman et al improve the time complexity from linear to $O(polylog(n))$. However, there is no guarantee that this coloring is conflict-free.

Regarding the applicability of the coloring to wireless ad hoc and sensor networks, the aforementioned vertex and edge coloring algorithms cannot be used as they are.
First, the randomized coloring results in color conflicts. Conflict resolution has not a negligible complexity. Second, the running time is usually not estimated because it depends on the random initial colors. Although randomized solutions scale well with the network size, determinism is still be a requirement for these networks especially under time and energy constraints.
Third, most of the algorithms mentioned above are destined to general graphs. Although there are works that target large graphs, few of them take into account specific properties of the topology (nature of the links, the eventual heterogeneity of the node/link, etc). However,

we believe that such details are worth considering if we want to adapt the algorithm to a specific application especially in networking. We will discuss in Section 3.3 which considerations have to be taken into account by any coloring algorithm for wireless ad hoc and sensor networks.

3.1.3 Applications of Graph Coloring in Networking

Many coloring algorithms have been designed to be used in wired and wireless networks to make the medium access more efficient. In this section, we only present its two main applications in networking: the time slot assignment and the frequency allocation.

3.1.3.1 Frequency Allocation

There has been extensive research on channel allocation particularly on base station frequency/channel assignment in cellular networks. The fundamental problem of frequency assignment in cellular networks is to assign frequencies to base-stations so that every client is served by some base-station. The graph coloring is used to solve this problem like for instance [95]: each base station is assimilated to a vertex, and an edge links two base stations if they are interfering. The goal is to minimize the number of assigned frequencies and hence the number of colors since the available spectrum is limited and costly.

The frequency allocation problem arises also in wireless mesh networks [96] and WSNs [97]. The coloring is applied as follows: the radio transceivers are modeled as vertices and are linked to other interfering units. A color corresponds to a frequency and two interfering nodes do not share the same color. The main challenge of this coloring is to minimize the overall network interference like in [96] which applies the graph coloring associated to the Tabu search heuristic, and like [97] that targets the interference reduction in WSNs. Another goal that was defined is the optimization of the spectrum use by allowing users to sense locally available channels and utilize them opportunistically like in [98, 99].

3.1.3.2 Time Slot Assignment

Coloring is applied in time slot assignment mainly in wireless ad hoc and sensor networks. The time slot allocation is usually associated to slotted contention-free medium access methods. It is used to allocate slots to nodes (respectively to links) to schedule the activities of these nodes (respectively transmissions on these links). In general, edge coloring is used to model the link scheduling problem [100, 101] while vertex coloring is used to model the node activity scheduling problem [64, 67]. Indeed, any color is associated to one or more time slots during which all nodes or links that share this color can be scheduled. In edge coloring, two neighboring nodes can communicate during the color of the edge connecting them. Similarly, in vertex coloring, a node is allowed to transmit only during the slots corresponding to its color. Consequently, packet collisions are avoided. Further, the bandwidth usage is improved

thanks to parallel conflict-free transmissions.

In our work, we are interested in this kind of application of graph coloring as explained in Section 3.2.

3.2 Coloring for Node Activity Scheduling: Problem Formulation and Complexity

In this section, we formulate the node coloring problem and prove that it is NP-complete.

3.2.1 Objectives and Design Choices

3.2.1.1 Objectives

Given a wireless ad hoc or sensor network composed of a number of nodes, our aim is to build a contention-free cycle of minimum length containing the time slots where each node can transmit its messages. A good schedule is a schedule that allows nodes to save energy by turning off their radio as long as possible. Also, to ensure an optimal spatial reuse, non interfering nodes must be able to access the medium simultaneously. Of course, the application tasks must be totally achieved. The schedule must be conflict-free: any two nodes that interfere should not be scheduled simultaneously. Otherwise, the transmissions of both of them will collide and the sent packets will be lost. For this purpose, we propose to use graph coloring and build a slotted contention free schedule based on these colors. In the following, **we consider the special case of TDMA**. However, what is said can be applied to any slotted contention-free medium access. Figure 3.1 illustrates our methodology.

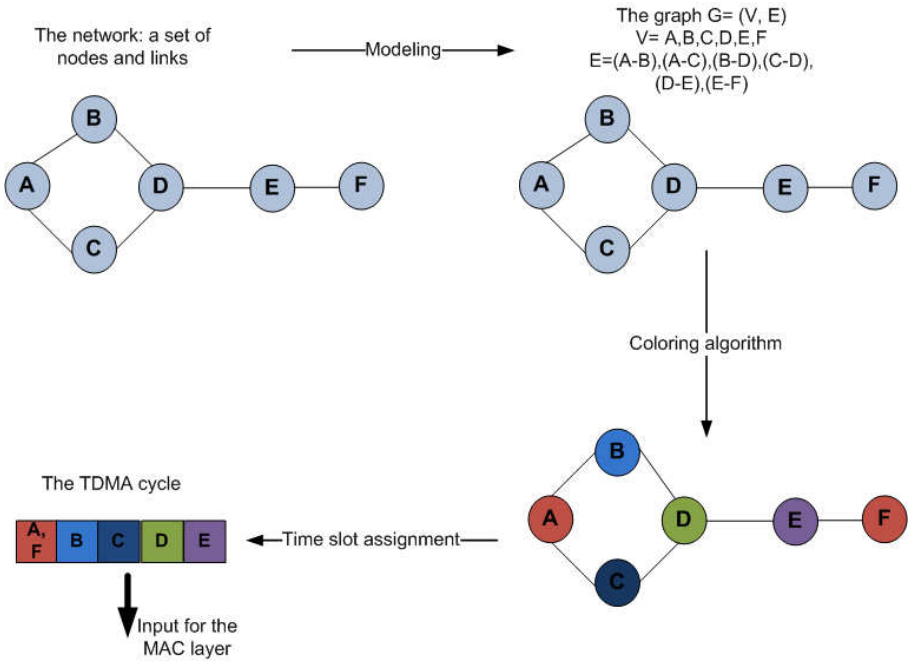


Figure 3.1: Strategy of application of coloring for node activity scheduling.

We now justify our design choices.

3.2.1.2 Why a Slotted Contention-free Scheduling?

This choice is motivated by the following:

- Medium access protocols that are contention-based protocols are clearly inadequate. They suffer from collisions and non deterministic delays especially under heavy traffic conditions.
- CSMA/CA based scheduling methods implies the use of the RTS/CTS exchange that results in an overhead and reduces the protocol efficiency. It is not adequate in the case of short messages which is usually the case in sensor networks.
- TDMA is a contention-free protocol where nodes share the same channel and time is divided into time slots. Each node transmits data in its allocated slots. Slots are usually organized in a cycle which is repeated periodically. Hence, it is obvious that the TDMA protocol is well adapted to collision-free packet transmission with QoS support.
- Furthermore, the TDMA deterministic scheduling is energy efficient. It avoids collisions that waste energy, does not need idle listening and allows low power devices to turn off their radio in time slots not allocated to them and during the inactive period. Although the use of TDMA requires synchronization between nodes, it is an efficient way of mitigating the limitations of CSMA based networks.

For these reasons, we adopt the TDMA-based scheme with the cycle format depicted in Figure 3.2.

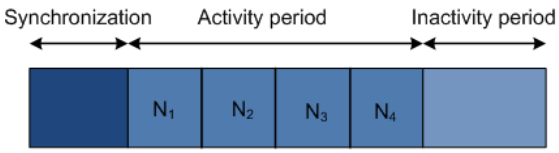


Figure 3.2: An example of generated cycle.

Assuming a TDMA-based medium access, one trivial solution to schedule nodes is to grant a time slot to each node. Then, each node has to be awake in the slots corresponding to its neighbors that transmit message to it, and to sleep during the remaining time. This solution has many drawbacks at many levels:

- **Delays:** The length of the activity period is equal to the number of nodes. This means that in a network of 1000 nodes, any node accesses the medium each 1000 time slots, which increases the end-to-end delays.
- **Bandwidth**: At any time slot, only one node is active. However, in a large network, nodes can be far enough to be able to transmit simultaneously without interfering. Otherwise, bandwidth is wasted.
- **Fairness regarding nodes demand:** In any application, nodes may have heterogeneous traffic demand. So, granting equal channel access to all nodes is not adequate.

In this thesis, we are guided by these remarks in order to enhance this basic TDMA scheme by avoiding the aforementioned drawbacks.

3.2.1.3 Why Graph Coloring?

For us, to model the activity scheduling problem as described in Section 3.2.1, the graph coloring is a natural and an immediate choice. Indeed, graph coloring has been used for many scheduling applications where it proved its efficiency to model the problem treated.

Obviously, the trivial stack layer where the node activity scheduling is used is the MAC layer since it manages the medium access. However, computing the schedule using the coloring has the advantage of an implementation flexibility: It can be at the MAC layer or at an upper layer. Consequently, in general, it avoids the coloring algorithm to deal with the MAC parameters like the slot duration, the slot time bounds. This fact is true for any graph labeling technique in fact. Moreover, it is easy to model a network by a graph and apply the vertex coloring on it (see Figure 3.1). Firstly, the network is modeled as a graph where the vertices represent the nodes and the edges represent the links of the network. The vertices are colored

with different colors. Second, two nodes share the same color if and only if they are not interfering. Finally, each color is mapped to one or more different time slots.

The expected benefits of coloring are threefold:

- At the bandwidth level where no bandwidth is lost in collisions, the overhearing and the interferences are reduced. Moreover, the use of the same color by several nodes ensures the spatial reuse of the bandwidth.
- At the energy level where no energy wasted in collision. Furthermore, nodes can sleep to save energy without loosing messages sent to them because of the schedule based on colors.
- At the delay level where the end-to-end delays can be optimized by a smart coloring ensuring data gathering in a single cycle.

The efficiency of a coloring algorithm is mainly measured by **the number of colors**. Indeed, this parameter defines the length of the activity period in the TDMA cycle. So, the smaller the number of colors is, the shorter the TDMA cycle is, and the smaller the end-to-end delays are. Further, other criteria should be considered:

- **The running time**: The complexity of any algorithm is measured by its running time. The coloring can be performed at the network deployment phase or as a maintenance step. In both cases, it should be fast in order to enhance the reactivity of the network.
- **The overhead**: Reducing the size of the message exchanged and the size of the data having to be stored is another goal. This allows the algorithm to adapt to ad hoc and sensor networks which are characterized by the small storage and computing capacity.

3.2.1.4 Edge or Vertex Coloring?

Regarding the medium access scheduling, and ssuming a bijective mapping between colors and time slots, the edge coloring is better than vertex coloring at the following levels [104]:

- **The concurrent transmissions:** The edge coloring can increase the concurrency of transmissions. To have a conflict-free scheduling, two 1-hop or 2-hop nodes cannot transmit at the same time slot. Consequently, a vertex coloring of the sample linear network depicted in Figure 3.3 would not assign the same color to nodes B and C (in this Figure, the integer next to the node (respectively to the edge) is the color of this node (respectively the edge)). However, with edge coloring and assuming directional links, links $(C \rightarrow D)$ and $(B \rightarrow A)$ are activated simultaneously, which means that nodes B and C share the same time slot.
- **The bandwidth:** With vertex coloring, a slot assignment to nodes restricts each node to transmit in at most one time slot in each TDMA cycle, irrespective of the number of neighbors a node might have. However, in edge coloring, each node has one time slot

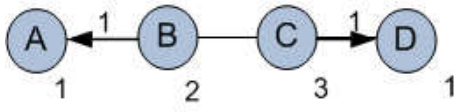


Figure 3.3: Difference between vertex and edge coloring: concurrent transmissions.

per neighbor in each cycle. Hence the bandwidth available to each node is proportional to its number of neighbors.

- **The energy conservation:** When time slots are assigned to nodes, each neighbor of the transmitting node has to switch to the receive state, irrespective of whether it is the intended receiver or not. Consequently, nodes waste energy in receiving frames not intended for them (overhearing). On the other hand, if each edge is assigned a slot, only the intended receiver switches to the receive state.

However, the edge coloring is not adequate in many cases:

- **Broadcast:** When colors are assigned to edges, each node has to repeat a broadcast messages as many times as the number of its outcoming links. This means that buffering the message until it is transmitted to all the neighbors is required. Meanwhile, in vertex coloring, all the neighbors of a node are active during the broadcast message transmission and can hence receive their message successfully.

- **Number of colors produced:** As the number of edges of a network is higher than the number of nodes, the running time of the coloring and the number of colors needed are higher in edge coloring. Further, to favor concurrent transmissions a color of a link corresponds to 2 time slots: one per each direction. Consequently, delays are increased. For instance, the vertex coloring of the graph of Figure 3.4 uses 3 colors while the edge coloring produces 4 colors.

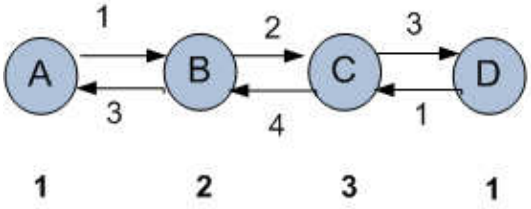


Figure 3.4: Difference between vertex and edge coloring: number of colors.

In our work, broadcast message are used to manage the network and build the neighborhood. For this reason, we choose the vertex coloring approach.

In the next sections, we introduce a formal definition of the coloring algorithm adapted to time slot assignment in ad hoc and sensor networks.

3.2.2 The Definition of the h-hop Coloring Problem

3.2.2.1 Network Model

Given the network, let $G(V, E)$ be the undirected graph representing the network topology. Each vertex $v_i \in V$ represents a network node with $i \in [1, n]$. For any two vertices v_1 and v_2 in V, the edge $(v_1, v_2) \in E$ if and only if the two nodes v_1 and v_2 are 1-hop neighbors.
Two nodes from this graph are said:

1. **1-hop neighbors** if there is en edge linking them.
2. **h-hop neighbors** if the minimum path between them is composed of h links.

3.2.2.2 Application Type

Depending on the application type, we define two modes of coloring.

1. **General mode**: This mode is adapted to general applications, where the destination of any unicast transmission can be any network node.
2. **Strategic mode**: This mode is adapted to data gathering applications, where a strategic node called the sink collects data from sensors. Nodes are in general organized into a tree rooted at this sink. The destination of any unicast transmission is either the parent or a child of the node in the data gathering tree.

3.2.2.3 Coloring Problem Definition

For simplicity reasons, colors are represented by natural integers, starting with zero. Based on the aforementioned model, we define the coloring problem as follows:

Definition 1. ***One-hop node coloring in the general mode*** *of G consists in assigning colors to nodes such that:*
(i) two adjacent nodes have not the same color;
(ii) the number of colors used is minimized.

With regard to the general mode, the strategic mode requires an additional constraint 1.

Definition 2. ***One-hop node coloring in the strategic mode*** *of G consists in assigning colors to nodes such that:*
(i) two adjacent nodes have not the same color;
(ii) the number of colors used is minimized;

Constraint 1. *Any node must select a color strictly higher than the color taken by its parent in the data gathering tree.*

In the graph theory, usually, coloring problems only prevent the 1-hop neighbors from sharing the same color, and allow other nodes to share it. However, in wireless networks, the interferences exceed 1 hop and are usually assumed to be limited to 2 hops. In our work, we will show that there are specific cases where even nodes lying at a distance higher than 2 must have different colors. That is why, we generalize the study of the coloring problem and define the h-hop coloring problem for any integer $h \geq 1$. We can easily extend one-hop node coloring to **h-hop node coloring**, for any integer $h \geq 1$. h-hop node coloring is also called, in [103] for instance, **h-distance node coloring** and 1-hop coloring of **the h^{th} power of the graph**[7] G.

Definition 3. ***h-hop node coloring in the general mode*** *is assigning colors to nodes such that:*
(i) no two nodes that are k-hop away, $1 \leq k \leq h$ use the same color;
(ii) the number of colors used in minimized.

Similarly,

Definition 4. ***h-hop node coloring in the strategic mode*** *is assigning colors to nodes such that:*
(i) no two nodes that are k-hop away, $1 \leq k \leq h$ use the same color;
(ii) the number of colors used in minimized.
Constraint 1. *Any node must select a color strictly higher than the color taken by its parent in the data gathering tree.*

A 1-hop or h-hop coloring is said **valid** if it verifies the following definition:

Definition 5. *For any integer $h > 0$, a h-hop node coloring in the general mode is said **valid** if and only if any two nodes that are k-hop neighbors, with $1 \leq k \leq h$ have not the same color.*

Definition 6. *For any integer $h > 0$, a h-hop node coloring in the strategic mode is said **valid** if and only if any two nodes that are k-hop neighbors, with $1 \leq k \leq h$ have not the same color, and the color of any node is strictly higher than the color of its parent in the data gathering tree.*

This coloring is said **optimal** if:

Definition 7. *A valid h-hop node coloring in the general mode (respectively in the strategic mode) is said **optimal** if and only if no valid h-hop node coloring in the general mode (respectively in the strategic mode) uses less colors than this coloring.*

[7]The h^{th} power of a graph G is obtained by adding an edge between any two vertices at a distance h or less.

3.2.3 The NP-completeness of the Coloring Problem

It has been proved in [43] that the 1-hop vertex coloring problem is NP-complete. In this section, we will demonstrate that h-hop ($h \geq 1$) vertex coloring problem in both general and strategic modes is NP-complete.

Theorem 1. *The decision problem of h-hop (h≥1) vertex coloring in both general and strategic modes is NP-complete.*

The coloring problems presented by Definition 3 and Definition 4 are two optimization problems. To evaluate their complexity, we define their associated decision problem denoted **k-color h-hop coloring**:

***k-color h-hop coloring in the general mode**: Can graph G be colored with k colors in the general mode (k is a positive integer smaller than the vertex number), such that two nodes that are p-hop neighbors with $1 \leq p \leq h$ have not the same color?*

***k-color h-hop coloring in the strategic mode**: Can graph G be colored with k colors in the strategic mode (k is a positive integer smaller than the vertex number), such that two nodes that are p-hop neighbors with $1 \leq p \leq h$ have not the same color, and any node has a color higher than the color of its parent in the data gathering tree?*

Further, we denote **k-color 1-hop coloring** the decision problem of the 1-hop coloring problem.

Our proof of Theorem 1 is done through the following steps.

• First, we prove the following lemma:

Lemma 1. *The k-color h-hop coloring problem in both general and strategic modes for $h \geq 1$ are in NP.*

Proof: Given a h-hop coloring of G, $h \geq 1$ we can check in polynomial time ($O(n^h)$, where n is the number of nodes) that the coloring produced by a given h-hop coloring algorithm in both general and strategic modes does not assign the same color to two nodes that are p-hop neighbors with $1 \leq p \leq h$, and that the total number of colors is k. Further, for the strategic mode, the verification of the superiority of a color of a node relative to the color of its parent (Constraint 1) is still time polynomial. ■

• Second, we define a reduction f of the k-color 1-hop vertex coloring problem that has been shown NP-complete in [43], to a k'-color h-hop coloring problem in the general and strategic modes, with k' a positive integer smaller than the nodes number. This reduction should be polynomial in time. Based on this reduction, we prove the two following equivalence:

Equivalence 1. *A k'-color h-hop vertex coloring problem in the general mode has a solution if and only if a k-color 1-hop vertex coloring problem has a solution.*

Equivalence 2. *A k'-color h-hop vertex coloring problem in the strategic mode has a solution if and only if a k-color 1-hop vertex coloring problem has a solution.*

In general, to demonstrate that a problem is NP-complete based on another problem that is known to be NP-complete, the required reduction should allow us to show that we can find a solution for the first problem if and only if we can find a solution for the second problem. In our case, we should transform a graph $G(V, E)$ into a graph $G' = (V', E')$ and show that finding a k-color 1-hop coloring of $G(V, E)$ can lead to finding a k'-color h-hop coloring of $G'(V', E')$ in the general mode and in the strategic mode separately, proving Equivalence 1, and Equivalence 2.
To prove this, the construction must obey the following constraints.

Constraint 2. *Any two nodes v_1 and v_2, 1-hop away in G must be at most h-hop away in G'.*

Thus, two nodes that are assigned different colors by 1-hop coloring of G are also assigned different colors by a h-hop coloring of G'.

Constraint 3. *Similarly, any two nodes v_1 and v_2, 2-hop away in G must be at least $(h+1)$-hop away in G'.*

Consequently, the reduction separates any two 1-hop neighbors v_i and v_j of the initial graph G by a set of nodes such that the distance between them in the new graph G' is at most h hops. V' is obtained from V by adding new nodes. We denote $V' = V \cup V_1$. The definition of these new nodes depends on h parity.
In order to simplify the determination of k', the number of colors used for the h-hop coloring of G', we add a new constraint to the transformation:

Constraint 4. *Any two nodes in V_1 must be at most h-hop away. Moreover, any two nodes $u \in V$ and $v \in V_1$ must be at most h-hop away.*

Thus, in a h-hop coloring of G', nodes in V_1 have different colors. Similarly, no node in V can reuse a color used by a node in V_1.
The transformation proceeds as follows, depending on the parity of h:

- **First case: h is odd**: see the example for $h = 5$ in Figure 3.5.

  ◦ *Definition of V'*

In this case, we first define $h' = (h-1)/2$ copies of V, denoted U_i and h' bijective functions f_i, with $i \in [1, h']$:

$$\begin{array}{rcccl} f_i & : & V & \rightarrow & U_i \\ & & v & \mapsto & f_i(v) = u_i \end{array}$$

We say that u_i is the associated or correspondent node to v at level U_i. Now, we can define the set $V_1 = \cup_i U_i \ \cup \{u_0\}$ and $V' = V \cup V_1$, $\forall i \in [1, h']$, where u_0 is a new node introduced to meet Constraint 4. Node u_0 is a neighbor of all nodes in $U_{h'}$.

  ◦ *Definition of E'*

To build the set E', four types of links are introduced. We then have: $E' = E_1 \cup E_2 \cup E_3 \cup E_4$ where:

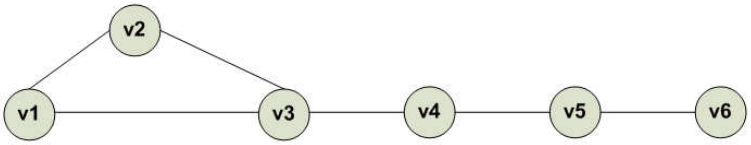
- $E_1 = \{(v, u_1)$ such that $v \in V$ and $u_1 = f_1(v) \in U_1\}$. Thus, each node v from the initial graph G is linked to u_1, its associated node from the set U_1 (see links of type e_1 in Figure 3.5).

- $E_2 = \cup_{l \in [1, h'-1]} \big\{(u_l, u_{l+1})$ such that $u_l \in U_l$ and $u_{l+1} \in U_{l+1}$ and $f_l^{-1}(u_l) = f_{l+1}^{-1}(u_{l+1})\big\}$. Each node u_j from U_j is linked to node u_{j+1} from U_{j+1} associated with the same node $v \in V$, that is $f_{j+1}^{-1}(u_{j+1}) = v$ *and* $f_j^{-1}(u_j) = v$ (see links of type e_2).

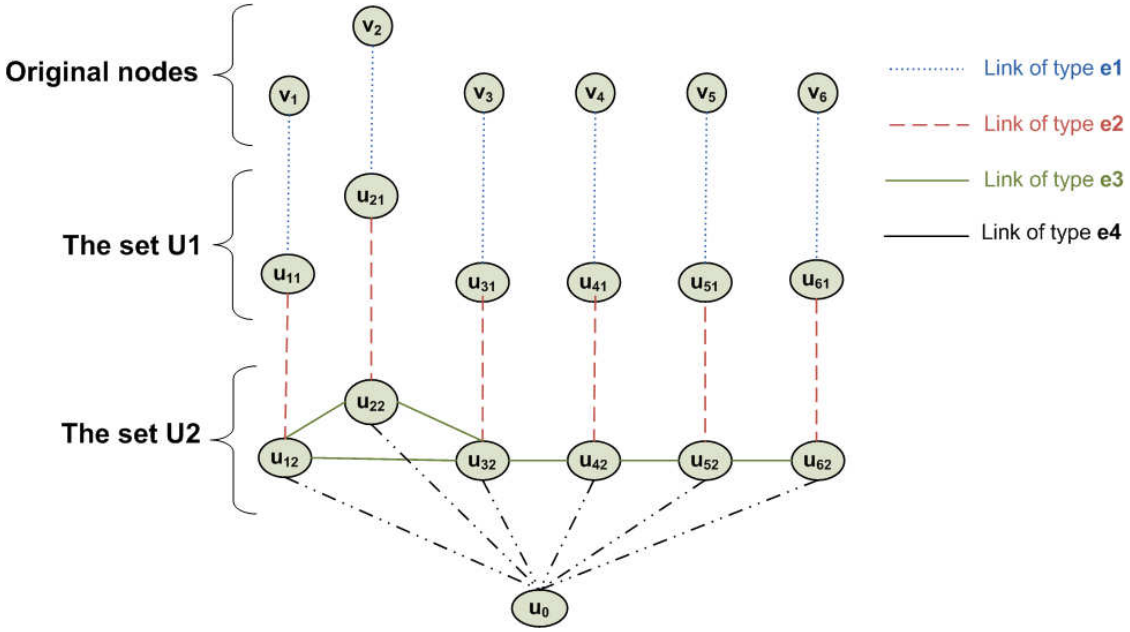
- $E_3 = \big\{(u_{h'}, v_{h'})$ such that $u_{h'}$ and $v_{h'} \in U_{h'}$ and $(f_{h'}^{-1}(u_{h'}), f_{h'}^{-1}(v_{h'})) \in E\big\}$. Two nodes $u_{h'}$ and $v_{h'}$ from $U_{h'}$ are linked to each other if their corresponding nodes in V are linked in E (see links of type e_3).

- $E_4 = \{(u, u_0)$ with $u \in U_{h'}\}$. Finally, the nodes in $U_{h'}$ are linked to the conjunction node u_0, which was added to meet Constraint 4 (see links of type e_4).

This construction is polynomial in time. An example of graphs G and G' with $h = 5$ is illustrated in Figure 3.5.



(a)



(b)

Figure 3.5: Example of: (a) Graph G; (b) Transformed graph G' for $h = 5$.

• **Second case:** h **is even**: see the example of $h = 6$ in Figure 3.6.

To build the graph G' when h is even, Constraints 2, 3 and 4 are considered. However, as the number of links to introduce between nodes in the initial graph G depends on the number of nodes to introduce between them, and thus, on h parity, the reduction is slightly modified.

◦ *Definition of* V'

In this case, let $h' = h/2$, we first define $h' - 1$ copies of V, denoted U_i and h' bijective functions f_i with $i \in [1, h' - 1]$:

$$\begin{array}{rcccl} f_i & : & V & \rightarrow & U_i \\ & & v & \mapsto & f_i(v) = u_i \end{array}$$

and the bijective function $f_{h'}$:

$$\begin{array}{rcccl} f_{h'} & : & E & \rightarrow & U_{h'} \\ & & e & \mapsto & f_{h'}(e) = u_{h'} \end{array}$$

We define the set $U_{h'}$. To each couple of nodes $(u_{h'-1}, v_{h'-1})$ in $U_{h'-1}$, we associate a node $u_{h'}$ in $U_{h'}$ if and only if there is an edge between $f^{-1}_{h'-1}(u_{h'-1})$ and $f^{-1}_{h'-1}(v_{h'-1})$. Now, we can define the set $V_1 = \{u_0\} \cup_i U_i$, $\forall i \in [1, h']$, where u_0 is a node introduced to model the data gathering tree in G'.

◦ *Definition of* E'

To build the set E', five types of links are introduced. We then have:
$E' = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$ where:

- $E_1 = \{(v, u_1)$ such that $v \in V$ and $u_1 = f_1(v) \in U_1\}$. Thus, each node v from the initial graph G is linked to u_1, its associated node from the set U_1 (see links of type e_1 in Figure 3.6).

- $E_2 = \cup_{l \in [1, h'-2]} \big\{(u_l, u_{l+1})$ such that $u_l \in U_l$ and $u_{l+1} \in U_{l+1}$ and $f_l^{-1}(u_l) = f_{l+1}^{-1}(u_{l+1})\big\}$. Each node u_j from U_j is linked to node u_{j+1} from U_{j+1} associated with the same node $v \in V$ (see links of type e_2).

- $E_3 = \{\, (u_{h'-1}, u_{h'}), (u_{h'}, v_{h'-1}) \,\}$ such that

$$\begin{cases} u_{h'-1} \textit{ and } v_{h'-1} \in U_{h'-1} \text{ and } u_{h'} \in U_{h'} \\ \text{and } f_{h'}^{-1}(u_{h'}) = (f_{h'-1}^{-1}(u_{h'-1}), f_{h'-1}^{-1}(v_{h'-1}). \end{cases}$$

  In other words, for each couple of nodes $u_{h'-1}$ and $v_{h'-1}$ in $U_{h'-1}$, we associate a node $u_{h'} \in U_{h'}$ if and only if $(f_{h'-1}^{-1}(u_{h'-1}), f_{h'-1}^{-1}(v_{h'-1})) \in E$. We then link $u_{h'}$ with $u_{h'-1}$ and $v_{h'-1}$ (see links of type e_3).

- $E_4 = \{(u_i, u_j)$ such that u_i and $u_j \in U_{h'}$ and $i \neq j\}$. This means that the nodes in $U_{h'}$ form a complete graph (see links of type e_4).

- $E_5 = \{(u_i, u_0) \forall\ u_i \in U_{h'}\}$. All nodes in $U_{h'}$ are linked to a node u_0 (see links of type e_5).

This construction is polynomial in time. The transformed graph G' for $h = 6$ of the initial graph G depicted in Figure 3.5.a is illustrated in Figure 3.6.

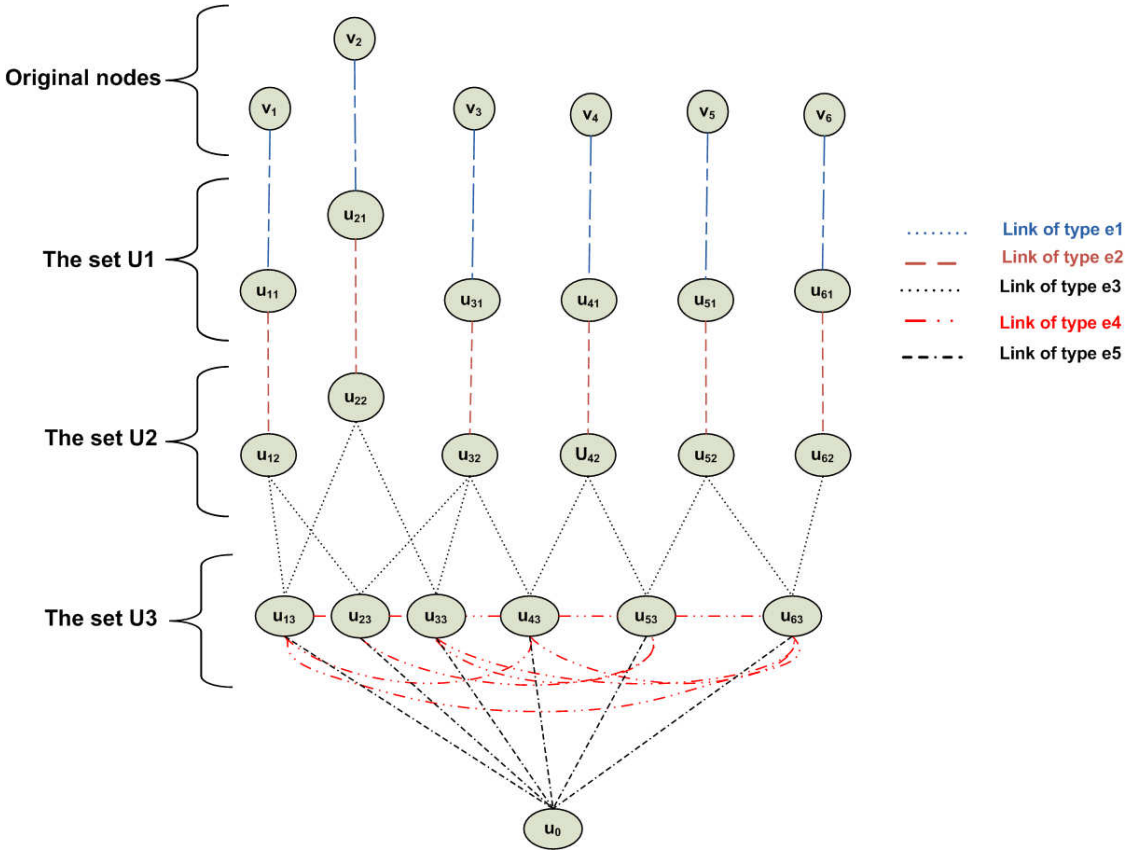


Figure 3.6: Transformed graph G' of G for $h = 6$.

We now show, that the k'-color h-hop vertex coloring problem in both general and strategic modes, for any value of $h \geq 1$ has a solution if and only if the k-color 1-hop vertex coloring problem has a solution. We have the following Lemma:

Lemma 2. *All nodes in $G' \setminus G$ are at most $(h-1)$-hop neighbors.*

Proof: By construction of G'. ■

Lemma 3. *To perform a h-hop coloring of the graph G', the number of colors taken by nodes in V_1 is equal to m with m is equal to $(h' \cdot n) + 1$ if h is an odd number, and $(h'-1) \cdot n + p + 1$ if h is an even number, where n is the number of nodes in G and p is the number of edges in G.*

Proof: From Lemma 2, all nodes in V_1 are at most $(h-1)$-hop neighbors. Hence, no color can be reused with h-hop coloring ($h \geq 2$) of G'. By construction of G', the number of these nodes is equal to $(h' \cdot n) + 1$ if h is an odd number, and $(h'-1) \cdot n + p + 1$ if h is an even number. ■

Lemma 4. *Any color of a node in V computed by a h-hop coloring of $G' = (V', E')$ cannot be used by any node in V_1.*

Proof: Let us consider any node $u \in V_1$ and any node $v \in V$. Let $d(v,u)$ be the number of hops between v and u. By construction, $d(v,u) = d(v,f_1(v)) + d(f_1(v),u)$. From Lemma 2, $d(f_1(v),u) \leq h-1$ and since $f_1(v)$ is a neighbor of v, we get $d(u,v) \leq h$. Hence, u and v must use different colors with h-hop coloring of G' for $h \geq 1$. ■

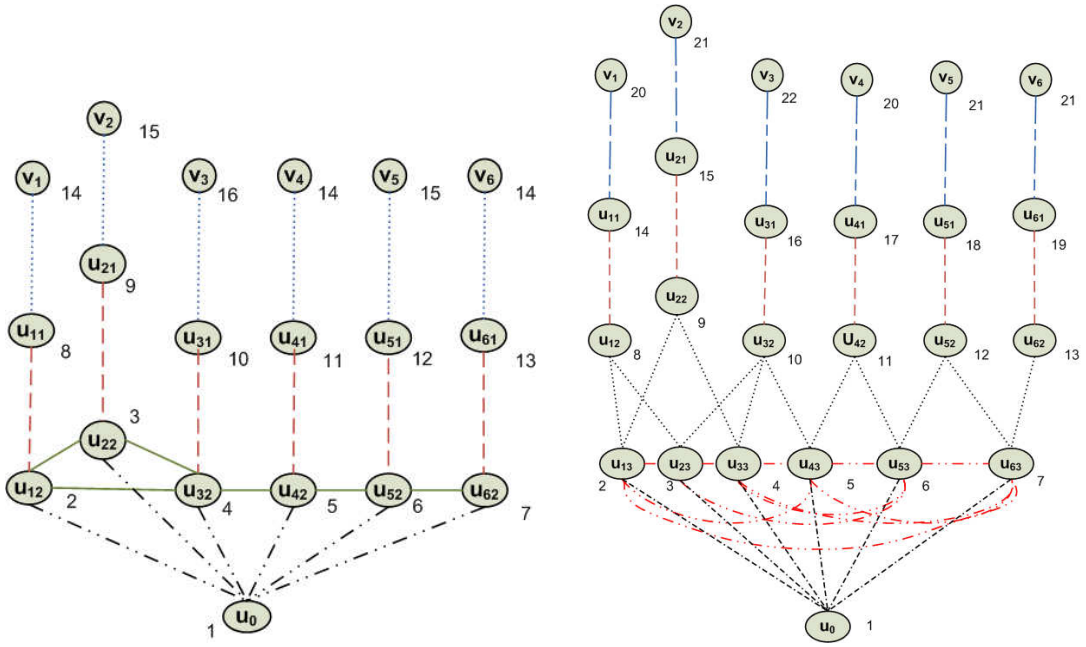
To complete the proof of Theorem 1, we now prove the following Lemma:

Lemma 5. *$G(V,E)$ has a one-hop coloring with k colors if and only if $G'(V',E')$ has a h-hop coloring in general mode with k' colors, with $h \geq 2$.*

Proof: Given a 1-hop coloring of G with k colors, we want to show that there exists a h-hop coloring of G' with k' colors as follows. According to Lemma 3, this h-hop coloring will use k colors for nodes in V and m colors for nodes in $V' \setminus V$ with m is equal to $(h' \cdot n)+1$ if h is an odd number, and $(h'-1) \cdot n + p + 1$ if h is an even number. From Lemma 4, colors used in V cannot be reused in $V' \setminus V$. It follows that there exists a h-hop coloring of G' with exactly $k' = k + m$ colors.

Now, let us assume that we have a h-hop coloring of G' with k' colors and we want to show that we can find a one-hop coloring of G with k colors. From Lemma 3, m colors are needed for h-hop coloring of nodes in $V' \setminus V$. From Lemma 4, colors used in V cannot be reused in $V' \setminus V$. Hence, $k' - m$ colors are used to color the nodes in V. Moreover, since any two nodes v_1 and v_2 in V that are one-hop neighbors in G are h-hop neighbors in G', by construction of G', we deduce that no two one-hop neighbors in G use the same color. Hence, we can find a valid one-hop coloring of G with $k = k' - m$ colors. ■

Figure 3.7 illustrates this coloring for $h = 5$, and $h = 6$ based on the original graph in Figure 3.5(a) (colors are the integers next to the node).



(a) (b)

Figure 3.7: The graph G' colored with (a) 5-hop coloring, (b) 6-hop coloring.

Lemma 6. *$G(V, E)$ has a one-hop coloring with k colors if and only if $G'(V', E')$ has a h-hop coloring in strategic mode with k' colors, with $h \geq 1$.*

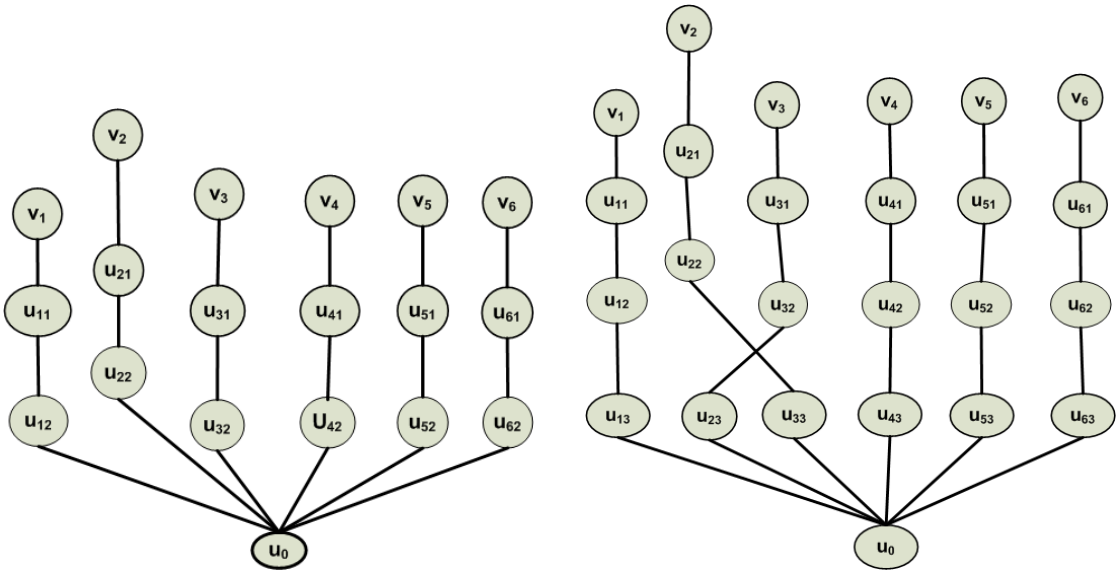
Proof: Given a one-hop coloring of G with k colors, we want to show that there exists a h-hop coloring of G' with k' colors in strategic mode such that Constraint 1 defined in Section 3.2.2.3 is met, as follows.

We start by building a tree T rooted at node u_0 from the graph G'. Nodes from V are the leaves of this tree (see Figure 3.8 as an example). In the case h is odd, any node v in V has as parent $f_1(v)$. Any node v_l in U_l with l a positive integer such that $1 \leq l \leq h' - 1$ has as parent the associated node from the level U_{l+1}. Any node in $U_{h'}$ has as parent the root u_0.

In the case h is even, T has as root the node u_0, any node v in V has as parent $f_1(v)$, and any node in U_l with l a positive integer such that $1 \leq l \leq h' - 2$ has as parent the associated node in U_{l+1}. Finally, we link the nodes in $U_{h'-1}$ to the tree. With any node $v_i \in U_{h'}$ $(1 \leq i \leq p$ where p is the number of edges in $G)$ linking two nodes u_i and u_j in $U_{h'-1}$ we associate as its child a node from the couple (u_i, u_j), such that this node has not yet a parent.

To color T, we start by coloring the node u_0 the root of the tree. Then, we color nodes level by level, to finally reach the original nodes in V. From Lemma 2 and Lemma 4, nodes in each level do not reuse colors from lower levels. Hence, each child has a color strictly higher than the color of its parent. ■

Figure 3.8 depicts the tree built from graph G' for $h = 5$ and $h = 6$, where only tree links are represented. The coloring of these graphs is similar to the one in Figure 3.7.



(a) (b)

Figure 3.8: Tree built from G': (a) h=5; (b) h=6.

3.3 Issues of Coloring Application in Real Wireless Environment

As proved in Section 3.2.3, the coloring problem is NP-complete. In fact, its complexity comes from the multitude of choices a node may have to select its color. Consequently, approximation and heuristics are in general used to find good solutions. Moreover, adapting the graph coloring to scheduling for wireless ad hoc and sensor networks implies further challenges, and many considerations have to be taken into account. Indeed, there are considerations relative to the wireless nature of the communications and others on the graph coloring problem itself. All these details make the application of the coloring in wireless ad hoc and sensor networks a difficult problem. In this section, we are going to list and discuss these issues.

- **Interferences**: To color any node, the first step is to determine its conflicting nodes, that is the set of nodes that are not allowed to share its color. This is equivalent to finding the value of the h parameter in the h-hop coloring problem, which itself implies the definition of the interference model. Determining the set of conflicting nodes of any node is crucial because: (1) If it includes less nodes than necessary the coloring will allow conflicting nodes to have the same color and hence collisions may occur and (2) If it includes more nodes than necessary, a color will be forbidden to a number of nodes higher than necessary, and hence the total number of colors will be high. Consequently, the set of the conflicting nodes must be finely chosen. We will prove that this set depends on the type of the application and the communications supported.

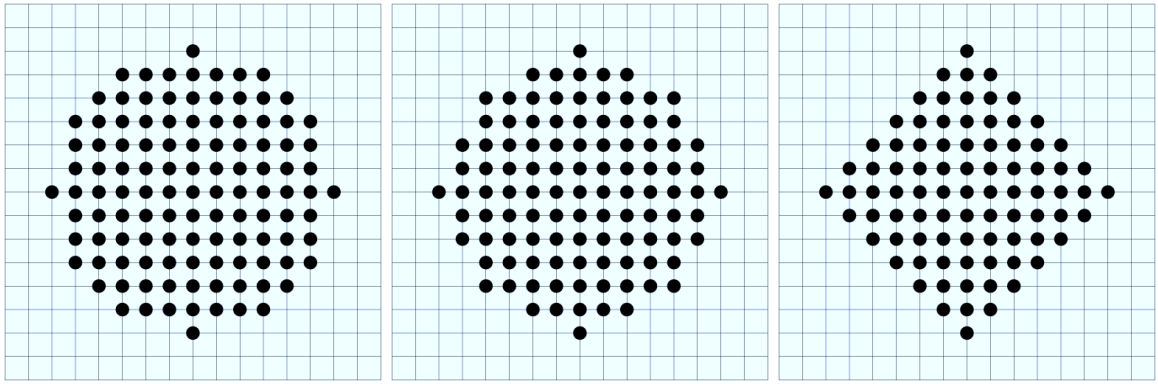
- **The transmission range**: The wireless interferences are function of the transmission range R. The value of R has to be taken into account in the coloring algorithm. Notice

also that while the 1-hop coloring seems to be easier than h-hop coloring for $h > 1$, it is not sufficient to change the value of R to switch from 1-hop coloring to h-hop coloring for $h > 1$. For instance, contrary to what one might think, 2-hop coloring of a graph with $R = 3$ is not the same as the 1-hop coloring of the same graph with $R = 6 = 3 * 2$. Figure 3.9 illustrates this fact. The example is taken in grid topologies, where the transmission range is expressed in grid step units. For any grid node u, Figure 3.9 depicts the nodes that cannot take the same color as u (which is the center of the grid in the figure) in case of:

(a) 1-hop coloring with transmission range=6;

(b) 2-hop coloring with transmission range=3;

(c) 3-hop coloring with transmission range=2.



(a) (b) (c)

Figure 3.9: Nodes having a color different than the color of the central node u in: (a) 1-hop coloring $R = 6$; (b) 2-hop coloring $R = 3$ and (c) 3-hop coloring $R = 2$.

These 3 sets are different, which means that 3-hop coloring of a grid with $R = 2$, is not equivalent to 1-hop coloring of the power 3 of this graph with $R = 6$, not equivalent to 2-hop coloring of the grid with $R = 3$. More generally, we have the following remark:

Remark 1. *For any graph G, for any transmission ranges R and $R' \geq 1$, for any integer h and $h' > 0$ such that $R' \cdot h' = R \cdot h$, h-hop coloring of G with transmission range R is not equivalent to h'-hop coloring of G with transmission range R'.*

This remark means that the coloring is a function of the values of h and R separately not the product $h * R$.

- **Unreliability of radio links**: Wireless networks are characterized by the link breakage and hence neighboring nodes departure. It can be easily shown that the breakage of a link does not compromise the validity of a coloring as no additional links are created. Whereas the creation of a new link may invalidate the coloring since nodes that were not conflicting may be closer and may interfere. The creation of new links can result

from nodes mobility or late arrival. In our work, we will discuss how these impairments are addressed by our coloring solutions (see Chapters 4 and 5).

- **The limited resources of nodes**: In ad hoc and sensor networks, nodes have limited storage and computing capacity. So reducing the overhead of the developed solutions is a real requirement. Also, as we mainly target the energy saving, our solutions must have a small time and message overhead.

- **The network and application properties**: Intuitively, the type of the topology is not a negligible parameter. We will show that being aware of the periodicity of the topology like for instance a grid topology has a great help to obtain a more efficient coloring. The application and the type of the communications contribute to the definition of the conflicting nodes of any node.

In the next section, we will show the existing coloring algorithm SERENA adapts to some of these issues, mainly: interferences, unreliably of radio links and application type.

3.4 Adaptivity of SERENA to Data Gathering Applications and Wireless Communications Failures

In Chapter 2, we briefly described a scheduling algorithm based on vertex coloring called SERENA [67] for SchEdule RoutEr Node Activity. This algorithm has been compared to other algorithms like ZMAC in [65] and has proved its superiority at many levels. For this reason, we focus in this section on SERENA and investigate in its adaptivity as follows. In Section 3.4.1, we describe the rules of SERENA. In Section 3.4.3, we use a cross layer approach with the application layer to adapt SERENA to data gathering applications using broadcast and immediate acknowledgement. In Section 3.4.4, we address the unreliable characteristics of wireless communications and show how to enhance the robustness of SERENA in the case of tree communications.

3.4.1 State of the Art: Presentation of SERENA Algorithm

In this section, we will introduce the general context and assumptions considered by SERENA. We will also present its requirements and its rules.

3.4.1.1 Assumptions and Requirements of SERENA

1. **Ideal environment:** SERENA considers an ideal environment defined as follows.

   Definition 8. *A wireless environment is said* ***ideal*** *if and only if the following assumptions are verified:*

   - Assumption A0: *All links are symmetric: if node u sees node v as a 1-hop neighbor, then conversely node v sees node u as a 1-hop neighbor.*

- Asssumption A1: *All links are stable. More precisely, link creation during or after the completion of the coloring algorithm is not taken into account.*
- Assumption A2: *Each node has a unique address in the network.*
- Assumption A3: *Any node does not prevent the correct receipt of any other node out of its transmission range.*

2. **Applications and communications:** SERENA algorithm is a generic algorithm since it adapts to various application requirements and communication types. Indeed, SERENA which can be implemented in the MAC layer or in an upper layer, establishes a cross layer with the application layer and with the MAC layer. The key idea of cross layer [108] is to exploit the dependencies and interactions between layers and allow them to cooperate and exchange useful information to obtain performance gains. For instance, providing knowledge about wireless channel conditions to routing, transport and application layers allows to design more sophisticated protocols.
   Accordingly, based on the type of the application, SERENA defines two functioning modes: (1) **general mode for general applications** where any node can communicate with any other node and (2) **strategic mode adapted to data gathering applications**. Moreover, in both modes, nodes can broadcast data if needed and transmit an **immediate acknowledgement**, that is the acknowledgement of a packet is sent in the same time slot as the packet. In fact, various factors, like fading, interference, multi-path effects, and collisions, lead to heavy loss rates on wireless links. Efficiently handling losses in wireless environments is very important especially for critical applications that require reliability. One way to improve end-to-end reliability in multi-hop paths is to employ hop-by-hop retransmissions. Accordingly, the medium access handles the acknowledgements that allow a sender node to verify if its message was successfully received. The immediate acknowledgement has in addition the advantage of reducing delays. Indeed, contrary to the deferred acknowledgement whose transmission can be delayed until the next slot granted to the sender node, the immediate acknowledgement is suitable for low latency traffic. Its implementation is simple, there is at most one packet whose acknowledgement is pending. Besides, it saves resources by reducing latency to schedule a deferred acknowledgement packet.
   To summarize, SERENA meets the following requirements.
   - *Req1 (general communications)*: Any node can communicate in unicast mode with any other node. This requirement aims at adapting to general applications.
   - *Req1' (tree communications)*: Any node can communicate in unicast mode only with its parent or its children in the data gathering tree.
   - *Req2 (immediate acknowledgement of unicast transmissions*): The receiver uses the slot of the sender to transmit its acknowledgement. Depending on the slot size, the sender can use its slot to retransmit if needed.

- *Req3 (broadcasts)*: Any node can broadcast information to its neighbors. Broadcast is a real requirement especially for control messages like the $Hello$ messages for instance.

In the following, we will describe how SERENA meets these requirements.

3.4.1.2 Rules of SERENA

1. **Generic rules of SERENA**
   SERENA is a distributed and localized algorithm. It proceeds by iterations or rounds (a round is an iteration where a node receives a message from each neighbor, processes it and transmits its own message) where each node periodically transmits a *Color* message to gather information useful for the coloring. Any node N proceeds as follows:
   (a) *N builds its set $\mathcal{N}(N)$*, denoting the set of conflicting nodes. For instance, when we speak about 2-hop coloring, $\mathcal{N}(N)$ contains all the neighbors up to 2 hops. In general, it is the task of the routing protocol to provide SERENA with the knowledge of the 1-hop and 2-hop neighbors via a neighborhood discovery process.
   (b) *N computes its priority*, denoted $prio(N)$. The priority of any node determines its coloring order.
   (c) *N colors itself according to its priority order in $\mathcal{N}(N)$*. That is, an uncolored node colors itself if it has a priority higher than any uncolored node in $\mathcal{N}(N)$.

   The general rules of SERENA are identical whatever the constraints enforced by the considered application. However, this solution is able to take advantage of the particularities of the application to obtain better performances. In other words, the application will pay for its requirements and not for a superset of them. For instance, the adaptation of SERENA to general an strategic applications implies a slight modification of its generic rules as detailed in the next section.
2. **SERENA rules for general and strategic applications**
   (a) **Determining the set of conflicting nodes:** It has been proved in [107] that this set depends on the requirements previously expressed in Section 3.4.1.1. In Tables 3.1 and 3.2, we present the conflicting nodes of any node N for different requirements ("ACK" stands for acknowledgement, "imm." stands for "immediate", and nodes M and N are two conflicting nodes).

	Without imm. ACK	With imm. ACK
Without broadcast	1 and 2-hop neighbors	1 and 2 and 3-hop neighbors
With broadcast	1 and 2-hop neighbors	1 and 2 and 3-hop neighbors

Table 3.1: Conflicting nodes for different application requirements in the general mode.

	Without imm. ACK	With imm. ACK
Without broadcast	N and M have the same parent M is 1-hop neighbor of the parent of N N is 1-hop neighbor of the parent of M	done in in this chapter
With broadcast	1 and 2 hop neighbors	done in this chapter

Table 3.2: Conflicting nodes for different application requirements in the strategic mode.

These different types of coloring are aware about all types of collisions that may occur in unicast, broadcast, or tree communications (see Table 1 and Table 2 in [107]). Consequently, the proposed types of coloring are collision-free.

In our work, we will complete Table 3.2 by determining the required type of the strategic coloring in case of broadcast and/or immediate acknowledgement support. We will also evaluate the cost of supporting these features.

(b) **Priority computation:** The priority of any node N is equal to the size of:

- $\mathcal{N}(N)$ in case of general communications. Indeed, node N and any node in $\mathcal{N}(N)$ cannot color themselves simultaneously. So, coloring the node with the highest number of conflicting nodes avoids them to wait for a long time before being able to select a color.
- $Descendants(N)$ in case of tree communications. This choice tends to alleviate the color constraints imposed to the descendants of the node and thus to reduce the number of colors used.

Ties are broken by the smallest node address or identifier. The priority heuristic choice is justified by simulation results published in [67, 105, 106].

(c) **Color selection:** Color selection is based on the following rules.

- *Rule R1:* any node N colors itself if and only if all nodes in $\mathcal{N}(N)$ having a higher priority than N are already colored.
- *Rule R2:* node N takes:
  - *Rule R2.1:* the smallest color unused in $\mathcal{N}(N)$ in case of general communications;
  - *Rule R2.2:* the smallest color unused in $\mathcal{N}(N)$ and higher than the color of its parent in case of tree-based communications. This rule has been introduced to minimize the delay needed to collect data by the sink. To illustrate this, consider the example in Figure 3.10. In this example, we assume that existing radio links are only those belonging to the tree. The colors produced by SERENA are presented by integers next to the node. Based on these colors, scheduling medium access according to the decreasing order of colors allows each node to aggregate the information received from its children before transmitting them to

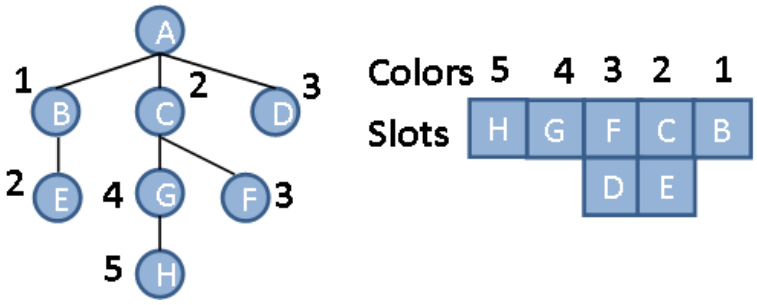


Figure 3.10: Example of SERENA 2-hop coloring applied to a tree with the corresponding slot assignment.

its parent. It is the case of the node C for instance that accesses the medium once all its children (nodes G and F) have transmitted their packets. Consequently, information from all nodes **can reach the sink in a single cycle**.

Remark 2. *Based on SERENA colors, any node is awake in its slots and the slots of its parent/children in the strategic mode and in the slots of its neighbors in the general mode.*

Remark 3. *Notice that applying Rule R2.2 implies that each node needs to buffer the packets received from its children. We assume that in our data gathering application, sensors generate a periodic traffic of small size and that data can be aggregated by any node and transmitted in a single time slot. This assumption is realistic especially when intermediate nodes perform some operations like computing the average value or the maximum value of the received data. However, there are cases where aggregation is not possible due to the size of the gathered data. This context will be studied in Chapter 6.*

3.4.2 Positioning of Our Contribution

In this chapter, we focus on the strategic mode of SERENA and optimize it as follows:

1. In case of tree communications, we specify SERENA rules for the two versions: communications supporting the immediate acknowledgement but not the broadcast, and communications with both broadcast and immediate acknowledgement.

2. We evaluate the benefit of using the strategic mode of SERENA for data gathering applications.

3. We compare different variants of SERENA depending on the communications supported.

4. So far, the proposed algorithm SERENA has assumed an ideal environment without topology changes. However, in wireless networks, nodes may fail and links may be broken, and hence topology may change. That is why, we adapt SERENA to such phenomenon and enhance its robustness against topology changes.

3.4.3 Adaptivity of SERENA Regarding the Application and Communication Requirements

In this section, we focus on the strategic mode of SERENA supporting both the broadcast and the immediate acknowledgement. We keep the same rules of SERENA and determine the set of conflicting nodes of any node depending on the communications supported. The questions we want to answer are: How SERENA is modified to be able to support these two requirements while keeping the same generic rules? And what is the cost of supporting these types of communications in terms of energy consumption and data delivery delays?

3.4.3.1 Strategic Mode of SERENA with Immediate Acknowledgement and without Broadcast

For each node, at least a subset of its 2-hop neighbors are conflicting (as depicted in Table 3.2) to ensure a collision free unicast communications. However, when the immediate acknowledgement is supported, 2-hop coloring is no longer sufficient because collisions may occur. The possible collisions in the case of 2-hop coloring are illustrated in Figure 3.11, where nodes N and M are 2-hop away and are assigned the same color.

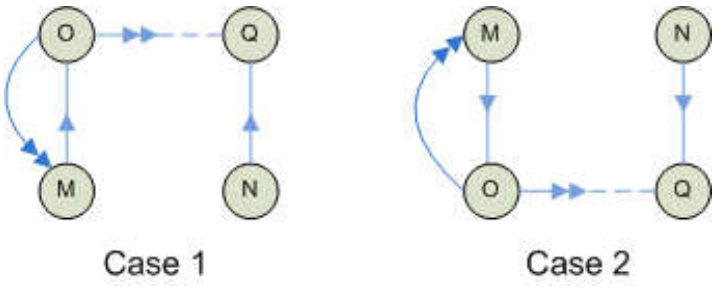


Figure 3.11: Collision between nodes N and M having the same color with two-hop coloring while supporting the immediate acknowledgement but not the broadcast.

As a legend for Figure 3.11 (and also Figure 3.12), we use the following convention: the link between a node and its parent is represented by a vertical plain line, the parent node being represented over its child node. A dotted line represents a 1-hop neighbor link that is not a parent link: it is either horizontal or diagonal. A single arrow represents the transmission of a data frame, whereas a double arrow represents the transmission of an acknowledgement.
Cases 1 and 2 correspond to a conflict at node Q between a unicast transmission by N and the acknowledgement of another unicast transmission originated from M. Consequently, for data gathering applications with immediate acknowledgement, to color itself, N should have a color different from:

- its 1-hop neighbors;
- its brothers;

- the 1-hop neighbors of its parent;
- the children of its 1-hop neighbors;
- the children of the one-hop neighbors of its parent: case 1 of Figure 3.11;
- the parents of the one-hop neighbors of its children: case 2 of Figure 3.11;

3.4.3.2 Strategic Mode in SERENA with Immediate Acknowledgement and Broadcast

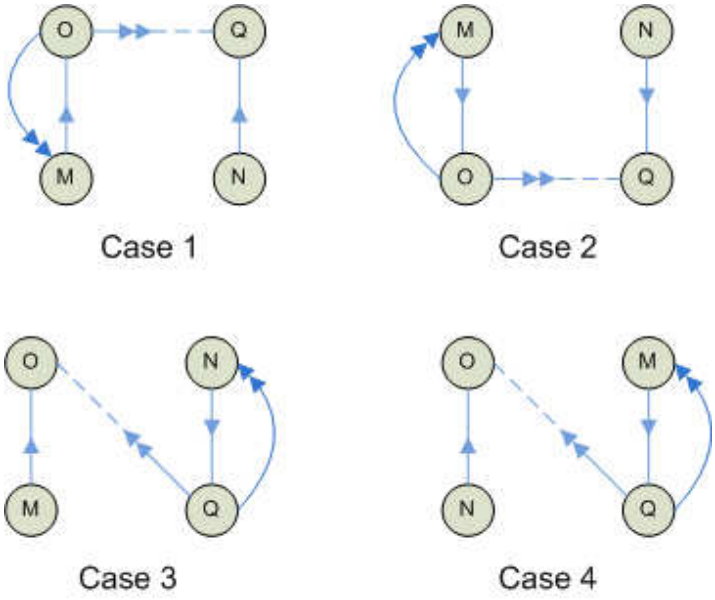


Figure 3.12: Collision between nodes N and M having the same color with two-hop coloring while supporting the immediate acknowledgement and the broadcast.

Broadcast communications (requirement *Req*3) requires a 2-hop coloring. Furthermore, with both broadcast and immediate acknowledgement, a subset of the 3 hop neighbors may conflict with the node as illustrated in Figure 3.12. Cases 3 and 4 correspond to a conflict caused by the acknowledgement of a unicast transmission and a broadcast transmission, originated from M in case 3 and from N in case 4. Consequently, to support both the immediate acknowledgement and the broadcast, conflicting neighbors of any node are:

- **Set 1**: its 1-hop and 2-hop neighbors;
- **Set 2**: the children of the 1-hop neighbors of its parent: case 1 of Figure 3.12;
- **Set 3**: the parents of the 1-hop neighbors of its children: case 2 of Figure 3.12;
- **Set 4**: the children of the 1-hop neighbors of its children: case 3 of Figure 3.12;
- **Set 5**: the parents of the 1-hop neighbors of its parent: case 4 of Figure 3.12.

To quantify the cost in terms of additional colors induced by the immediate acknowledgement and the broadcast, we compare the following variants of SERENA:

- without immediate acknowledgement and without broadcast;
- without immediate acknowledgement and with broadcast;
- with immediate acknowledgement and without broadcast;
- with immediate acknowledgement and with broadcast.

Also, we present a comparison of SERENA with TDMA-ASAP [64].

3.4.3.3 Simulation Results

1. **Simulation setup:**
   We developed a Java simulator. Wireless nodes are randomly deployed in a square network. The parameters we vary in our simulations are the number of nodes and the density. The transmission range is equal to $250m$. For a given number of nodes and a given density, we determine the network area where nodes are deployed. For a given number of nodes and density value, we generate 10 random topologies. Hence, each result is the average of 10 simulation runs. We build a tree of minimum path length to the sink. We start by the sink node. Any neighbor of the sink that has not a parent is linked to the sink node. We repeat the process until all nodes except the sink have a parent. Notice that for topologies with some disconnected nodes, we cannot build the tree. Such topology instance is not considered in the simulations.
   Concerning the neighborhood construction, we use the following definition:

   Definition 9. *1-**hop and** h-**hop neighborhood**: Two nodes u and v are 1-hop neighbors if and only if their distance is $\leq R$, with R is the radio range.*
   For any integer $h > 1$, any two nodes u and v are h-hop neighbors if and only if u is $(h-1)$-hop away from a 1-hop node of v.

2. **Cost and benefits of adaptivity to the application type:**
   As previously mentioned, the adaptivity of SERENA to the data gathering application mainly concerns two steps of the algorithm: the determination of the conflicting nodes, and the color selection (in particular the *Rule R2.2* which obliges a node to select a color higher than the color of its parent). In this section, we will discuss the cost and the benefits of this adaptivity regarding these two steps.
   (a) **Determination of the conflicting nodes:** Evidently, the size of the conflicting nodes of any node impacts the number of colors, the number of rounds, and the number of messages sent to achieve the coloring. To evaluate this impact, we compare the defined strategic mode supporting the broadcast and the immediate acknowledgement (denoted also SERENA tree coloring) with a general 3-hop coloring, where any node should have

a color different from the colors of its neighbors up to 3-hop. Notice that 3-hop coloring of the data gathering tree produces a collision-free coloring of the tree because the set of conflicting nodes of any node in tree communications supporting broadcast and immediate acknowledgement is included in the 3-hop neighborhood of this node.

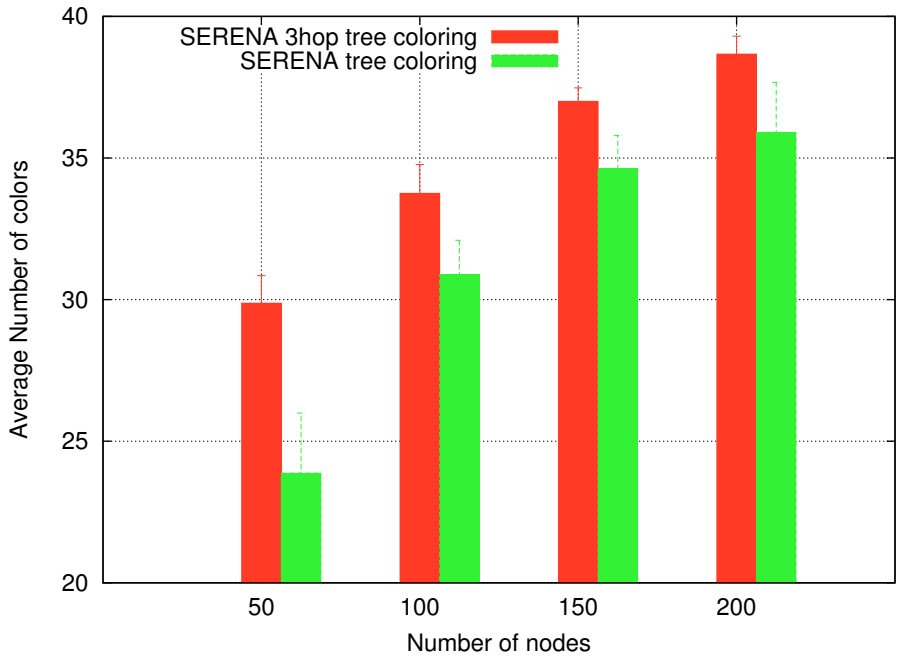


Figure 3.13: Gain in number of colors between the general and the strategic coloring of a data gathering tree(density=10).

From Figure 3.13, we notice that SERENA achieves a benefit higher than 6% for the density 10. In particular, this benefit exceeds 20% for 50 nodes. This result means also that despite the application of *Rule R2.2*, the strategic mode uses a smaller number of colors. Indeed, by restricting the set $\mathcal{N}(N)$, SERENA uses a smaller number of colors. This is confirmed by Figure 3.14 that illustrates the average number of conflicting nodes for any node. This number is higher in the general 3-hop coloring. The benefit in colors is perceived by the user in terms of higher energy saving (a node is awake for a duration equal to (1+the number of its neighbors) slots during each TDMA cycle), and also smaller end-to-end data gathering delays.

We compare also the overhead induced by the general and the strategic coloring in terms of number of rounds and average number of sent messages. The density is set to 10. Figure 3.15 shows that 3-hop general coloring of the tree has higher overhead. For instance, the average number of rounds increases from 83.7 to 109.6 for 100 nodes, and the average number of sent messages increases from 20.1 to 28.6 for 100 nodes.

All these results highlight how much the cross layer with the application improves the solution performance and reduces its overhead.

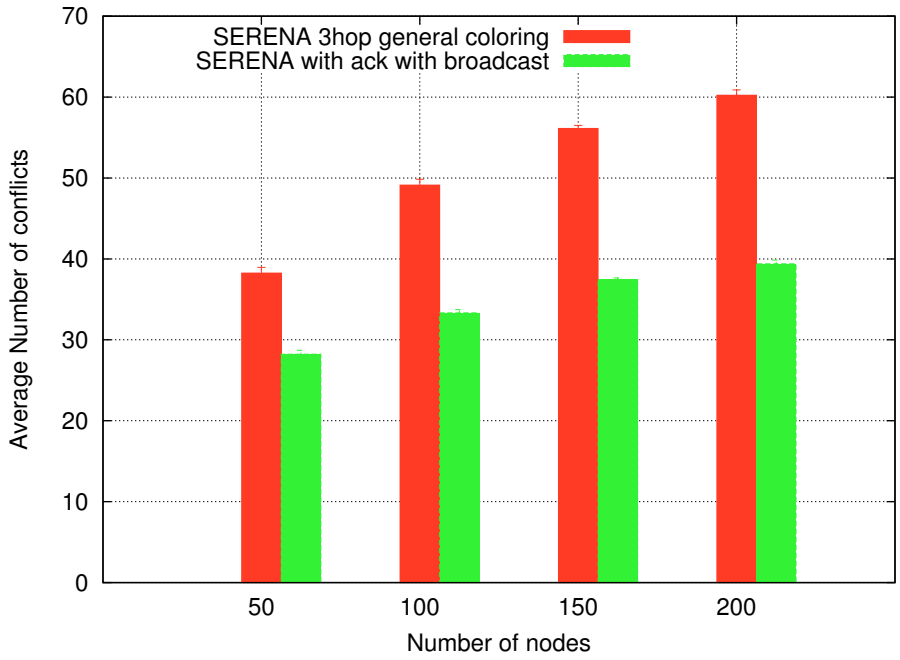
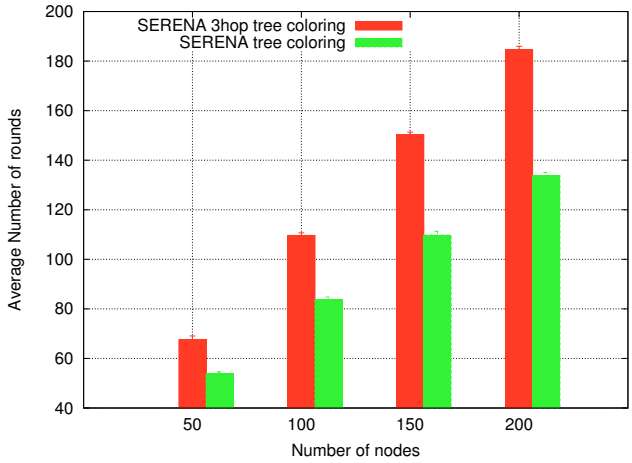


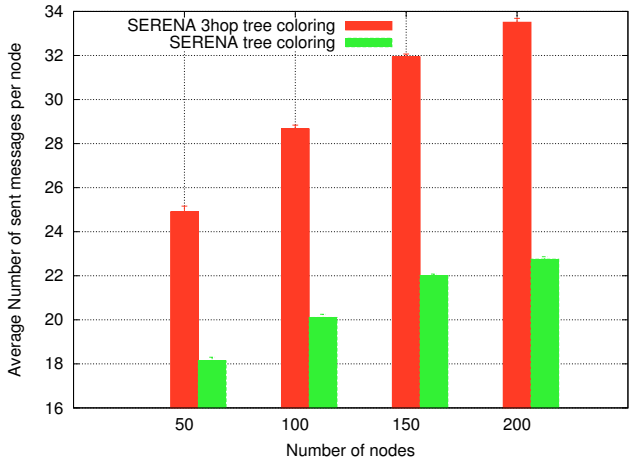
Figure 3.14: Average number of conflicting neighbors per node in general 3hop coloring and strategic tree coloring.

(b) **Color selection:** With *Rule R2.2*, data collected from sensors will be aggregated and forwarded to the sink in a single cycle. If this coloring rule was not introduced, the time needed by a data originated from a sensor to reach the sink could reach a number of cycles equal to the distance in hop number of this sensor to the sink. That would be unacceptable especially for time-critical applications. This result means that it is better to apply the strategic coloring mode in case of data gathering applications instead of the general mode. There are simulation results published in [107] (see Figure 10 and Figure 13 in this reference) that prove this benefit. Authors compared the end-to-end-delays; Assuming a cycle of 4 seconds, the general 3-hop coloring results in average end-to-end delays equal to 8.53 seconds, whereas in the strategic mode data reach the sink in 2.94 seconds.

This reduction in delays comes at a price which is an additional number of colors. To evaluate this cost, we compare the number of colors used in SERENA in the strategic mode while integrating or not the *Rule R2.2* as illustrated in Figure 3.16. Notice that in these two cases the number of rounds and the number of exchanged messages are the same. Figure 3.16 shows that at worst, the percentage of additional colors is 18.6% for 100 nodes and density=10. This overhead is light regarding the benefit in terms of delays. The number of additional colors increases with the number of nodes. Moreover, for 100 nodes with density=10, in average, 33.7 colors are needed for the 3-hop coloring of the tree, 30.8 colors are needed for the strategic coloring integrating *Rule R2.2*, and 25.1 colors are needed by the strategic coloring without integrating *Rule R2.2*. This means that the most impacting factor on the number of colors is the interferences and not the support of



(a) Number of rounds.



(b) Average number of sent messages.

Figure 3.15: Comparison between the strategic and the general coloring of a tree.

Rule R2.2. This is because the heuristic that we use (the priority of any node is equal to the number of its descendants) tends to alleviate the impact of *Rule R2.2* by favoring color reuse. Meanwhile, when a node has more conflicting nodes as in the 3-hop coloring, color reuse will evidently decrease, and hence the number of colors will increase.

3. **Cost of adaptivity to the communication types and comparison with TDMA-ASAP**

   In Figure 3.17, we compare the number of colors used for different variants of SERENA depending on the types of communications supported. We also compare the performance of SERENA with the algorithm TDMA-ASAP. TDMA-ASAP colors the tree, level by level, starting with the farthest level from the sink (*i.e.* the reverse of SERENA). A node receives a color strictly higher than the color of its parent. Two nodes having the same parent do not share the same color. A parent and a child cannot have the same color. Moreover, a node that is neighbor of the parent of another node does not share its color.

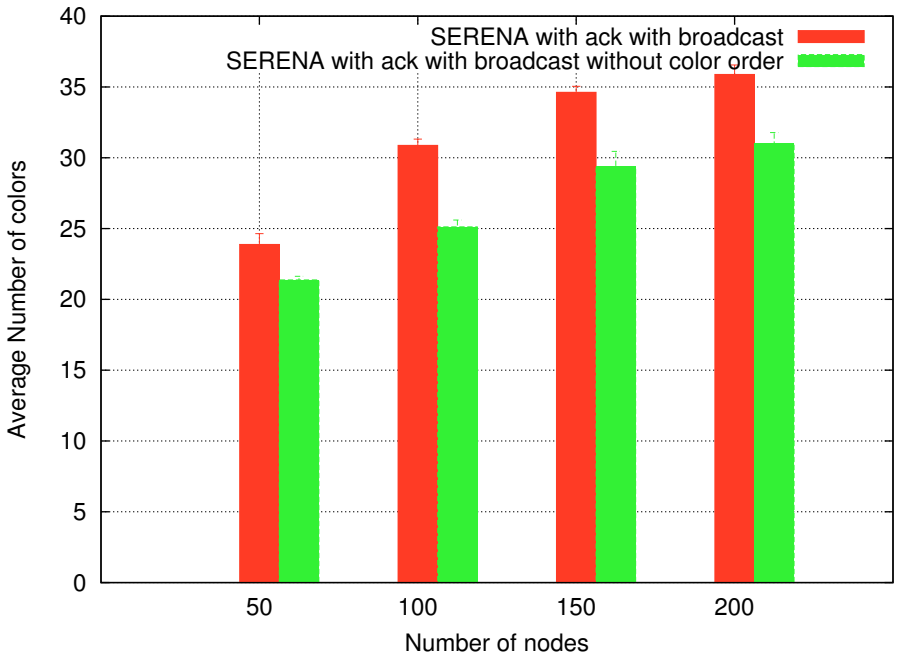


Figure 3.16: Strategic coloring with and without a color order between the node and its parent.

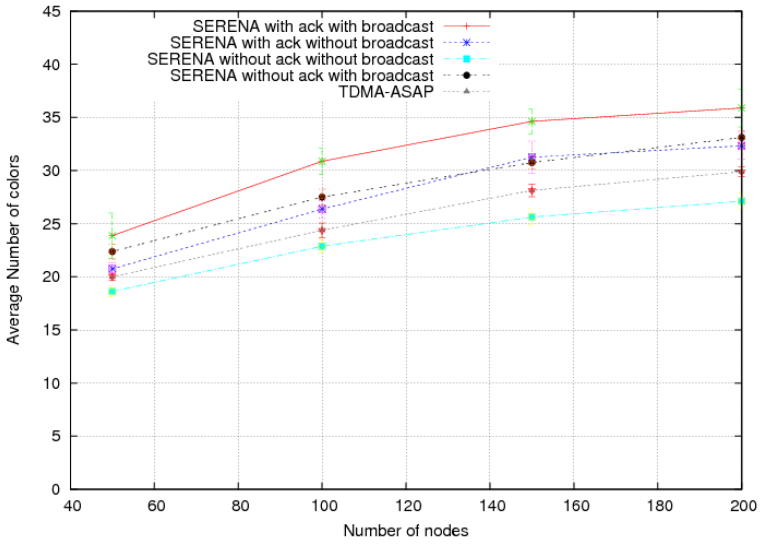


Figure 3.17: Comparison of the number of colors used in SERENA and TDMA-ASAP.

From Figure 3.17, we notice that SERENA without immediate acknowledgement and without broadcast outperforms TDMA-ASAP in terms of number of colors. However, in these two algorithms, any node has the same conflicting nodes. This proves that the heuristic used by SERENA is more appropriate. Indeed, let us consider a tree where the root has n

children. Each child is a root of a subtree. Since any two children cannot have the same color, each one will have a different color. Let u be the child with the largest number of descendants. If u is the last node to be colored, it will be assigned a color at least equal to n. And since the color of any node is higher than the color of its parent, descendants of u will have colors higher than n. Moreover, the biggest subtree is likely to use the highest number of colors, so there will be colors used only in this subtree. Consequently, the total number of colors will increase. To avoid this, SERENA starts by coloring the node u.
TDMA-ASAP uses a number of colors smaller than the two variants of SERENA with immediate acknowledgement. That is because TDMA-ASAP does not support the immediate acknowledgement.
From Figure 3.17, we notice that the difference between the 4 SERENA variants is more visible for large topologies. As expected, the support of both the immediate acknowledgement and the broadcast is the most color consuming variant because it implies the highest number of conflicts per node. For instance, for density=10 and 100 nodes, any node has in average 33.3 conflicting neighbors in the variant '*SERENA with ack with broadcast*', 25.3 in the variant '*SERENA with ack without broadcast*' and 21.2 in the variant '*SERENA without ack without broadcast*'.

We also compare the average number of rounds required to color the network in the aforementioned variants of SERENA. Figure 3.18 shows that adding a new requirement produces an additional number of rounds. For example, SERENA supporting the immediate acknowledgement and the broadcast requires 83.7 rounds, while '*SERENA without ack without broadcast*' requires 54.5 rounds. The variants supporting either the immediate acknowledgement or the broadcast requires almost the same number of rounds. Notice that this overhead is paid only when the coloring is performed.

3.4.4 Adaptivity of SERENA to Node/Link Failure in Data Gathering Applications

3.4.4.1 Problem and Methodology

In addition to the limited energy of a sensor node, wireless sensor networks are prone to interferences, message losses, and node/link failures. In particular, in a data gathering tree, when any node can no longer reach its parent because the parent or the link with this parent has failed, the tree must be repaired. Any solution should address the following issues:

1. How to tolerate a maximum number of failures,
2. How to reduce the overhead induced by the tree repair,
3. How to minimize the delay needed to collect data through the repaired tree.

Upon neighborhood changes, routing protocols manage to repair routes to ensure data delivery. The issue is that **color conflict** may occur. Indeed, after any node u replaces its

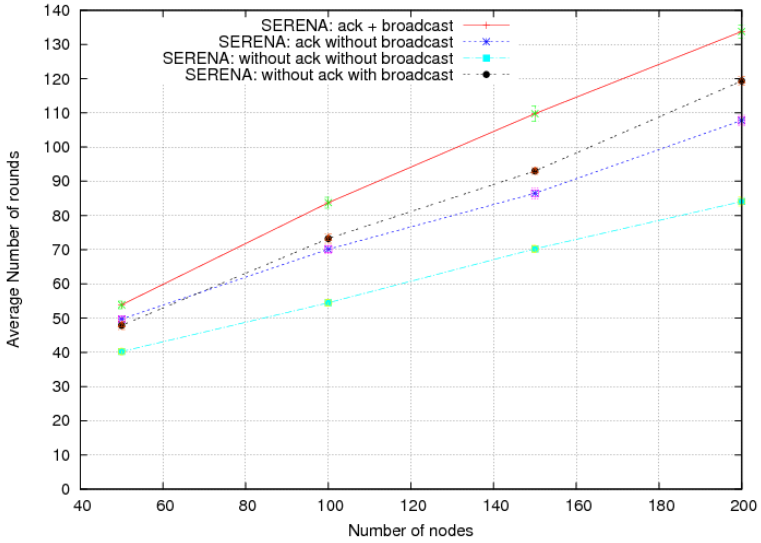


Figure 3.18: Comparison of the number of rounds used in different variants of SERENA.

unreachable parent by a 1-hop neighbor, among its conflicting neighbors, **Sets 1, 3, 4** remain unchanged, while **Sets 2 and 5** are updated (see the definition of these sets in Section 3.4.3.2). Notice also that even if the coloring remains valid, it may be no longer optimized. Indeed, *Rule R2.2* may be no longer verified (see Section 3.4.1.2), hence data gathering delays are increased. To cope with this issue, the trivial solution is to recolor the network. However, the drawback is the induced overhead.

Recoloring the network in case of color conflicts is a potential solution that costs the coloring overhead and the delay between color conflict detection and recoloring. Our aim is to avoid this overhead by proactively determining potential substitute(s) of the parent in the data gathering tree, called **parent backup**, and integrating it in the coloring. In other words, we color the network such that when the parent is replaced by a predetermined backup, no color conflict occurs and the initial coloring remains valid.

This solution can be integrated with a routing protocol based on redundant paths. In fact, most of the existing multipath routing protocols were mainly developed to provide fault tolerance at the network layer. In particular, mobile ad hoc networks where the nodes mobility results in many topology changes require fault-tolerant routing protocols. In this category we can cite for instance DBR2P [109] and AODV-BR [110]. For WSNs we can cite Braided Multipath Routing Protocol [111].

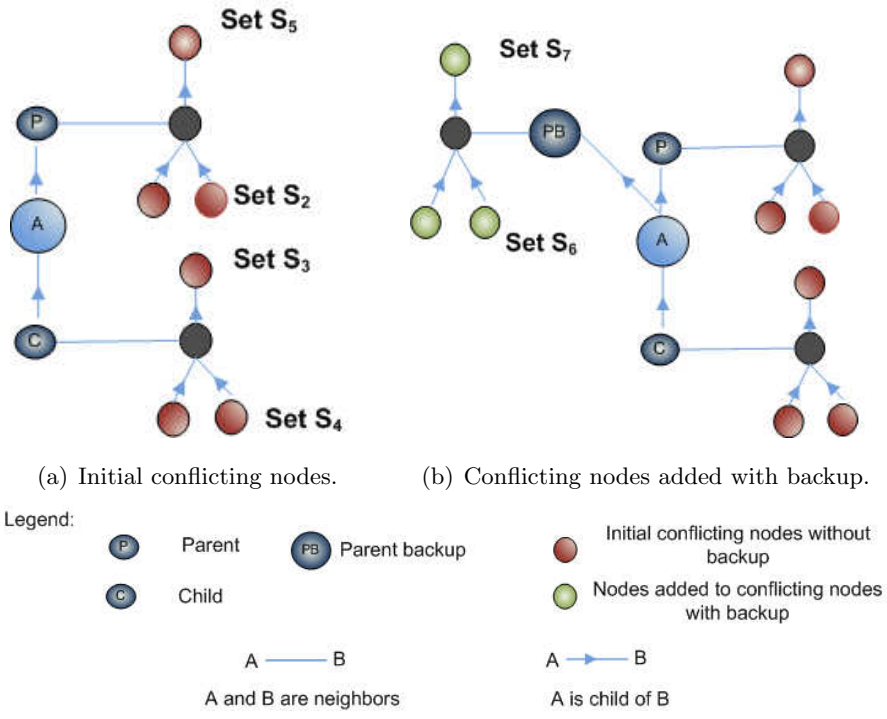
We propose two solutions and compare them with SERENA. The benefit of these solutions is the enhancement of the reliability and the efficiency of wireless communications. We will show by simulation that the cost is the data delivery delays and the energy consumption.

3.4.4.2 First Solution(*Sol1*): Tree coloring with one backup per parent

Our first solution tolerates a single failure per parent, this failure can be the parent failure or the failure of the link with this parent. Any node u has a single parent backup. This parent backup is a 1-hop neighbor that belongs to a smaller tree level than the node considered, assuming the root level is 0. This parent backup is denoted *uncle*. If the node that is unable to reach its parent is unable to find an uncle, it will select a *brother*, that is a 1-hop neighbor that belongs to the same tree level and has an identifier smaller than u. Notice that with these constraints, any node has 0 or 1 backup and moreover, the tree repaired by the choice of either the *uncle* or *brother* does not induce routing loops. As illustrated in Figure 3.19, the set of conflicting nodes must contain **Set 1, Set 2, Set 3, Set 4** and **Set 5** and also:

- **Set 6**: the children of the 1-hop neighbors of the parent backup of u.
- **Set 7**: the parent of the 1-hop neighbors of the parent backup.

We apply *Rule R2.2* and maintain an order between the color of a node and the color of its parent. However, there is no order between the color of the node and the color of its parent backup. Hence, the new coloring may be not delay optimized.



(a) Initial conflicting nodes. (b) Conflicting nodes added with backup.

Figure 3.19: The set of conflicting nodes of the node A.

3.4.4.3 Second Solution(*Sol2*): Tree coloring with several backups per parent

The second solution consists in 3-hop coloring where the set of conflicting nodes of each node contains the nodes up to 3-hop. Consequently, a node has as potential parent backup all its 1-hop neighbors. The Rule *R2.2* is taken into account with the initial parent. However, there is no imposed order between the color of the node and the color of any of its parent backups. Notice that applying this solution, routing loops are possible. However, we address this issue by one of the two following hints:

- We assume that the routing protocol handles the loops with detection and resolution techniques.
- Or, when a node chooses a parent backup, it chooses only a node that has a smaller identifier.

3.4.4.4 Comparative Study

In this section, we present a table comparing the aforementioned two solutions with the basic solution SERENA. We then present simulation results.

1. **The trade-off reliability, data delivery delays and energy consumption**
   Table 3.3 compares the performances of the aforementioned solutions denoted respectively *Sol1* and *Sol2*. For *Sol1* two variants are evaluated:
   - **With order:** where the color of a node is greater than the color of its parent backup.
   - **Without order:** when this order is not required.

   Notice that for *Sol2*, it is impossible to define an order between the color of a node and its parent backups which are in fact its 1-hop neighbors. So, *Sol2* implies automatically that no color order is imposed with regard to the backups.
   From Table 3.3, we can draw the following conclusions.
   (a) *Sol2* guarantees the highest reliability whereas *Sol1* tolerates only one failure per parent. Moreover, as we will see, the number of colors is the highest because all 3-hop neighbors of any node cannot share its color.
   (b) The integration of *Rule R2.2* allows to have optimal end-to-end delays as long as the link with the parent is operational. Otherwise, each time there is an inversion between the color of a node and the color of its new parent, an additional cycle is required, since in this case the child accesses the medium after its new parent.
   (c) Concerning the overhead, *Sol2* induces higher storage overhead because a node stores information about all its neighbors up to 3 hops. This results also in a higher number of messages exchanged as we will show through the simulation results.

   To conclude, we notice that the preference of a solution to another is a trade-off between the tree reliability on the one hand and the delays and energy efficiency on the other hand.

Table 3.3: Comparison of coloring with/without backup

		***Sol1*: One backup**		***Sol2*: many backups**	
	SERENA	**without order**	**with order**	**without order**	**with order**
number of failure tolerated in the tree	0	1 failure per parent		as much as 1hop neighbors	impossible
coloring of the repaired tree	tree should be recolored	still valid		still valid	impossible
delay	optimal	no longer optimal: one additional cycle per inversion in the color order between the child and its new parent	optimal	no longer optimal: one additional cycle per inversion in the color order between the child and its new parent	impossible
overhead	optimal	light: a node stores the parent backup information and replaces the parent when needed		relatively important: the node stores information about all its 3-hop neighbors and changes the parent when needed.	impossible

2. **Simulation results**

(a) **Simulation setup:** We use the same Java simulator used in Section 3.4.3.3, and vary the same parameters of the generated graphs: the number of nodes and the density. We build a tree of minimum path length to the sink. For a given number of nodes and density value, we generate 20 random topologies. Hence, each result is the average of 20 simulation runs.

(b) **The average number of nodes that fail to have a backup:** In this section, we focus on *Sol*1, where any node has 0 or 1 backup. Figure 3.20 illustrates the average number of nodes that fail to have a parent backup for different configurations. This number is small, it ranges from 1.8 to 10.6 for 25 and 81 nodes respectively. At worst, almost 14

among 81 nodes with density=8 fail to have a parent backup. As expected, the number of nodes without backup is inversely proportional to the density and does not depend on the size of the network. That is because the backup is chosen among the neighbors and since we increase the number of nodes while increasing the area size, adding nodes does not mean increasing the number of neighbors. Meanwhile, increasing the density allows nodes to have more neighbors and then the probability to have a backup is higher. Figure 3.20 witnesses the fact that *Sol*1 is suitable for high density networks.

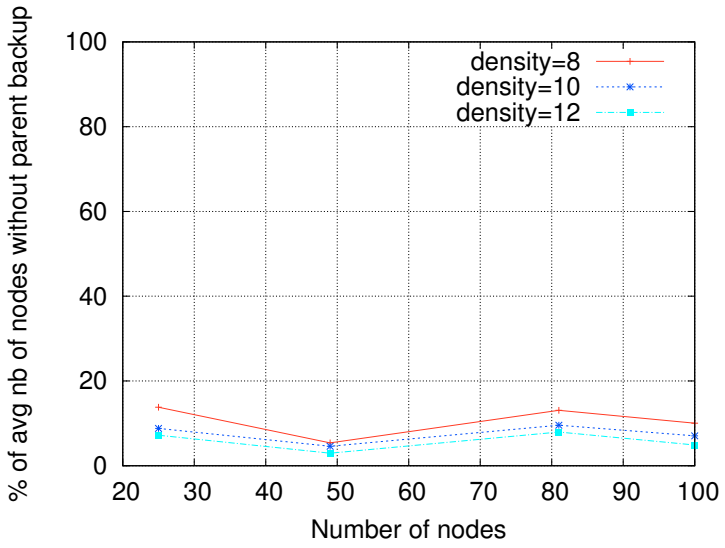


Figure 3.20: Percentage of the nodes without parent backup.

(c) **Comparison of the three solutions:** In this section, the density of nodes is set to 10. Regarding an algorithm of node activity scheduling, the total number of colors represents the activity duration in the network. The smaller this number is the more energy efficient is the solution, and the smaller the end-to-end delays are. That is why, the number of colors is the main performance criteria.

As expected, Figure 3.21 shows that *Sol2* induces the highest number of colors. Consequently, *Sol*2 ensures the reliability of wireless communications at the detriment of both data gathering delays and energy consumption.

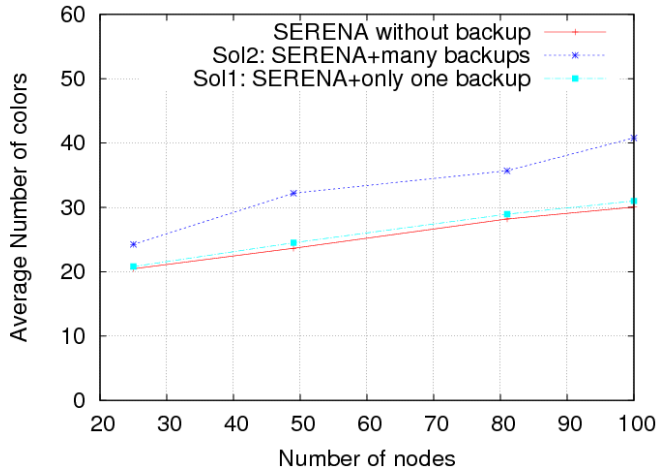


Figure 3.21: The number of colors used by *Sol1* and *Sol2*.

Also, as illustrated in Figures 3.22 and 3.23, *Sol*2 requires the highest number of rounds and sent messages. This is explained by the fact that to color itself, any node collects information about nodes that are at most 3 hops away from it. The propagation of this information adds a time overhead for the solution and requires more messages.

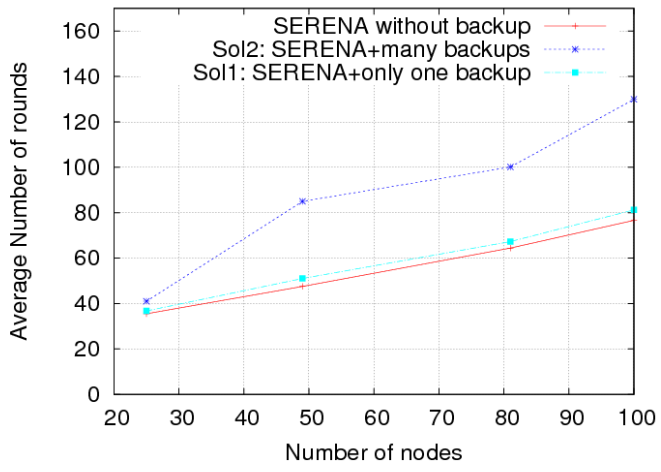


Figure 3.22: Comparison of the number of rounds produced by *Sol*1 and *Sol*2.

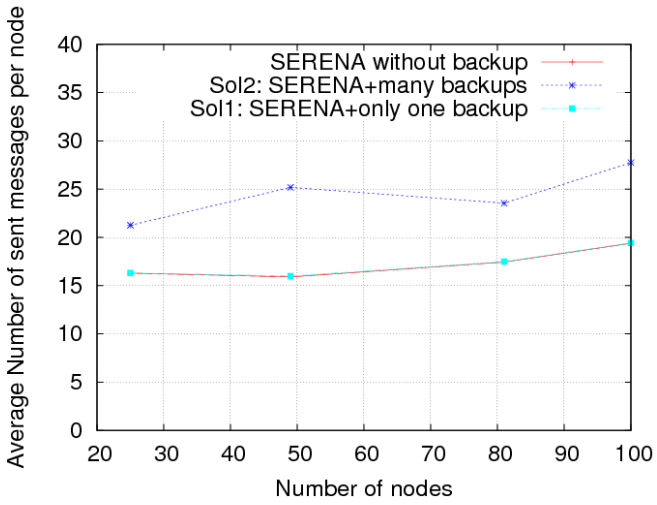


Figure 3.23: Comparison of the number of sent messages used by $Sol1$ and $Sol2$.

Concerning $Sol1$, it generates almost the same number of colors and rounds as SERENA. This is because in fact, this solution does not add a high number of conflicting nodes (nodes depicted in Figure 3.19(b)). For instance, the average number of additional conflicting nodes in SERENA+one backup compared to SERENA is 1 for 49 nodes with density 8. Moreover, these nodes are a subset of the 3-hop neighbors, that is why $Sol1$ produces a smaller number of colors than $Sol2$.

3.5 Conclusion

In this chapter, we formulated the coloring problem for the node activity scheduling in wireless ad hoc and sensor networks. This definition is generalized since it has as a parameter the minimum distance between two interfering nodes. One of the results described in this chapter is the proof of the NP-completeness of this problem. We proved that it is an efficient way to schedule the nodes activity in these networks as it has many benefits especially at the bandwidth and energy levels. We also discussed the issues of the applicability of the coloring to real wireless environment. Our preliminary conclusion is that in order to have good performance of the coloring algorithm, it should be aware of the characteristics of the network and the application and to be able to adapt to them. This conclusion is confirmed by simulation results of the adaptivity of SERENA to data gathering applications and to unreliable wireless communications. In the next chapter, we will focus on another requirement which is the support of dense networks. One of the difficulties that a coloring algorithm faces in such environment, is the limited storage capacity of sensor nodes. We will see in the next chapter how to address this problem.

Chapter 4

OSERENA: Optimized SchEduling RoutEr Node Activity in Dense Wireless Networks

Introduction

In the previous chapter, we presented SERENA and its optimization. In fact, this protocol was designed to be integrated in a real platform of sensors, with MAC, network and application protocols. However, to achieve this integration, we faced a practical problem: the memory size of sensors is very limited and make the scaling a real challenge. This challenge is more present in dense networks, where any node has many neighbors. Examples of dense WSNs are given by smart dust [112] where microelectomechanical systems called MEMS can measure temperature, vibration or luminosity.

This is the motivation behind the design of OSERENA, a new distributed 3-hop coloring algorithm for general applications in dense wireless ad hoc and sensor networks. Compared to SERENA, OSERENA considers a new message format and new processing rules. The maximum size of this message is a constant and depends neither on the density nor on the size of the network. Moreover, OSERENA does not require any node to store its 2-hop neighbors. Hence, bandwidth is spared and the scaling is possible. Further, OSERENA produces the same number of colors as SERENA and usually the same number of rounds.

This chapter contains 2 parts:

1. *First part: Presentation of OSERENA.* In Section 4.1, we present the principles that OSERENA is based on to reduce the size of messages. Section 4.2 specifies OSERENA. In Section 4.3, we present the properties of OSERENA and prove that it is equivalent to the centralized algorithm First Fit. In Section 4.4, we study the performance of OSERENA and compare it to SERENA.

2. *Second part: Application to the OCARI project.* We describe in Sections 4.5.2 and 4.5.3

our contribution to the OCARI project which consists in OSERENA and the routing protocol EOLSR [30] and their integration in a testbed of sensors. In Section 4.5.5, we give some results of OCARI.

4.1 Optimization Principles

As explained in Chapter 3, nodes in SERENA periodically exchange *Color* messages to color themselves. This message includes the address, the priority and the color of 1) the node u itself, 2) its 1-hop neighbors in $\mathcal{N}(u)$ (the set of its conflicting neighbors), as well as 3) its 2-hop neighbors in $\mathcal{N}(u)$. The data locally maintained by any wireless sensor include these data as well as the priority and color of any neighbor up to 3-hop. Let us assume that the 2-hop neighbors of a node are the nodes in a disk of radius $2R$ where R is the transmission range. Consequently, the average number of nodes in the neighborhood up to 2 hops is equal to $4 \cdot density$, where $density$ stands for the average number of nodes in the disk of radius R (that is the average number of neighbors). Such an overhead can be unacceptable for wireless sensors with limited storage and processing capabilities as well as low residual energy. That is why, the key idea of OSERENA is to avoid the need to store the 2-hop neighbors. This is achieved thanks to the following observations.

- It is necessary that any node u knows the highest priority taken by its uncolored neighbors up to 3-hop in order to be able to color itself. Furthermore, in 3-hop coloring, any node u must send information concerning itself, its 1-hop and 2-hop neighbors to let its 1-hop neighbors know information about their 1-hop, 2-hop and 3-hop neighbors. Hence, node u must send its priority, the highest priority taken by its uncolored 1-hop neighbors as well as the highest priority taken by its uncolored 2-hop neighbors. We will see in Section 4.3.3 how to compute the near optimal number of these priorities.

- Similarly for the choice of the color, node u must know the colors already used in its neighborhood up to 3 hops. However, it does not matter for u to know which node up to 3 hops has which color, but only which colors are taken by 1-hop, 2-hop and 3-hop neighbors respectively. That is why, we use the fields $bitmap1$, $bitmap2$ and $bitmap3$ for the bitmaps of colors used at 1-hop, 2-hop and 3-hop neighborhood respectively.

In the next section, we will see how OSERENA takes advantage from these remarks to optimize the size of the data stored and exchanged.

4.2 Presentation of OSERENA

4.2.1 Assumptions

We assume that we have **an ideal wireless environment** (see Definition 8 in Chapter 3). This assumption is usually made by all coloring algorithms when applied to wireless networks.

Although we assume an ideal environment, we show that OSERENA tolerates message losses (see Rule R6 in Section 4.2.3 and Property 2 in Section 4.3.1).

Definition 10. *A 3-hop node coloring is said* ***valid*** *if and only if no two 1-hop, 2-hop or 3-hop nodes have the same color. The smaller the number of colors obtained, the better the coloring algorithm.*

4.2.2 The Color Message

Figure 4.1 depicts the Color message used by OSERENA.

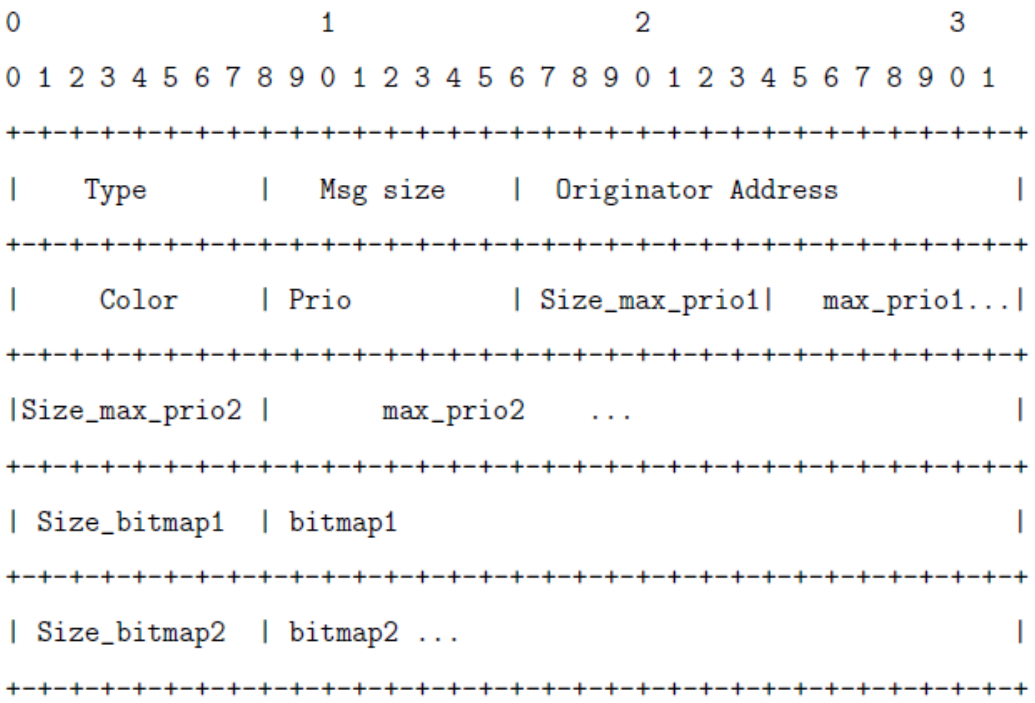


Figure 4.1: Format of the Color message in OSERENA.

We now describe this message format.

• Each node has a priority that determines its order in the coloring. It is composed of two values: the most important one is the integer *prio*, and the other one breaks ties, it is the address of the node. For two nodes that have the same value of *prio*, the node with the highest priority is the node with the smallest address.

• For any set X, we define the function $max_i\{X\}$, $i \geq 2$, that provides the i highest values in X:

$$max_i\{X\} = \left\{ \begin{array}{ll} \emptyset, & \text{if } X = \emptyset; \\ X, & \text{if X contains less than } i \text{ elements;} \\ \{max\{X\}, max_{i-1}\{X \setminus max\{X\}\}\}, & \text{otherwise} \end{array} \right\}$$

If X is a set of priorities, we have the following property for $i = 2$. Let $max_2\{X\} = (X_1, X_2)$, where $X_1 = (p_1, a_1)$ and $X_2 = (p_2, a_2)$ then we have,

- $p_1 > p_i \ \forall i \neq 1$, and if $\exists\ X_j = (p_j, a_j)$ such that $p_j = p_1$ then we have $a_1 < a_j$.
- $p_2 > p_j \ \forall j > 2$, and if $\exists\ X_k = (p_k, a_k)$ such that $p_k = p_2$ then we have: $a_2 < a_k$.

• We define the field $max\_prio1(u)$ as: $max\_prio1(u) = max_{4\ v\ uncolored \in 1hop(u)}\ priority(v)$, where the set $1hop(u)$ denotes the 1-hop neighbors of the node u. In other words, $max\_prio1(u)$ is equal to:

- the four highest priorities of the uncolored 1-hop neighbors of u, if four such nodes exist;
- the priority of the only three (respectively two, respectively one) uncolored 1-hop neighbors, if only three (respectively two, respectively one) such nodes exist;
- empty, denoted $\varnothing$, if none exists.

Notice that $max\_prio1(u)$ can only contain nodes that are 1 hop away from u.

• Similarly, we define $max\_prio2(u) = max_{3\ v\ uncolored \in 1hop(u)}\ max\_prio1(v)$.
$max\_prio2(u)$ contains the three highest priorities of the uncolored 1-hop neighbors of the 1-hop neighbors of u, if they exist.
Notice that $max\_prio2(u)$ may contain the node u itself, 1 or 2-hop neighbors of u.

• The variable $max\_prio3(u)$ is defined as the highest priority of the uncolored 1-hop neighbors of the 1-hop neighbors of the 1-hop neighbors of u.
$max\_prio3(u) = max_{\ v \in 1hop(u)}\ max\_prio2(v)$.
Notice that $max\_prio3(u)$ is locally maintained and may contain the node u itself, 1, 2 or 3-hop neighbors of u.

• The computation of $max\_prio1(u)$, $max\_prio2(u)$ and $max\_prio3(u)$ is done from the *Color* messages received during each round. The values computed for $max\_prio1(u)$ and $max\_prio2(u)$ are inserted in the *Color* message sent by the node u.

• It follows that the *Color* message sent by any node u contains $priority(u)$, $max\_prio1(u)$ and $max\_prio2(u)$, as well as the color of u.

• To further reduce the *Color* message size, three bitmaps of colors are used. $bitmap1(u)$ (respectively $bitmap2(u)$) denotes the colors used by 1-hop neighbors (respectively by 2-hop neighbors) of u.

4.2.3 Rules of OSERENA

In OSERENA, any node u proceeds as follows to color itself:
1. Node u characterizes the set $\mathcal{N}(u)$ of nodes that cannot have the same color as itself. In 3-hop coloring, this set includes the neighbors up to 3 hops.
2. Node u computes its priority, denoted $priority(u)$. This priority consists of two components: the most important one is denoted $prio(u)$. From simulation feedback, we have noticed that the assignment of $prio(u)$=number of neighbors up to 2 hops outperforms the assignment $prio(u)$=number of neighbors up to 3 hops from u, or a random assignment. However, as OSERENA avoids the expensive computation of the list of neighbors up to 2 hops, OSERENA defines for any node u, $prio(u)$ as the number of its neighbors + the sum of the number of 1-hop neighbors of its 1-hop neighbors. This computation is done during the initialization phase.
3. Concerning **color selection**, OSERENA applies the following rules.

Rule R'1: Any node u colors itself if and only if:

$$priority(u) = max\left\{max\_prio1(u), max\_prio2(u), max\_prio3(u)\right\}. \tag{4.1}$$

Rule R'2: When a node u selects a color, it selects the smallest color unused in $bitmap1(u) \cup bitmap2(u) \cup bitmap3(u)$.

Notice that in addition, this color should not be used by heard nodes (nodes with which there is no symmetric link). Indeed, any node stores information about the heard neighbors and does not select the same color as them in order to avoid color conflicts.

The aim of rules R3 and R4 is **to improve convergence time**. The *Color* message does not contain the whole list of colored 2-hop and 3-hop neighbors. However, any node can deduce the recently colored nodes and stores them in a local data structure denoted *implicit_node_colored_list* whose size is equal to *implicit_node_colored_size*. As we will see later, the storage of this list accelerates the coloring convergence. To build this list, any node u proceeds as follows:

Rule R3: When a node u receives the *Color* message from any neighbor node v, it compares the current value of $max\_prio1(v)$ (respectively $max\_prio2(v)$) with the previous one sent by v, denoted $previous\_max\_prio1(v)$ (respectively $previous\_max\_prio2(v)$). Any priority of $previous\_max\_prio1(v)$ (respectively $previous\_max\_prio2(v)$) higher than the highest value of $max\_prio1(v)$ (respectively $max\_prio2(v)$) corresponds to a recently colored node. This node is then inserted in the list *implicit_node_colored_list*.

The motivation behind the integration of Rule R3 is that naturally and because of the different distances between any two nodes, nodes are not informed about the coloring of their

conflicting neighbors at the same time. So, among the information they receive, they should consider the fresher information. The scenario in Figure 4.2 illustrates this.

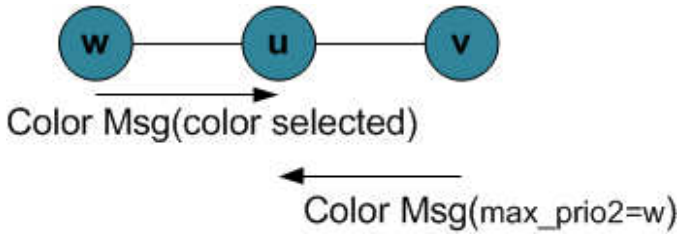


Figure 4.2: An example illustrating the role of the list *implicit_node_colored_list*.

Let assume that the node w is colored. It broadcasts a message to its 1-hop neighbors with the selected color. Meanwhile, node v not informed yet, sends a message to u where the field max_prio2 contains the node w despite w is already colored. Consequently, if u considers the information received from v as the most recent one and transmits it to its neighbors, coloring will be delayed. Indeed, the node to be colored after w, will suppose this latter is uncolored so does not color itself in the current round. Thus, u should discard the information received from v and considers that w is colored (either directly if they are 1-hop neighbors, or via the use of the *implicit_node_colored_list* if they are 2-hop or 3-hop neighbors) like in the example. That is why, OSERENA uses Rule R3 to allow nodes to store some colored neighbors.
In addition, as we have seen in the example of Figure 4.2, the node u discards the information of the node v by removing the node w from the list max_prio2 sent by v. However, not any colored node can be discarded from max_prio1 or max_prio2. To illustrate this, consider the scenario in Figure 4.3.

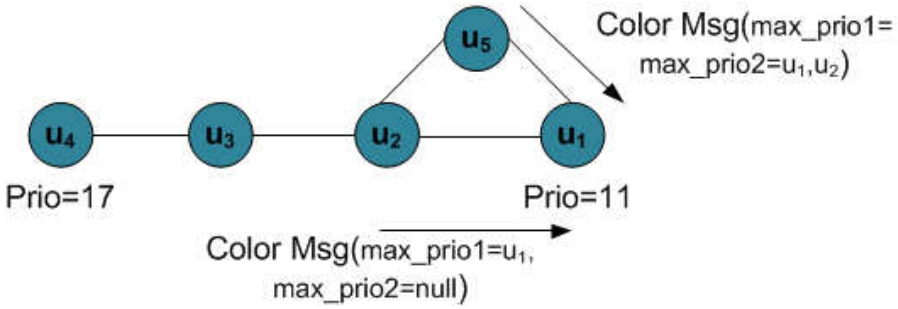


Figure 4.3: An example illustrating the role of rule R4.

In Figure 4.3, all nodes are colored except u_1 and u_4 that have the priorities 11 and 17 respectively. Node u_2 is coloring itself at the current iteration. The node u_1 receives two *Color* messages from u_2 and u_5 containing the fields max_prio1 and max_prio2 depicted in the figure. Consequently, u_1 is aware about the color of u_2 and discards it from max_prio1 and max_prio2 of u_5. The problem is that u_1 will conclude that it is its turn to color itself,

however, it is the turn of its 3-hop neighbor u_4. This results in the violation of the coloring order, and may cause color conflicts. In reality, nodes u_2 and u_5 would inform u_1 about the existence of the node u_4 that has a higher priority if max_prio1 and max_prio2 could contain more nodes. For this reason we adopt Rule R4. A node can be discarded from max_prio1 or max_prio2 if there is still enough information to make right conclusions about colored nodes.
Rule R4: When a node computes max_prio1, max_prio2 and max_prio3 from the values received in the *Color* messages, it proceeds as follows:

- in the computation of max_prio1, it discards any priority value corresponding to an already colored node (that is a node that belongs to the list *implicit_node_colored_list*).
- in the computation of max_prio2, it discards for any sender v, any priority value p corresponding to an already colored node received in $max\_prio1(v)$ if and only if:
  1. either p is the highest priority in $max\_prio1(v)$.
  2. or p is the second highest priority in $max\_prio1(v)$ and the third or fourth highest priority in $max\_prio1(v)$ is equal to $\varnothing$.
  3. or p is the third highest priority in $max\_prio1(v)$ and the fourth highest priority in $max\_prio1(v)$ is equal to $\varnothing$.
- in the computation of max_prio3, it discards for any neighbor v, any priority value corresponding to an already colored node received in $max\_prio2(v)$ if and only if it is the highest or the second highest priority in $max\_prio2(v)$.

Rule R5 is related to **the termination of the coloring algorithm**.

Rule R5: Any node u stops sending its *Color* message as soon as it is colored, $max\_prio1(u) = \varnothing$ and it has received from all its 1-hop neighbors v a *Color* message with $max\_prio1(v) = max\_prio2(v) = \varnothing$.

Rule R6 has been introduced **to tolerate message losses and link failures**.

Rule R6: If at a round $r > 1$ of the coloring algorithm, any node u that does not receive a message from its 1-hop neighbor v, uses the information received from v at round $r-1$. After n successive rounds, with $n \geq 2$ without receiving a *Color* message from v, v is no longer considered as a 1-hop neighbor of node u.

4.3 The properties of OSERENA

In this section, we will present the properties of OSERENA regarding: its correctness, the equivalence to First Fit, the overhead and the convergence time. To study these properties,

we present some Lemmas that are proved in Appendix A.

4.3.1 Correctness of OSERENA Coloring

In this section we prove that in an ideal wireless environment, OSERENA provides a valid 3-hop node coloring avoiding collisions. Furthermore, we prove that this algorithm ends when all nodes are colored.

Lemma 7. *With OSERENA, any node u colors itself if and only if it has the highest priority among all the uncolored nodes in $\mathcal{N}(u)$.*

Lemma 8. *With OSERENA, when node u colors itself, it knows all the colors taken in $\mathcal{N}(u)$ with a higher priority.*

Lemma 9. *OSERENA coloring ends when all nodes are colored.*

Lemma 10. *In an ideal wireless network and in the absence of message loss and node failure, all nodes color themselves with OSERENA and stop sending their Color message.*

Property 1. *OSERENA provides a valid 3-hop node coloring in any ideal wireless environment.*

Proof: For 3-hop coloring, for any node u, the set $\mathcal{N}(u)$ contains by definition all nodes up to 3-hop from u, assuming an ideal environment. From Lemma 7 (see Appendix A for the proof), with 3-hop coloring, any node u can color itself if and only if no uncolored node in $\mathcal{N}(u)$ has a priority higher than u.
According to rule R'1, the priority of u meets Equation 4.1. Moreover, since no two nodes have the same priority, we cannot have a simultaneous coloring of two nodes up to 3-hop away each other. According to Lemma 8 (see Appendix A), when coloring itself, any node u knows all the colors taken by nodes in its $\mathcal{N}(u)$, so it selects the smallest color according to rule R'2. Consequently, assuming an ideal wireless environment, no 2 nodes within 3-hop neighborhood from each other take the same color. Which means that OSERENA provides a valid coloring. With this coloring, nodes that belong to $\mathcal{N}(u)$ cannot create collisions with data sent by u or an acknowledgement sent to u. ■

Property 2. *A failure to receive a Color message from a 1-hop neighbor induces an additional latency in network coloring and does not compromise the validity of coloring with OSERENA.*

Proof: Deduced from rule R6. ■

4.3.2 Equivalence of OSERENA to a Centralized Algorithm

In this section, we compare the behavior of OSERENA with the well-known centralized First Fit 3-hop node coloring [81]. First Fit coloring is a simple centralized coloring, called also the greedy coloring. With First Fit 3-hop node coloring, nodes are sorted according to their priority and are colored in that order. Any node u receives the smallest unused color in $\mathcal{N}(u)$. In this Section, we prove that the colors granted to nodes by this algorithm and OSERENA are the same.

Lemma 11. *For any node u, for any given priority assignment, nodes* $\in \mathcal{N}(u)$ *color themselves in the same order with OSERENA and First Fit.*

Property 3. *For any topology, OSERENA provides the same coloring as a centralized First Fit 3-hop node coloring algorithm using the same priority assignment.*

Proof: For any topology, for any node u in this topology, the color of u is determined by the colors already used in $\mathcal{N}(u)$ when u colors itself. According to Lemma 11 (see Appendix A), all nodes in $\mathcal{N}(u)$ color themselves in the same order with OSERENA and First Fit. Let u_1 be the first node that colors itself in $\mathcal{N}(u)$. It takes the smallest available color in $\mathcal{N}(u_1)$. Let u_2 be the first node that colors itself in $\mathcal{N}(u_1)$, and so on. After a finite number of iterations (at most equal to the number of nodes in the topology), we get a node u_{k+1} the first node that colors itself in $\mathcal{N}(u_k)$ and colors itself without being constrained by any other node in OSERENA: u_{k+1} has the highest priority in $\mathcal{N}(u_{k+1})$. This node takes the color 0 in OSERENA. With First Fit, since no node in $\mathcal{N}(u_{k+1})$ is already colored, u_{k+1} takes color 0. Nodes in $\mathcal{N}(u_{k+1})$ with a priority higher than or equal to $priority(u_k)$ are colored according to their priority order with OSERENA and First Fit. Consequently, they receive the same colors. We apply the same reasoning to node u_k and nodes in $\mathcal{N}(u_k)$ with a priority higher than or equal to $priority(u_{k-1})$, going back up to node u_1 and finally node u that receives the same color with OSERENA and First Fit, because the same colors are already assigned in $\mathcal{N}(u)$. ■

This property is very important because the colors of the nodes can be easily predicted. Hence, it is easier to check the correctness of the implementation of OSERENA.

4.3.3 Reduced Overhead

Compared to SERENA, OSERENA does not require the need to store the 2-hop neighbors. Further, the size of the message is considerably reduced which saves bandwidth. In this section, we show that the size of the $Color$ message impacts the running time and we determine its optimal size.

4.3.3.1 Message Size

Assuming the near optimal size of max_prio1 and max_prio2 determined later on (see Lemma 12), we can compute the maximum size of the message $Color$.

Property 4. *With the setting in bytes of* $Size\_max\_prio1 = 4$ *and* $Size\_max\_prio2 = 3$*, OSERENA uses a Color message whose size is at most* $4 + 8 \cdot (size\_address + size\_prio) + size\_color + size\_bitmap1 + size\_bitmap2$ *bytes.*

Proof: This is deduced from the *Color* message format, where the 8 factor is the maximum size of $priority + max\_prio1 + max\_prio2$, and $size\_address$, $size\_prio$ and $size\_color$ are respectively the size in bytes of the address and the fields $Prio$ and $Color$. ■

4.3.3.2 Constraints for the Computation of max_prio1 and max_prio2 Sizes

Decreasing the size of max_prio1 and max_prio2 may delay coloring. Hence, the optimal size of max_prio1 and max_prio2 is a trade-off between bandwidth consumption and convergence time of the coloring algorithm.

The simplest solution would be to maintain only one priority for max_prio1 and max_prio2. This would not suffice to color any wireless network with the same number of rounds as SERENA. Indeed, a node v, 2 hops away from node u colored at round r would not know at round $r + 2$ that it has the highest priority. Hence, MORE THAN ONE highest priority at respectively 1-hop and 2-hop must be maintained and sent. Moreover, this solution does not allow to remove already colored nodes in the computation of max_prio1, max_prio2 and max_prio3 which may delay coloring. That is why, several priorities are maintained in max_prio1 and max_prio2. The question is how many? To be able to discard one value corresponding to an already colored node and sent by neighbor v in $max\_prio2(v)$ implies that v sends at least 2 values in $max\_prio2(v)$. To be able to compute the 2 highest values in $max\_prio2(v)$ and discard one value, node u must receive at least 3 values in $max\_prio1(u)$. Hence, the minimum sizes are $Size\_max\_prio1 = 3$ and $Size\_max\_prio2 = 2$.

Unfortunately, we can still exhibit scenarios with this minimum setting, where only the first address in max_prio2 is discarded if already colored, producing a number of rounds higher than SERENA. In simulations, we identified a scenario with 100 nodes uniformly distributed with a density of 20 that need 175 rounds to be colored in OSERENA instead of 134 rounds with SERENA. That is why, we select $Size\_max\_prio1 = 4$ and $Size\_max\_prio2 = 3$.

4.3.3.3 Computation of the Optimal Size of max_prio1 and max_prio2

In this Section, we use the most adopted model in wireless sensor networks: **the unit disk graph model** [113]. Hence:

- Nodes are deployed in the 2-dimensional plane.
- A uniform transmission range R is defined.
- A node receives a transmission from another node, if and only if, its distance is lower than R.

- There are neither message losses nor node failures.

This model is widely employed for the study of ad hoc and sensor networks. Clearly, this model is a simplification of reality since (1) packet losses are possible, and (2) nodes can have different transmission ranges. Indeed, even if nodes are homogeneous, this model does not account for the presence of obstacles which might obstruct the signal propagation. However, this model is simple enough to promote theoretical results. Thus, we still use it to model the wireless ad hoc and sensor networks. Furthermore, in some calculation in this Section and also in Section 4.3.4 (but not the simulations), we also make the following approximation:

Assumption (approximation) A4: Any node at a distance between R and hR is a h-hop neighbor (e.g. a node at distance between R and $2R$ from another node, is assumed to be at 2 hops).

The assumption is valid asymptotically when the density converges towards infinity. For a more detailed exploration of the exact relationship between number of hops and distance, see for instance [114].

Lemma 12. *With the setting* $Size\_max\_prio1 = 4$ *and* $Size\_max\_prio2 = 3$ *and rules R3 and R4, OSERENA colors any node* u *in the same round as SERENA, except when three nodes* 2 *hops away from* u*, but* 4 *hops away from each other are coloring simultaneously just before* u*.*

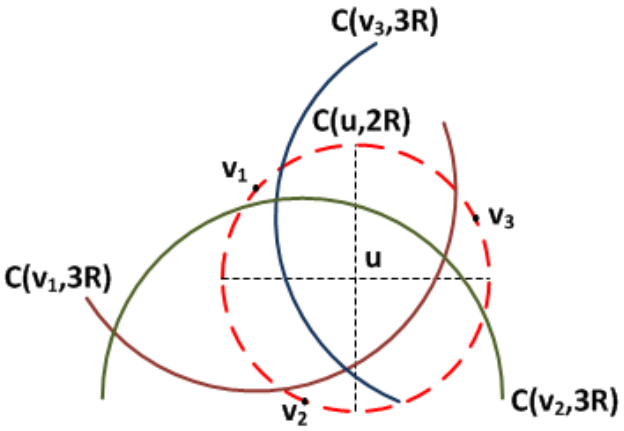


Figure 4.4: A scenario where OSERENA can make more rounds than OSERENA.

Proof: We first identify this scenario illustrated by Figure 4.4 (C(u,r) is the circle centered at node u and of radius r) and then compute its probability in the next section. When **three nodes** two-hop away from u, and 4-hop away from each other are coloring simultaneously just before u, node u is not allowed by rule R3 to discard three of them in the received $max\_prio2(v)$, hence the coloring of node u is delayed. We can show that this scenario is the only one that will delay u coloring. On the one hand, two one-hop neighbors of u are not

allowed to color simultaneously, because they are at most two-hop away. On the other hand, a one-hop and a two-hop neighbor of u are not allowed to color simultaneously, because they are at most three-hop away. It results that the only case of simultaneous colorings in $\mathcal{N}(u)$ involves nodes that are 2 hops away from u and 4 hops away from each other. ■

Lemma 13. *The setting $Size\_max\_prio1 = 5$ and $Size\_max\_prio2 = 4$ provides the same number of rounds as SERENA.*

Proof: With the setting, $Size\_max\_prio1 = 5$ and $Size\_max\_prio2 = 4$, it is no longer possible to have a bad scenario where **four nodes** two-hop away from u, but 4-hop away from each other are coloring simultaneously. We prove it by contradiction. Let u be any node. We assume that the four nodes v_1, v_2, v_3 and v_4 that are 4-hop away from each other and 2-hop away from u are coloring themselves simultaneously. We notice that the distance between these four nodes is maximized when they belong to the circle centered at u and of radius $2R$ and are diametrally opposed. We can compute the distance of two adjacent points denoted v_1 and v_2, we then have $d(v_1, v_2)^2 = d(v_1, u)^2 + d(u, v_2)^2 = 4R^2 + 4R^2 = 8R^2$ (assuming approximation $A4$). Hence $d(v_1, v_2) = 2\sqrt{2}R < 3R$: which means that v_1 and v_2 are 3-hop neighbors which contradicts our assumption. ■

That is why in the following, we take $Size\_max\_prio1 = 4$ and $Size\_max\_prio2 = 3$ leading to a smaller bandwidth use.

4.3.4 Convergence Time

As shown in the previous section, the selected setting of the size of max_prio1 and max_prio2 provides the same number of rounds as SERENA, except when the bad scenario depicted in Figure 4.4 occurs.

Remark 4. *Notice that even in this case, the total number of rounds required by OSERENA can still be equal to the total number of rounds required by SERENA. The occurrence of the bad scenario is a necessary but not sufficient condition to increase the number of rounds with OSERENA.*

To conclude, the scenario where one node u is colored with a delay in OSERENA compared to SERENA happens if the following events occur:

- E_1*:* $\exists$ v_1, v_2 and v_3, three nodes that are 2-hop away from u and 4-hop away from each other.
- E_2*:* these three nodes v_1, v_2 and v_3 have a priority higher than u.
- E_3*:* v_1, v_2 and v_3 are colored simultaneously.

We assume the unit disk graph model including assumption *A4*. We adopt the following notations. Let $d(u, v)$ denote the euclidian distance between nodes u and v. Let P denote

the probability that the bad scenario occurs. We want to estimate an upper bound of this probability. Let P_i denote the probability that the event E_i occurs, with $i \in [1, 3]$. We have:

$$P \leq \prod_{i=1}^{3} P_i.$$

For any node u, let $\mathcal{D}(u, R)$ (respectively $\mathcal{C}(u, R)$) denote the disk (respectively the circle) centered at u of radius R. Let $A \setminus B$ denote the set containing exactly the elements of A but not those of B.
The computation of upper bounds of probabilities P_1 and P_2 is done geometrically. Details of the computation of P_1, P_2 and P_3 are given in Appendix A. Using these results we have the following property:

Property 5. *The probability of occurrence of the bad scenario on a given node u is upper bounded by $\frac{27}{64}(1 - \frac{3}{4M+1}) \cdot (\frac{1}{4} - \frac{\sqrt{4-2\sqrt{3}}}{2\Pi})$, where M is the number of nodes to 1-hop away from u.*

Proof: Since $P \leq \prod_{i=1}^{3} P_i$, we get $P \leq \frac{27}{64}(1 - \frac{3}{4M+1}) \cdot (\frac{1}{4} - \frac{\sqrt{4-2\sqrt{3}}}{2\Pi})$. ■

A numeric evaluation of the bound yields: $P \leq 0.0564 * (1 - \frac{3}{4M+1})$. Notice that $(1 - \frac{3}{4M+1}) \leq 1$, which leads to $P \leq 0.0564$.

4.4 Performance Evaluation by Simulation

We now evaluate the performance of OSERENA by simulation using the C language for various WSNs.

4.4.1 Simulation Modules and Parameters

We consider various wireless network configurations where the number of nodes varies from 50 to 200 and the average number of neighbors per node, called density, varies from 8 to 30. We check the connectivity of all the topologies generated by our random topology generator. Three modules are simulated:

- *The OSERENA Module* in charge of coloring the wireless network.
- *The SERENA Module* is used as a reference for a comparative performance evaluation.
- *The Neighborhood Discovery Module* in charge of detecting the creation of new links, testing their symmetry and detecting their breakdown. This is done by means of periodic exchanges of *Hello* messages. The *Hello* message contains the list of addresses of heard/symmetric nodes. Consequently, OSERENA that performs 3-hop coloring considers the following definitions:

- Two nodes u and v are 1-hop neighbors if there is a symmetric link between them.
- Two nodes u and v are 2-hop neighbors if there is a node w that is a neighbor of both of them.
- Two nodes u and v are 3-hop neighbors if there is a node w that is a 2-hop neighbor of one of them.

We evaluate the number of colors used, the number of rounds needed to color the whole network, the average number of *Color* messages sent per node as well as the average size of these messages. Each result is the average of 10 simulations.

4.4.2 Performance Results of OSERENA

4.4.2.1 Number of Colors

The number of colors depends on network topology. First, we want to evaluate the impact of node density and node number on the number of colors used by OSERENA.

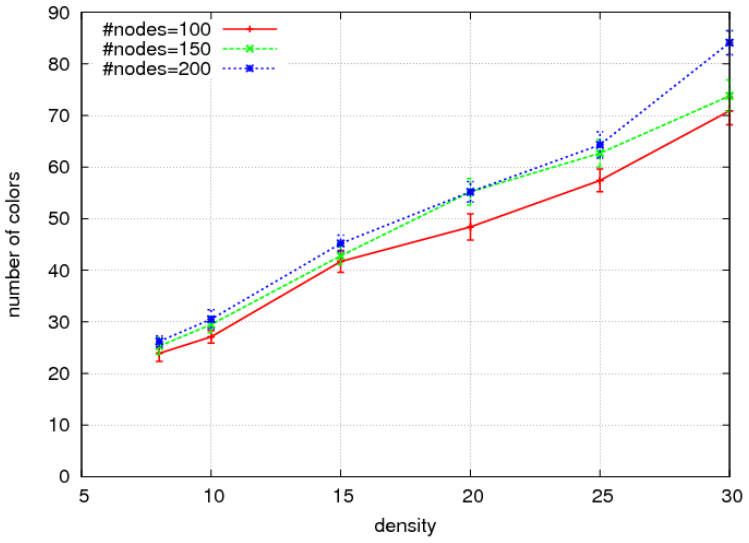


Figure 4.5: Number of colors.

Figure 4.5 shows that the number of colors strongly depends on the density of nodes and much less on the number of nodes. Intuitively, the reason is that the color selected by a node, depends only on its 3-hop neighborhood, hence is related to the number of the 3-hop neighbors (which is itself directly proportional to the density).

Furthermore, the size of the 3-hop neighborhood is not related to the number of nodes in the network, hence this last parameter has less impact.

4.4.2.2 Number of Rounds

To measure the time complexity of OSERENA, we evaluate the number of rounds needed to color the whole network. More precisely, what is the impact of node density and node number on the number of rounds?

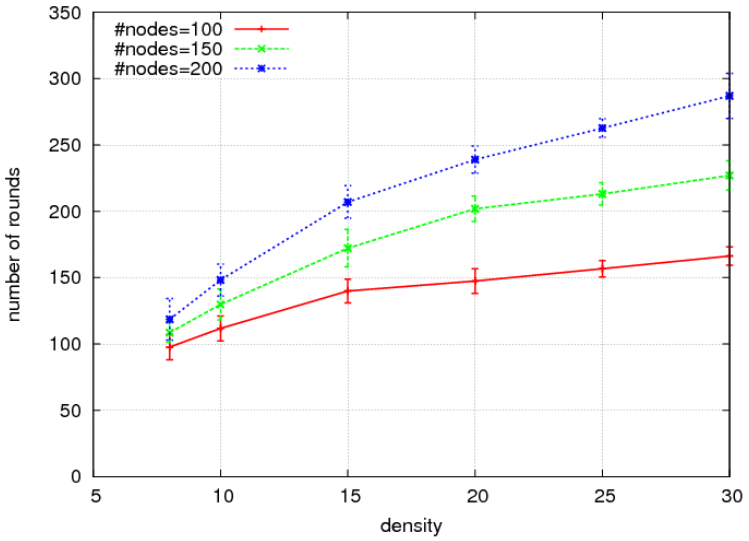


Figure 4.6: Number of rounds.

In Figure 4.6, we observe that the number of rounds depends more on the number of nodes in the network than on density. There is one natural explanation of this observation that the number of nodes has an impact on the number of rounds (and much less on the number of colors, see previous section): in OSERENA, every node u must wait until all the nodes in $\mathcal{N}(u)$ having higher priority than itself color themselves. Recursively, each node in this set should do the same. This is likely to lead to waiting "chains" and such chains are longer in larger networks. This of course contributes to coloring delay.

4.4.2.3 Number of Messages Sent per Node

To compute the overhead induced by OSERENA, we first evaluate the average number of messages sent per node for various network configurations, pointing out the influence of node density and node number. As illustrated in Figure 4.7, the average number of messages is close to the number of rounds (depicted in Figure 4.6). This is expected since every node sends one message per round until a stopping condition is fulfilled (rule R5): in the simulations, for most nodes, most of the time, rule R5 is not verified.

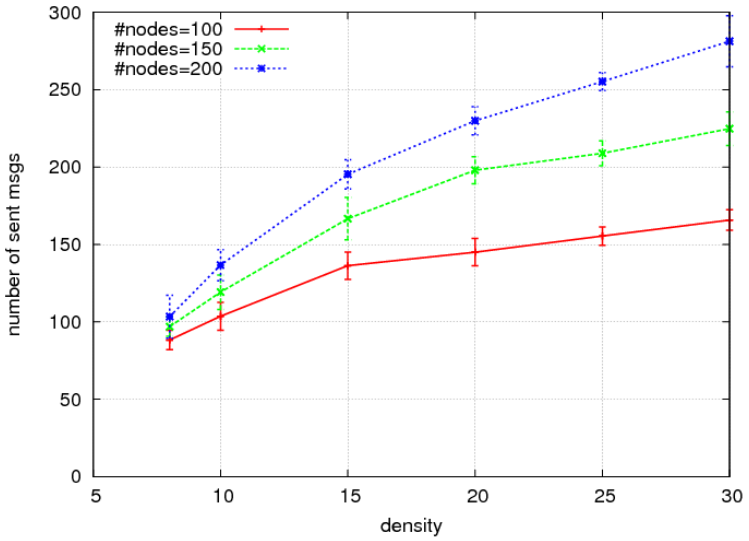


Figure 4.7: Average number of messages sent per node.

4.4.2.4 Number of Bytes Sent per Node

Another expression of the message overhead is given by the average number of bytes sent per node for various network configurations. What is the impact of node density and node number on the number of bytes exchanged during the coloring?

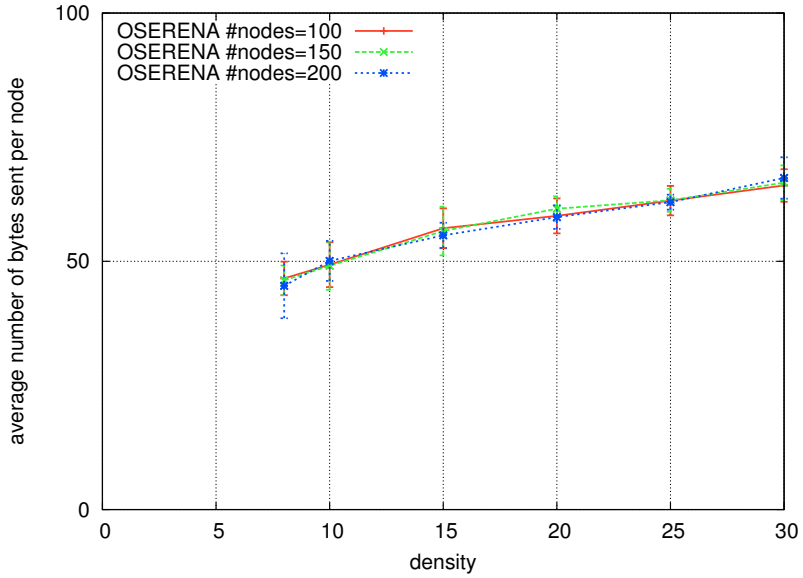


Figure 4.8: Average number of bytes sent per node.

From Figure 4.8, the number of bytes sent per node per round is almost independant

from the number of nodes, whereas density has a limited, but direct impact. This is a direct consequence of the structure of the *Color* message, which includes 2 bitmaps of colors of the 1-hop and 2-hop neighborhood, which increases linearily with density (e.g. 2 additional bits in message, per additional color in the 3-hop neighborhood).

4.4.3 Comparison with SERENA

We now compare the performances obtained by OSERENA with those of SERENA. OSERENA ensures that nodes get the same colors as with SERENA. The open question is at which expense?

4.4.3.1 Number of Colors

Simulation results are compliant with the expected behavior of OSERENA: any node receives the same color with SERENA and OSERENA.

4.4.3.2 Number of Rounds

Simulation results show that OSERENA needs the same number of rounds as SERENA in all the network topologies tested as shown in Figure 4.6 and Figure 4.9.

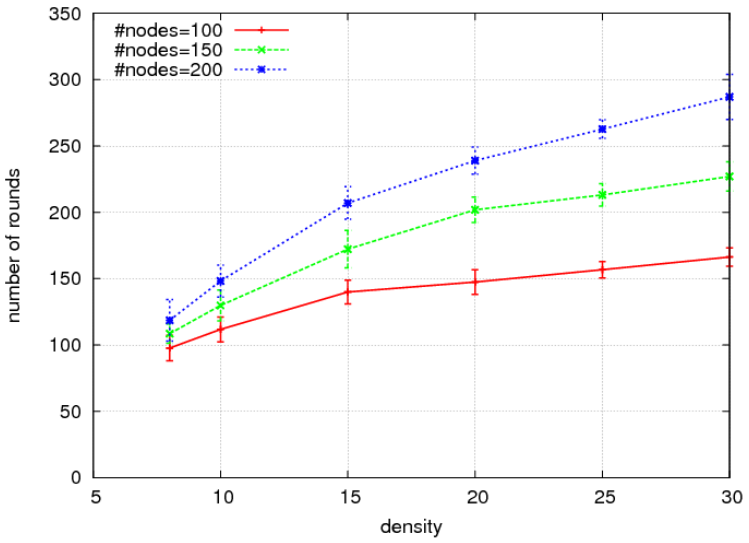


Figure 4.9: Number of rounds for SERENA.

The reason is that the scenario where OSERENA requires more rounds than SERENA on one node has a low occurrence probability (see Section 4.3.4). Moreover, the occurrence of

this scenario on one node does not automatically increase the total number of rounds for the coloring of the whole network.

4.4.3.3 Number of Messages Sent per Node

Simulation results show that the average number of messages sent per node is the same with SERENA and OSERENA for various network configurations.

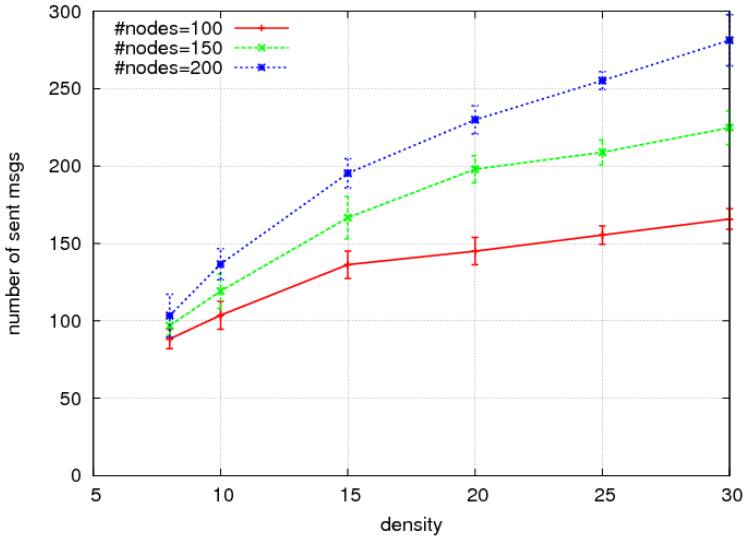


Figure 4.10: Average number of messages sent per node with SERENA and OSERENA.

4.4.3.4 Number of Bytes Sent per Node Per Round

Figure 4.11 depicts the average number of bytes sent per node per round with SERENA and OSERENA. It points out the benefit brought by the optimization of OSERENA: the size of the *Color* messages is much smaller.

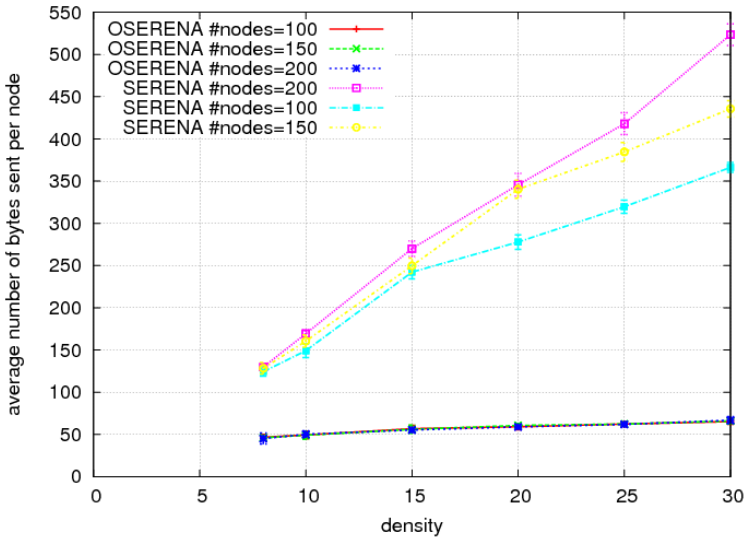


Figure 4.11: Average number of bytes sent per node with SERENA and OSERENA.

In SERENA, a *Color* message includes information for each node in its entire 3-hop neighborhood (address, priority, color): several bytes per node in the 3-hop neighborhood. In OSERENA, only a small fixed subset of priorities and addresses of these nodes are exchanged, and only 2 bits per color are required.

Notice that for wireless sensor networks based on the standard IEEE 802.15.4, the maximum packet size is 127 bytes. Hence SERENA messages are problematic even at the lowest density (and would have probably to be fragmented in several packets). On the contrary, OSERENA fits within this limit until high densities. The proof is that we integrated OSERENA with routing and MAC protocols in a testbed of sensors having 8 K bytes of RAM in the OCARI project. The description of this project is the subject of the remaining of this chapter.

4.5 Application to the OCARI project: Integration of OSERENA and EOLSR

One of the characteristics of my thesis is that it is close to the industrial field. Indeed, an important part of my work is integrated in the OCARI[1] project which provides a communication protocol for industrial WSNs. Hence, our research faces real challenges and requirements. OCARI started in 2006 and when I joined the project in 2009, there were many nice adopted solutions constituting the OCARI protocol stack. However, optimizing these solutions and adding new features with the objective to integrate them in a real industrial network remained

[1] http://en.wikipedia.org/wiki/OCARI

a real challenge. Concerning our contribution in this project, our objective was mainly maximizing the network lifetime and efficiently routing data. Consequently, we integrated the node activity scheduling based on node coloring via OSERENA and energy efficient routing based on EOLSR in a testbed of sensors. In this section, we give an overview about the project and describe some results.

4.5.1 Overview about the OCARI Project

4.5.1.1 Context and Goals

OCARI stands for "Optimization of Communication for Ad hoc Reliable Industrial networks". It is a project made of industrial partners like EDF, DCNS and TELIT and academic partners like LIMOS, LATTIS, LRI and Inria. OCARI project tends to contribute to the emergence of an open and proven standard designed for industrial WSNs. Examples of targeted applications are: monitoring of industrial equipments or civil engineering infrastructure, health monitoring of people working in hard conditions, predictive maintenance and environmental monitoring to detect pollution in industrial plants.

Such applications have the following requirements with regards to wireless communications:

1. Energy efficiency: Network lifetime maximization for battery-operated routers;
2. Spectrum efficiency in time and space;
3. Time-constrained communications (bounded delays and predictable data delivery time);
4. Support of micro-mobility of sensors;
5. Scalability and self-healing.

4.5.1.2 OCARI Architecture

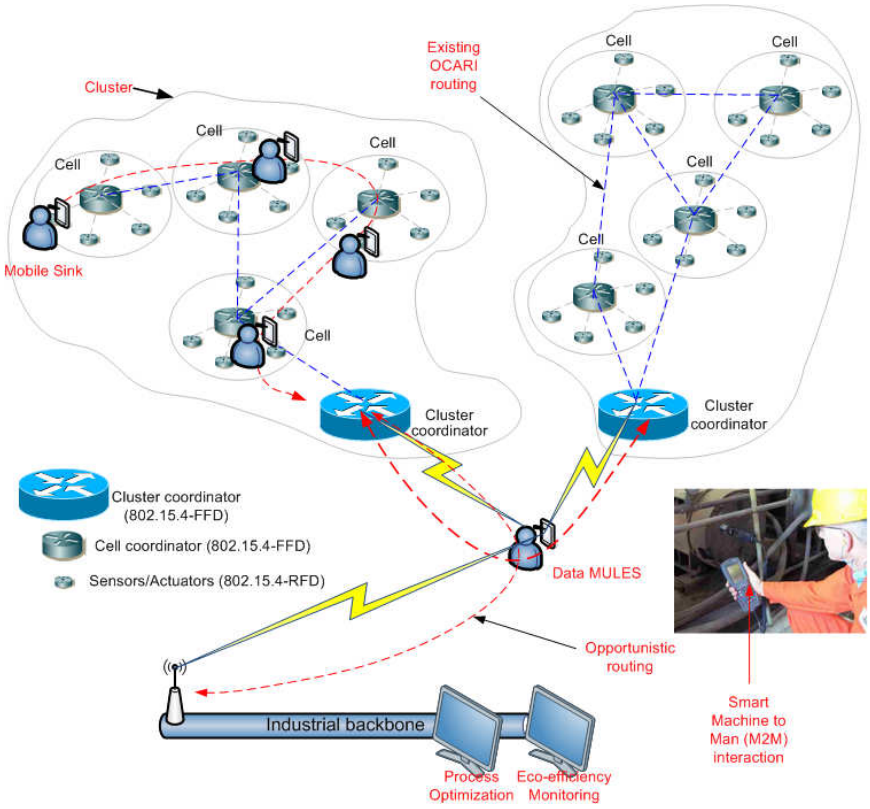


Figure 4.12: Topology of OCARI network.

An OCARI network is a mesh topology organized in a cluster of cells (see Figure 4.12). The cluster is managed by the cluster coordinator, called the CPAN. The CPAN is responsible for network address assignment, network access control, etc. It is also a gateway with the industrial facility backbone. Each cell has a star topology including a cell coordinator and its end device nodes. The cell coordinator is in charge of coordinating its end device nodes and routing their data packets. Indeed, end devices are reduced function devices (RFDs), have limited resources (energy, memory...) and can only communicate with the cell coordinator. As depicted in Figure 4.12, the OCARI network can contain mobile sinks or data mules which can be monitor operators equipped with a PDA (Personal Digital Assistant) and collecting data from close sensors.

4.5.1.3 OCARI Protocol Stack

Figure 4.13 illustrates the OCARI stack layer as in December 2011:

1. OCARI is based on the IEEE 802.15.4 physical layer like many existing industrial standards. Also, the application is based on ZigBee specification.
2. The medium access protocol, called MaCARI, guarantees a deterministic medium access

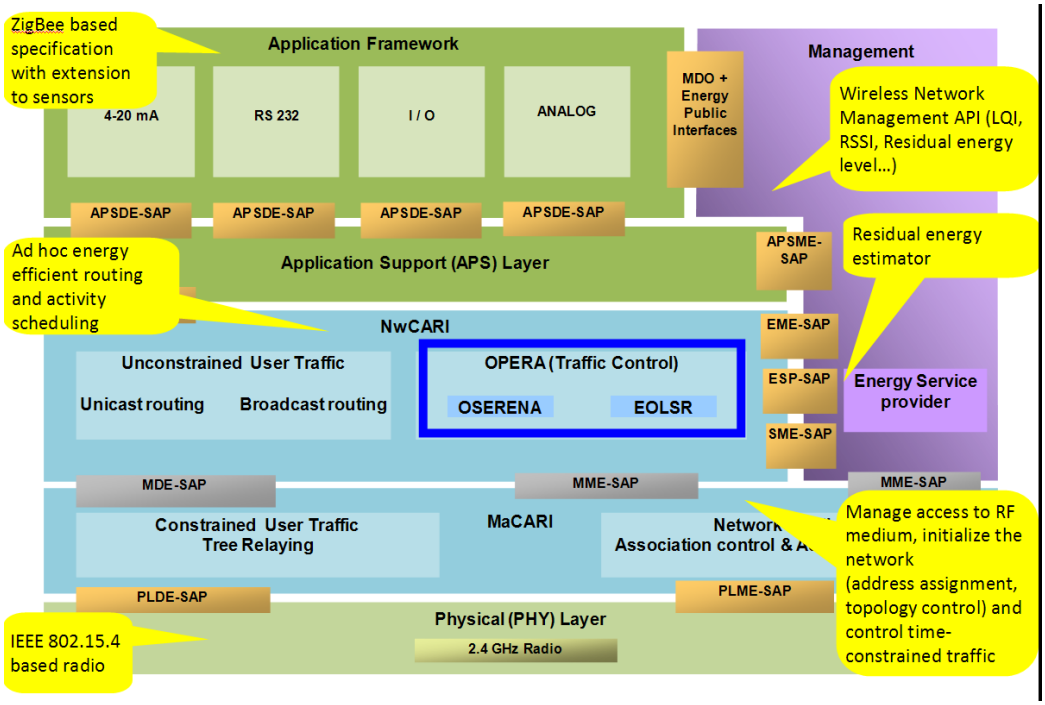


Figure 4.13: The OCARI stack as in December 2011.

for time-constrained applications. It ensures also energy efficiency and differentiation of services. MaCARI relies on a global cycle composed of three periods as illustrated in Figure 4.14.

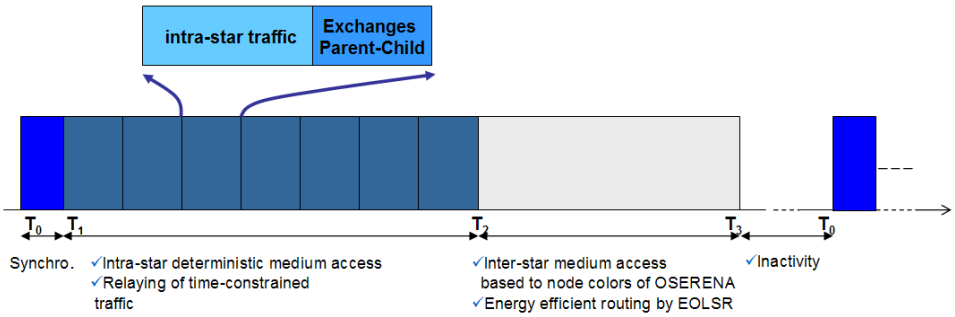


Figure 4.14: OCARI global cycle.

- $[T_0, T_1]$: A synchronization period characterized by a beacon cascading: each cell coordinator sequentially repeats the beacon received from its parent in the tree.
- $[T_1, T_2]$: It is a period where time constrained traffic is relayed. During this period, each coordinator is allocated a time interval to collect data from its end-devices on the one hand and forward data to its parent on the other hand.
- $[T_2, T_3]$: This period corresponds to the unconstrained time traffic. As we will explain

hereafter, EOLSR is used to route data between coordinators and OSERENA is used to schedule the nodes activity.

- $[T_3, T_0]$: Inactivity period for all nodes. This period is optional.

3. At the network layer, the software OPERA "OPtimized Energy efficient Routing and node Activity scheduling" provides an adaptive multi-hop routing supporting micro-mobility and enables spatiotemporal reuse of channel capacity. Indeed, OPERA consists in OSERENA and EOLSR that we will detail in Section 4.5.3. This module consists our main contribution in this project. I collaborated to the implementation of the OPERA module.
   (a) OSERENA computes the colors of the nodes and also the total number of colors max_color. The total number of colors allows MaCARI to dimension the cycle. Besides, each color computed by OSERENA corresponds to a time slot where nodes having this color can transmit.
   (b) EOLSR ensures neighborhood discovery and route construction. Indeed, EOLSR builds a routing tree rooted at the data sink. Selected routes have the smallest energy cost. EOLSR consists in two modules: "EOND" responsible of neighborhood discovery and "EOSTC" responsible for routing tree construction.

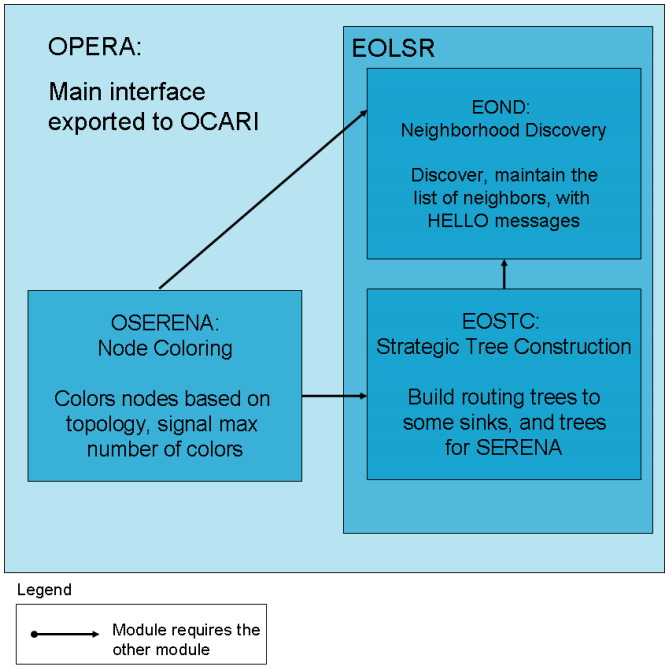


Figure 4.15: Contribution of Inria in OCARI project: OPERA module (EOLSR and OSERENA).

4. OCARI stack contains a module which estimates the residual energy of each node. This information is used by EOLSR for route selection.

In the next section, we will present our design choices for OSERENA in OCARI project.

4.5.2 OSERENA in OCARI

The coloring that we propose in OCARI is an **hybrid version** of OSERENA. Indeed, we propose 3-hop coloring as defined in the general mode, integrating rules of strategic mode: Each node has a color that is (1) different than all its neighbors up to 3 hops and (2) greater than the colors of its parent in the data gathering tree.

This combination has many benefits. First, it allows a higher flexibility of supported communications (3-hop coloring enables unicast, broadcast and immediate acknowledgement). Second, as we have explained in Chapter 3, with 3-hop coloring, any node is able to replace its parent by any 1-hop neighbor if its parent is no longer reachable, without invalidating the coloring. As we will describe in the next section, EOLSR allows any node to have a parent and a parent backup to reach the tree root. Third, the strategic mode rule obliging a node to have a color greater than the color of its parent significantly reduces the end-to-end delays and ensures freshness and time consistency of collected data.

4.5.3 EOLSR for Data Gathering Applications in OCARI

In OCARI, the routing protocol EOLSR routes time unconstrained traffic in data gathering applications. It builds a routing tree rooted at the sink (the CPAN). EOLSR is an energy efficient extension of OLSR [44]. Compared to the version presented in [30], this version is simplified as follows:

1. To enhance scaling, any node does not store its 2-hop neighbors. Consequently, routing is no longer based on MPRs (MultiPoint Relays).

2. Given that maintaining a route toward any other network node is expensive in terms of energy, storage and bandwidth, EOLSR maintains on each node only a route per sink.

3. Route selection takes into account the energy level of each node.

4. A new message is introduced (Tree Status message described in Section 4.5.4.2) to inform the tree root about the stability of the links in the tree. This information is used by the CPAN to trigger the coloring.

In this Section, we will describe all these features. We do not give details on the specifications, they are internal to the OCARI project.

4.5.3.1 EOND: Neighborhood Discovery

In EOLSR like in OLSR, the neighborhood discovery is done via the exchange of the messages *Hello*. In EOLSR, we enrich the Hello messages by information relative to the energy level of the node. This information is used to compute the energy cost of routes. Consequently, any node has knowledge about: (1) its 1-hop neighbors, (2) the type of the links with these neighbors (symmetric, asymmetric) and (3) the energy level of these neighbors. Notice here that

unlike OLSR, EOLSR does not require the storage of the 2-hop neighbors. This optimization is adopted to fit the limited storage capacity of sensor nodes.

4.5.3.2 EOSTC: Data Gathering Tree Construction

1. **Cost of routes and route selection**

In EOLSR, the cost of the route takes into account not only the length of the path, but also its energy cost:

$$\text{Cost(path)} = \sum_{\text{i=1}}^{\text{h}} \text{Cost(transmission by the i}^{\text{th}} \text{ node on the path)} \tag{4.2}$$

Where h is the number of hops in this path.
The energy consumed by any transmission of the i^{th} node, N_i, in the path takes into account the energy dissipated by the sender and the energy consumed by all the n receivers of this packet:

$$\text{Cost(transmission of N}_\text{i}\text{)} = \text{E}_{\text{transmit(i)}} + \text{n} * \text{E}_{\text{receive}} \tag{4.3}$$

Moreover, to select a route, we take into account the residual energy of each node represented by its energy class. Thus, each node determines its energy class that may vary in time. To belong to one class, the energy level of each node should be between two specific thresholds relative to this class. We define 3 classes of energy:

(a) **Class High**: Class of nodes that do not have any energy constraint, like nodes on sector for instance.

(b) **Class Medium**: Class of nodes on batteries and having enough energy.

(c) **Class Low**: Class of nodes on battery having a low residual energy.

In order to maximize the network lifetime, route selection is based on the following rules:

- Any node from class Low is chosen as a next hop if and only if there is no other option.
- Any node prefers routes composed of nodes of the class High provided that the number of hops is not prohibitive compared to a route using nodes of the class Medium.

To compute the energy dissipated in a transmission, we define:

- n_1: the number of neighbors from the class High.
- n_2: the number of neighbors from the class Medium.
- n_2: the number of neighbors from the class Low.

We define also three coefficients: α_{High}, α_{Medium}, α_{Low}, with $1 \leq \alpha_{High} < \alpha_{Medium} < \alpha_{Low}$.

We define the cost of a transmission of any node N_i based on its energy class denoted "class" and the energy class of its neighbors. From Equation (4.3), we have:

$$\begin{aligned} \text{Cost(transmission of N}_\text{i}\text{)} &= \text{E}_\text{transmit} * \alpha_{class} \\ &+ (n_1 * \alpha_{High} \\ &+ n_2 * \alpha_{Medium} \\ &+ n_3 * \alpha_{Low}) * \text{E}_\text{receive}. \end{aligned}$$

This equality allows any node to compute the cost of a route by adding the costs of different transmissions on the path applying Equation 4.2. The routes construction is done via the messages STC as we will explain in the next section.

2. **Message STC**

Routes advertisement and selection is done via the transmission of **the messages STC** (Strategic Tree Construction). Each strategic node (data collector) periodically generates a message STC containing its address and a tree sequence number to identify the tree. Nodes receiving the messages and choosing the transmitter as a parent in the tree, update the route cost in the message and transmit it (update the cost by adding to the value received in this message, the cost of a transmission of the node itself). Any node without a parent receiving a message STC selects the message transmitter as a next hop to the sink node. Moreover, any node can change its parent if it receives a message with a lower cost. Like that, EOLSR adapts to topology changes. In addition to route construction, the STC messages allow nodes to choose a parent backup. Indeed, if a node has a parent and receives a message STC from a 1-hop neighbor, it locally stores the STC transmitter as a parent backup. This parent backup should provide the second smallest cost after the parent. Consequently, if the parent is no longer reachable, the node is not obliged to wait for the next STC generated by the root, it can rather replace the parent by the parent backup if of course this latter is a valid neighbor.

4.5.4 Integration of OSERENA and EOLSR

4.5.4.1 Objectives

In OCARI, EOLSR provides information to OSERENA for:

1. **Coloring triggering** (see Section 4.5.4.2): EOLSR monitors a specific condition to allow OSERENA to start coloring using the **message Tree Status**.

2. **Topology information** (see Section 4.5.4.2): EOLSR builds the neighborhood tables used by OSERENA.

3. **Topology changes** (see Section 4.5.4.3): EOLSR informs OSERENA about the topology changes using the messages Tree Status.

4.5.4.2 Coloring Triggering

OSERENA starts automatically when the CPAN verifies that the topology is stable. This avoids recoloring in case of late start-up of a node for instance. It is the responsibility of EOLSR to notify the links stability to the CPAN.

1. In each node, EOLSR monitors a **local topology stability condition** that is defined by: (1) having a parent in the routing tree and (2) absence of neighborhood changes (node disappearance or appearance) for a predefined time interval.

2. Each EOLSR node having verified the local stability condition and received a Tree Status(stable) from all its children generates a Tree Status(stable) and transmits it to its parent.

3. At the same time, OPERA copies the topology database from EOLSR to OSERENA. This database contains: the 1-hop neighbors, the parent and the children in the routing tree, the number of descendants of the node in the tree that constitutes the priority of the node in OSERENA, the identification of the tree (address of the root and sequence number of the tree).

4. When the CPAN receives a Tree Status(stable) from all its children, it deduces that the topology is stable and can trigger the coloring.

4.5.4.3 Topology Changes

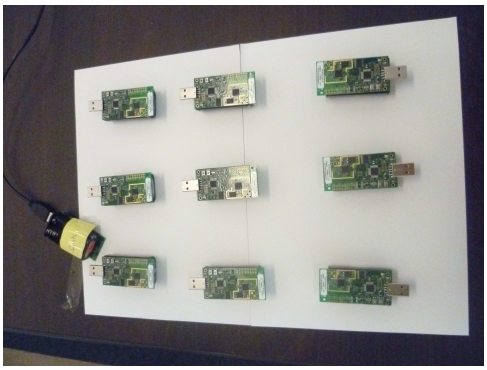
While running, EOLSR informs OSERENA about:

1. Neighborhood, parent or children changes: This information allows OSERENA to update its neighborhood table. For instance, it avoids a node to wait for a dead neighbor to color itself.

2. Instability of the tree: In addition to notifying the stability of the topology to the CPAN, Tree Status messages are transmitted to the CPAN when any node detects an instability state of the tree (Tree Status(unstable) in case of appearance of new neighbor, new parent, etc). This information is useful for the CPAN that can decide to recolor the network for instance.

4.5.5 Snapshots of Some Results of OCARI

The last event of OCARI was a meeting in December 2011 that brought together many industrials and academics. In this meeting, Inria, Limos and Telit performed a demonstration and proved the industrial feasibility of the OCARI solution by integrating all the OCARI components on the TELIT ZE51 card, based on a RF CC2420[2] with 8 Kbytes of RAM. Figure 4.16 illustrates the testbed used.



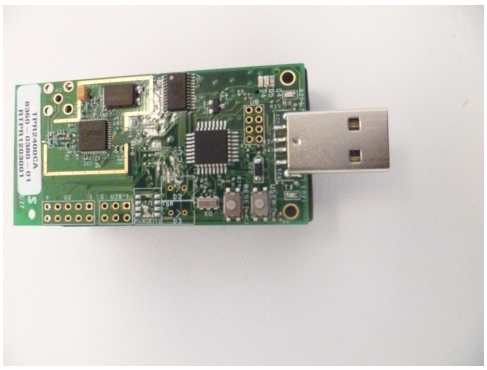
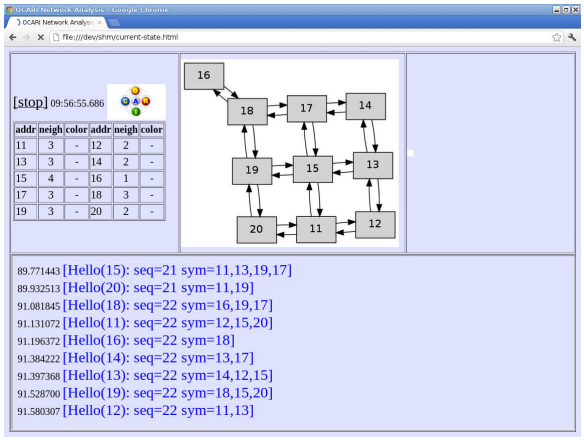
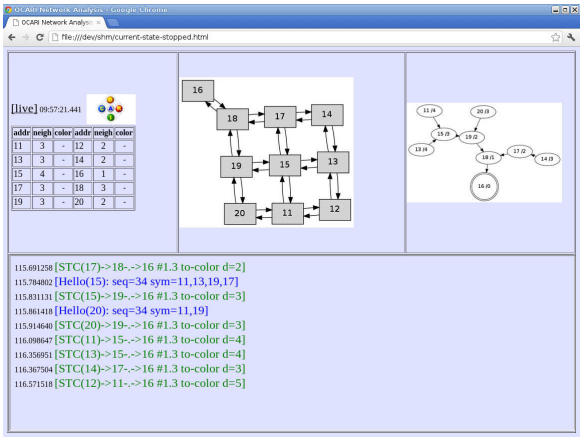


Figure 4.16: OCARI testbed.

We consider a topology reference where nodes form a grid topology. To visualize some snapshots of OCARI running on sensors, I use a tool developed in the team that is able to monitor and to show the running algorithms (messages exchanged, topology, colors, etc).



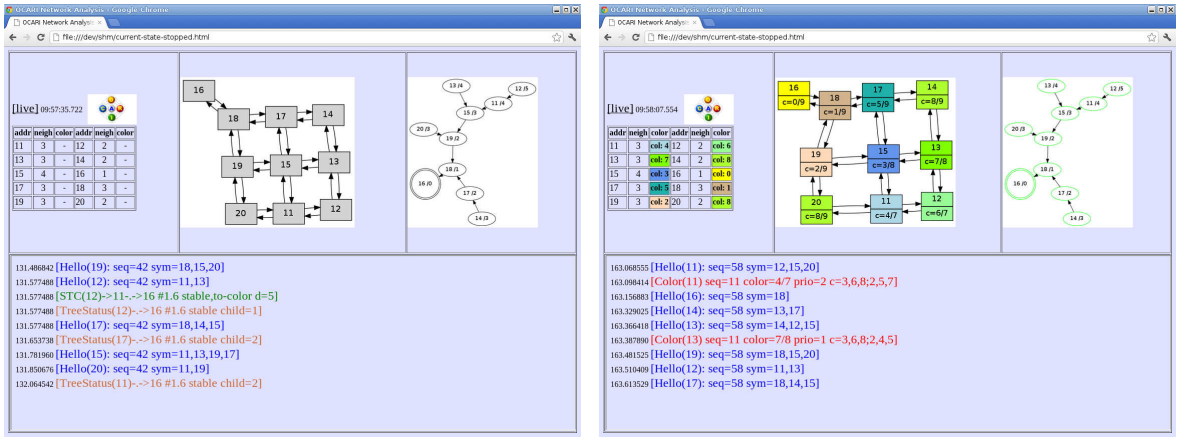
(a) Nodes exchange Hello Messages and detect symmetric links (double arrow).



(b) The sink generates a message STC that is forwarded by nodes when they select a parent. On the right, we see the tree topology. The sink node is the node 16.

Figure 4.17: Neighborhood discovery and tree Construction.

[2]http://docs.tinyos.net/tinywiki/index.php/CC2420



(a) Tree Stability: nodes detecting that the stability condition is verified start sending Tree Status(stable) messages.

(b) All nodes exchange Color messages and select colors.

Figure 4.18: Tree stability and coloring.

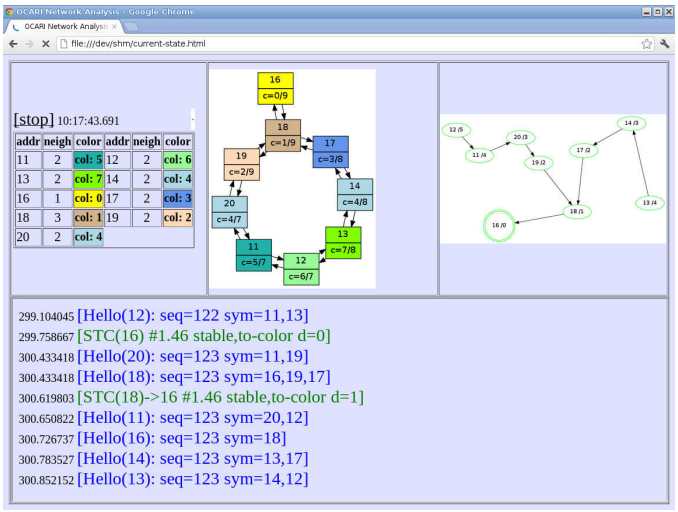


Figure 4.19: Topology changes after we removed the node 15: tree is reconstructed thanks to the periodic transmission of message STC.

4.6 Conclusion

In this chapter, we presented a new node coloring algorithm adapted to dense wireless networks that fits the limited capacity storage of sensors and the limited bandwidth. In the next chapter, we will focus on another adaptivity aspect in coloring algorithms which is the adaptivity to the topology type. We also described the OCARI project. One of the characteristics of OCARI that distinguishes it from protocols such as ZigBee, WirelessHART and Isa100.11a is its energy resource management thanks to OSERENA and EOLSR. The goal of the OCARI Alliance is to promote an open standard safe and validated for highly constrained industrial

environments. We wish the OCARI solution to be (i) largely used to increase the size of the user group, (ii) built by several manufacturers to ensure the diversity of supply sources and (iii) be a sustainable and reliable solution.

Chapter 5

Optimal Periodic Node Coloring of Grid Wireless Ad hoc and Sensor Networks

Introduction

In the previous chapters, we studied different aspects of the adaptivity of the coloring algorithms we proposed: adaptivity to application requirements, to unreliable wireless communications and to dense wireless networks. In this chapter, we propose a coloring method adapted to a specific type of topologies which is the grid. The motivations behind the focus on this kind of topology are multiple.

Indeed, the grid organization is easy to deploy and is efficient in terms of coverage, connectivity and management. For instance, a grid topology is one of the best methods to ensure sensor coverage for surveillance [115, 116]. In [117], authors proved that the grid deployment provides an optimal coverage compared to random and tri-hexagon[1] deployments. Also, in [118], it was shown that a network with regular topology yields the best average bit-error-rate performance. Grid deployment helps to collect measurements with a uniform spatial sampling such as for instance in precision agriculture and irrigation as in [119]. When physical phenomena are numerically modeled, measurements from a grid pattern may be a direct input or directly compared to the output of equations solved with the finite element method on a grid — without requiring additional sensor localization and numerical measurement interpolation. From research point of view, compared to random topologies, working on grids relatively helps to obtain interesting results, especially theoretical results. For this reason, in addition to the aforementioned characteristics of these topologies, studying these topologies is gaining more and more focus by the research community regarding many thematics like the localization and the network coverage [116]. As an example of a famous project, we can cite the project

[1] In a tri-hexagonal tiling, there are two triangles and two hexagons alternating on each vertex. http://en.wikipedia.org/wiki/Trihexagonal_tiling

SenSlab[2] that deploys an experimental testbed where sensors are disposed in a 3D grid. Given such grid topologies, the question is: What is the method to color grids while minimizing the number of colors used?

To answer this question, we start by applying OSERENA algorithm to grids with various sizes and transmission ranges (in Section 5.1). This step allows us to make some observations useful to efficiently color grids. Indeed, we notice that designing periodic coloring based on the repetition of a color pattern leads to good performance. In [120], we determined color patterns that are able to color grids for specific transmission ranges. In this chapter, we generalize this study and propose a method called VCM (Vector-based Coloring Method) able to provide the optimal coloring for any grid with any transmission range. VCM is composed of three components (a) Optimal Vector Search, (b) Node Color Computation and (c) Validity Check. We also conduct a theoretical study and determine optimal bounds on the number of colors needed to color an infinite grid.

The remaining of this chapter is organized as follows. In Section 5.2, we present the problem statement and an overview about the proposed solution. Section 5.3 defines the periodic coloring and describes its principles. Sections from 5.4 to 5.6 present the different components of VCM. In Section 5.9, we present simulation results and compare VCM to OSERENA. In Section 5.10, we discuss the applicability of VCM in real wireless ad hoc and sensor networks.

5.1 Preliminary Results and Methodology

Our objective is to color a grid with a minimum number of colors. To realise this goal, we started by applying OSERENA to a sample of grids with various sizes and transmission ranges. We considered 3-hop coloring and set the node priority to one of the following heuristics: the position of the node in the line or random. Simulation results of the number of colors and the number of rounds are summarized in Table 5.1. The optimal number of colors ('Optim' column) is given for comparison. The reader can refer to [122] for the proof of the optimal number of colors needed to color some grids with various transmission ranges. The grid step is the distance between one node and its successor/predecessor in a line/column. The transmission range is expressed as r times the grid step with r being a rational. The number of nodes ranges from 100 to 900. The density is computed as the average number of neighbors per node (i.e. average number of nodes in radio range of a sender). It varies from 3 to 25. '$Prio1$' means that there is only one criterion used for node priority assignment: either the position of the node in the grid line, or a random number. '$Prio2$' means that two criteria are used: first the number of nodes up to 2-hop and second either the position of the node in the grid line or a random number. Notice that identical results have been obtained when considering the column instead of the line in the grid. Results are the average over 10 simulation runs. From this table, we can draw the following observations:

[2] http://www.senslab.info/

Table 5.1: Number of colors and rounds obtained by OSERENA for various grids and transmission ranges ('C' means colors, 'R' means rounds).

Radio range	Grid size	Dens.	Optim	Prio1		Prio2	
				line	random	line	random
1	10x10	3.6	8C	8C 58R	13.8C 67.4R	8C 54R	11.8C 77.8R
	20x20	3.8	8C	8C 118R	15.4C 82.4R	8C 114R	14.8C 110R
	30x30	3.87	8C	8C 178R	15.4C 93R	8C 174R	15.4C 116.6R
1.5	10x10	6.84	16C	16C 67R	23.6C 94.8R	16C 65R	19.4C 107.6R
	20x20	7.41	16C	16C 137R	27.6C 144.6R	16C 134R	26.6C 166.6R
	30x30	7.6	16C	16C 207R	28.4C 169.2R	16C 204R	27.8C 195.4R
2	10x10	10.04	25C	30C 85R	33.8C 136.4R	30C 123R	28.2C 121.8R
	20x20	11.01	25C	33C 175R	41.8C 236.6R	33C 174R	36.4C 282.4R
	30x30	11.34	25C	33C 265R	44.4C 278R	34C 264R	42.8C 376.6R
2.5	10x10	15.8	45C	52C 94R	50.8C 176.8R	49C 105R	42.33C 146.17R
	20x20	17.85	45C	54C 194R	66.8C 348.2R	54C 197R	64.8C 359.8R
	30x30	18.56	45C	55C 294R	74C 422.2R	58C 297R	73.6C 440.8R
3	10x10	21.16	68C	70C 112R	68C 193R	71C 116R	67.6C 197.8R
	20x20	24.49	68C	80C 232R	93.4C 449.4R	79C 231R	93C 451.4R
	30x30	25.64	68C	83C 352R	107.8C 601.2R	81C 351R	107.8C 601.4R

1. The number or colors increases with the grid size. For instance, for a transmission range=3, we have 70 colors for 10x10 grid and 83 colors for 30x30 grid. This result is expected and is compatible with previous results of OSERENA applied to random topologies. However, given the regularity of the grid topology, we wonder if we can avoid the dependence of the number of colors on the size of the grid.

2. Mostly, we observe that no priority assignment tested provides the optimal number of colors for any grid size and any radio range. For random priority assignment and a 30x30 grid with transmission range equal to 1, we have 15.4 colors while the optimal

number is 8.

3. For a 30x30 grid and transmission range equal to 1, the optimal number of colors is reached when the priority is given by the line position. This makes us think that the regularity of the grid must be taken into account.

These results prompt the following questions: Does a priority assignment exist in grid topologies such that the number of colors does not depend on the number of nodes but only on the radio range? Is it possible to take advantage of the regularity of the grid to design a h-hop coloring algorithm able to find the optimal number of colors for any grid with any transmission range value? Moreover, does a periodic color pattern that can tile the whole topology for a given radio range, exist? What is the optimal number of colors? Is the optimal number of color reached by periodic colorings?

To answer these questions, our methodology is to determine an optimal coloring of the grid. This coloring should be periodic, that is based on the repetition of a periodic coloring pattern. Section 5.2 provides an overview about the proposed method.

5.2 Overview About the Proposed Method

5.2.1 Notations, Definitions and Assumptions

In the remaining of this chapter, we reason in a space of dimension 2 and use the following notations:

- $\vec{UV}$ denotes a vector of extremities the nodes U and V.
- For any point W, let w be the vector such that $\vec{OW} = w$ and O is the origin of the plan.
- $|\vec{UV}|$ denotes the norm, or length of the vector $\vec{UV}$ and $d(U,V)$ the euclidean distance between nodes U and V.
- $det(\vec{UV_1}, \vec{UV_2})$ is the determinant of the two vectors.
- $\vec{UV_1} \cdot \vec{UV_2}$ denotes the scalar product of the vectors.
- $\Lambda(\vec{UV_1}, \vec{UV_2})$ denotes the lattice[3] having as a basis the vectors $(\vec{UV_1}, \vec{UV_2})$. We call $\vec{UV_1}$ and $\vec{UV_2}$ the basis (or generator) vectors. $\mathcal{P}(\vec{UV_1}, \vec{UV_2})$ is the fundamental parallelotope associated to $\Lambda(UV_1, UV_2)$. In 2 dimensions, $\mathcal{P}$ is a parallelogram. Moreover, the number of nodes in $\mathcal{P}(\vec{UV_1}, \vec{UV_2})$ called *lattice determinant* is given by $det(\vec{UV_1}, \vec{UV_2})$. These notations and definitions are adopted from [124]. We also denote $\mathcal{P}_U(\vec{UV_1}, \vec{UV_2})$ the fundamental parallelotope translated at node U.

[3]Generally speaking, a lattice is a graph that exhibits a regular tiling. In our case we consider point lattice (points represent the network nodes) that is a periodic arrangement of discrete points. Two-dimensional examples are the square and the hexagonal lattices.

- For x, y in $\mathbb{Z}^2$, with $y \neq 0$, we denote "x modulo y" the integer z in $\{0, 1, \ldots |y| - 1\}$ such that we have $x \equiv z(mod|y|)$.

Moreover, the following definitions and assumptions are used:

- **Nodes:** The nodes are disposed in a grid. Without loss of generality we assume that the grid step is 1, hence the set of nodes is identified with $\mathbb{Z}^2$.
- **Neighborhood:** R denotes the radio range and is a real number ≥ 1. We adopt the unit disk model assuming that any two nodes U and V communicate directly in both directions if and only if their euclidean distance is less than R. Hence, the set of neighbors of a node U is: $\mathcal{N}(U) \triangleq \{V \in \mathbb{Z}^2 \mid 0 < d(U, V) \leq R\}$.
- **Coloring:** A coloring ϕ is a mapping from the nodes $\mathbb{Z}^2$ to a set of colors, identified with the set of integers $\{1, \ldots k\}$, with k a positive integer.
- **Valid Coloring:** A coloring is said to be a *valid h-hop coloring*, with h an integer ≥ 1, when all nodes that are less than or equal to h-hop away are assigned different colors.
- **Periodic Coloring:** A coloring is denoted "periodic" if there exists two vectors v_1 and v_2 such that for any color c all the nodes having this color form a lattice generated by vectors v_1 and v_2. In this work, we consider a "strict" definition of periodicity because colors might otherwise be periodically repeated with a pattern generating more than a single lattice for instance.
- **Reduced vectors:** We call reduced vectors of a lattice, the vectors obtained by applying the Gauss lattice reduction on these vectors. Indeed, for any pair of vectors u_1, u_2 generating a lattice $\Lambda(u_1, u_2)$, the Gauss lattice reduction algorithm provides two *reduced* vectors v_1, v_2 generating exactly the same lattice and verifying the System (5.1)

$$\begin{cases} |v_1| \leq |v_2| \\ 2|v_1 \cdot v_2| \ \leq \ |v_1|^2 \end{cases} \tag{5.1}$$

  See for instance [121] for more details.

5.2.2 Problem Statement

Our general goal is the following: *Find a valid h-hop coloring of nodes of the infinite grid $\mathbb{Z}^2$, with a minimized number of colors for $h \geq 1$.* Because the set of colorings of $\mathbb{Z}^2$ is infinite, in this chapter we restrict ourselves to colorings exhibiting a periodic color pattern (following the definition of periodic coloring of section 5.2.1).

> **Problem Statement:** Find one of the valid periodic h-hop colorings with the minimum number of colors for $h \geq 1$.

5.2.3 Intuitive Idea and Overview

To answer the problem statement, we propose the method *VCM*. The intuitive idea of the method is as follows: As the grid topology presents a regularity in terms of node position, *VCM* produces a similar regularity in terms of colors and generates a color pattern that can be periodically reproduced to color the whole grid. Our aim is to find such a color pattern that minimizes the number of colors used. More precisely, given a colored node U, we determine where its color can be reproduced. The method starts by the choice of two vectors $U\vec{V}_1$ and $U\vec{V}_2$ such that V_1 and V_2 use the same color as U. Of course, the vectors $U\vec{V}_1, U\vec{V}_2$ must provide a valid h-hop coloring. The color pattern is given by the set of colors in the finite parallelogram $\mathcal{P}(U\vec{V}_1, U\vec{V}_2)$ translated at U. Hence, the color of U is repeated also at any node W where $U\vec{W}$ is a linear combination of $U\vec{V}_1$ and $U\vec{V}_2$. Consequently, as illustrated in Figure 5.1, *VCM* is composed of the following components:

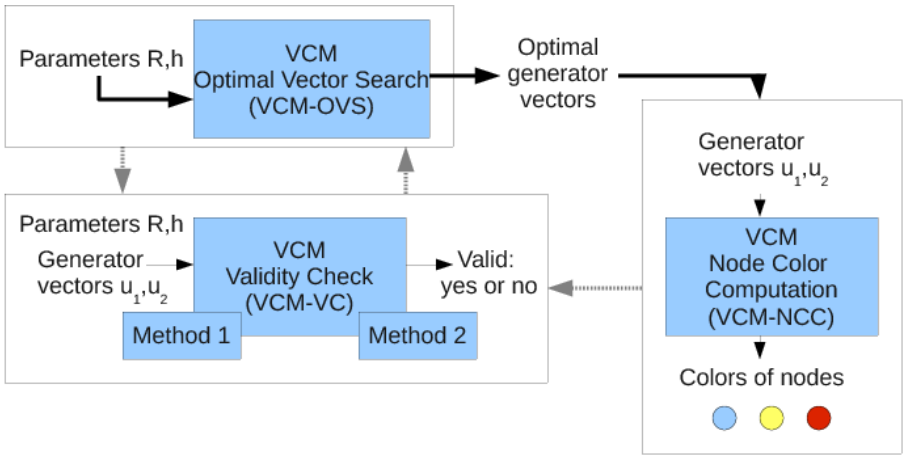


Figure 5.1: Components of VCM

C1. VCM-OVS An algorithm to search for optimal generator vectors, that is, yielding the minimum number of colors. Indeed, we limit the set of candidate vectors to find the vectors providing the optimal number of colors.

C2. VCM-NCC A periodic coloring based on two generator vectors. This component determines the color of any node.

C3. VCM-VC An algorithm to check the validity of the periodic coloring: VCM incorporates two methods to verify the validity of the coloring for a given couple of generator vectors.

In Sections 5.4 to 5.6 we detail the components of *VCM*.

5.3 Periodic Coloring

Figure 5.2 presents an example of a periodic 3-hop coloring of a 10×10 grid with a transmission range $R = 1$¡ and two generator vectors $U\vec{V}_1$ and $U\vec{V}_2$ (see [123] for more examples). The basic principles of the periodic coloring are:

P1. (Generator vectors) The two vectors $U\vec{V}_1$ and $U\vec{V}_2$, if linearly independent, generate the parallelogram $\mathcal{P}(U\vec{V}_1, U\vec{V}_2)$ of the color pattern.

P2. (Parallelogram color unicity) Inside $\mathcal{P}(U\vec{V}_1, U\vec{V}_2)$, there is no color reuse.

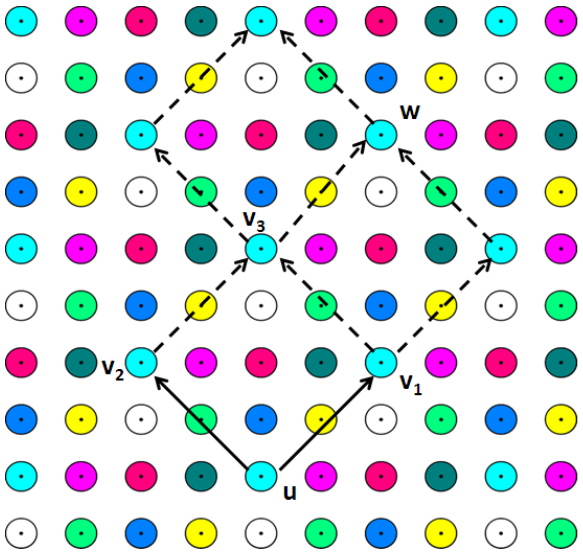


Figure 5.2: Example of periodic 3-hop coloring (R=1)

P3. (Lattice color repetition) Because the periodic coloring is obtained by repeating the color pattern, the nodes W such that $U\vec{W}$ is a linear combination of $U\vec{V}_1$ and $U\vec{V}_2$ have the same color as U. More precisely, the set of nodes W having the same color as U forms a *lattice* of $\mathbb{Z}^2$ with generator vectors $U\vec{V}_1$ and $U\vec{V}_2$: the vector $U\vec{W}$ can be written as $U\vec{W} = \alpha U\vec{V}_1 + \beta U\vec{V}_2$ with α and β in $\mathbb{Z}^2$.

P4. (Coordinate-based color computation) The grid can be seen as a tiling with the pattern $\mathcal{P}_{\mathcal{U}}(U\vec{V}_1, U\vec{V}_2)$. Thus, each node W belongs to its own parallelogram and has coordinates relative to this parallelogram. Consequently, W has the same color as any node having the same coordinates in its own parallelogram.

To be applied to a wireless ad hoc and sensor network, these principles have to be enhanced. For instance, since nodes having the same color can simultaneously access the wireless medium, validity of the coloring must be verified. Hence, we search **valid** vectors (i.e. vectors generating valid coloring). Moreover, to ensure an efficient usage of the bandwidth, the number of colors used should be minimized. These criteria are taken into account in our work and progressively introduced in the paper.

In the following we detail the components of *VCM*.

5.4 Optimal Vector Search (OVS)

To achieve an optimal spatial reuse, the coloring algorithm should minimize the number of colors used. For *VCM*, this number is entirely defined by the two generator vectors describing the periodic tiling (see property **P3**). Hence, our aim is to judiciously choose these vectors in order to reduce the number of colors used to color a grid.
However, by default, the infinite lattice $\mathbb{Z}^2$ is a possible set for candidate vectors. So, to reduce the size of the set of candidate vectors, our approach takes advantage of two hints:
1. We determine upper and lower bounds on the number of colors needed in a h-hop coloring of the grid, for $h \geq 1$.
2. Because for any couple of initial generator vectors, reduced vectors always exist, it is sufficient to search for some optimal vectors in the space of reduced vectors defined by System (5.1).
This is used to restrict the search space as we will see in Section 5.4.2.

5.4.1 Bounds on the Number of Colors in Periodic Colorings

In this section, we prove that the number of colors of optimal coloring when $R \to \infty$ is asymptotically $\frac{\sqrt{3}}{2}h^2R^2 + O(R)$. We first start with two examples of construction of valid generator vectors, that are used to determine the bounds on the number of colors and to limit the vectors search space.

5.4.1.1 Examples of Valid Generator Vectors

• **Near hexagonal generator vectors**

Here, we detail how to construct two valid vectors $U\vec{V}_1$ and $U\vec{V}_2$ which provide an approximation of an hexagonal lattice (when R is large). Figure 5.3 illustrates how some points V_1 and V_2 are constructed starting from some reference point U.

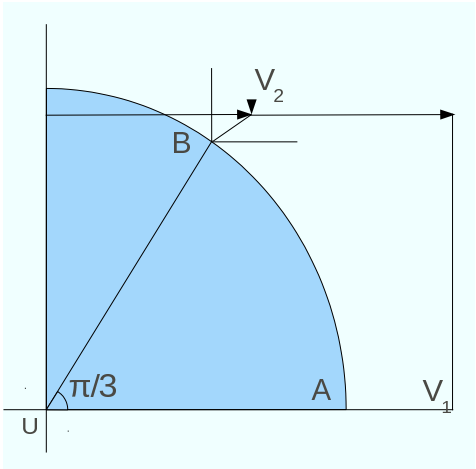


Figure 5.3: Selecting vectors for a near-hexagonal lattice

- Starting from the point U, the line with an angle $\frac{\pi}{3}$ with the horizontal line is considered, and its intersection with the circle of radius hR yields the point B.
- Next, let V_2 be the closest point of B on the grid having larger coordinates x_2 and y_2 than B.
- Then V_1 with coordinates (x_1, y_1) is selected with $(x_1, y_1) = (2x_2, 0)$.

Notice that by construction x_1 is greater than hR and we have a valid choice of vectors $U\vec{V}_1$ and $U\vec{V}_2$.

• **Square generator vectors**

An alternate, simpler, choice of vectors is to compute the integer $\lambda = \lfloor hR \rfloor + 1$, (this symbol denotes the integer part of hR) and select the vectors with coordinates $u'_1 = (\lambda, 0)$ and $u'_2 = (0, \lambda)$. They generate a square lattice.

5.4.1.2 Lower Bound

For the lower bound on the number of colors, we have the following theorem that is valid for any coloring (not just periodic coloring):

Theorem 2. *The number of colors required to color an infinite grid with $R > \sqrt{2}$ is at least $\frac{\sqrt{3}}{2}h^2(R - \sqrt{2})^2$.*

Proof: Consider any h-hop coloring of the grid. Consider a fixed color c and now let S_c be the set of nodes having this color. We first establish a lower bound of the distance of nodes in S_c. Let us define $\rho = (R - \sqrt{2})h$. Consider two nodes A, B of S_c. By contradiction: if their distance verifies $d(A, B) \leq \rho$, they would be at most h-hop away, contradicting the definition of a h-hop coloring (see Lemma 17 in Appendix B). Therefore, all nodes of S_c are at a distance at least ρ from each other.

Now consider the set of circles $\mathcal{C}$ of radius $\frac{1}{2}\rho$ and whose centers are the nodes of S_c. The fact that any two nodes of S_c are distant of more than ρ, implies that none of the circles in $\mathcal{C}$ overlap. Hence $\mathcal{C}$ is a *circle packing* by definition. From the Thue-Tóth theorem [125, 126] establishing that the hexagonal circle packing is the densest packing, with a density of $\frac{\pi}{\sqrt{12}}$, we deduce that $\mathcal{C}$ must have a lower or equal packing density. This implies an upper bound of the density of set S_c of centers of the disks of $\frac{1}{(\rho/2)^2\sqrt{12}}$.

Because each color yields a set of nodes with at most this density, it follows a lower bound of the number of colors that is the inverse of this quantity, hence the theorem. ■

5.4.1.3 Upper Bound

For an upper bound, we have the following theorem:

Theorem 3. *The number of colors required to color an infinite grid is at most* $\frac{\sqrt{3}}{2}h^2R^2 + 2hR + (2 + hR)\sqrt{2}$.

Proof: This number is obtained from the near hexagonal vectors constructed in Section 5.4.1.1, which thus yields an upper bound.

Denote (γ, δ) the coordinates of $B\vec{V}_2$ from Section 5.4.1.1. By construction: $0 \leq \gamma \leq 1$ and $0 \leq \delta \leq 1$, hence: $|B\vec{V}_2| \leq \sqrt{2}$. Moreover,

$$\begin{aligned} |A\vec{V}_1| &= 2x_2 - hR \\ &= 2(hR\cos(\frac{\pi}{3}) + \gamma) - hR, \text{ with } \gamma \leq 1 \\ &\leq 2 \end{aligned}$$

Consequently, we can write n_c, the number of colors in the associated coloring as:

$$\begin{aligned} n_c &= det(U\vec{V}_1, U\vec{V}_2) \\ &= det(U\vec{A}, U\vec{B}) + det(A\vec{V}_1, U\vec{B}) + det(U\vec{V}_1, B\vec{V}_2) \\ &\leq det(U\vec{A}, U\vec{B}) + |A\vec{V}_1||U\vec{B}| + |U\vec{V}_1||B\vec{V}_2| \end{aligned}$$

As we have $|A\vec{V}_1| \leq 2$, $|B\vec{V}_2| \leq \sqrt{2}$ and $|U\vec{V}_1| = hR + 2$, we get:

$$n_c \leq \frac{\sqrt{3}}{2}h^2R^2 + 2hR + (hR + 2)\sqrt{2}.$$

Thus the theorem. ■

Remark 5. *The number of colors of the construction of square generator vectors, proposed in previous section, is higher than for the near hexagonal vectors. It is* $h^2R^2(1+O(\frac{1}{R}))$*, hence the vectors cannot yield the tight bound, but the vectors are used later in Section 5.4.2.2 for an alternate upper bound.*

5.4.1.4 Asymptotic Number of Colors

Theorem 4. *The number of colors* $n_c(R)$ *of an optimal periodic h-hop coloring for a fixed h verifies:*

$$n_c(R) = \frac{\sqrt{3}}{2}h^2R^2(1 + O(\frac{1}{R}))$$

when $R \to \infty$.

Proof: Combining the lower bound and the upper bound of the two theorems yields the result. ■

Corollary 1. *Theorem 4 is true even considering periodic and non periodic coloring.*

Proof: A periodic coloring is a special case of general coloring (including periodic and non periodic). Hence, the optimal number of colors in general coloring is less than or equal to n_c. ■

Corollary 2. VCM *is asymptotically optimal even considering all possible valid colorings. In other terms* VCM *is an* $(1 + g(R))$*-approximation of the optimal coloring of the grid with some* g *verifying* $g(R) \to 0$ *when* $R \to \infty$.

Proof: *VCM* will find vectors with better or equal performance than those in Theorem 3. Indeed, in the worst case, the generator vectors for the near-hexagonal lattice will be selected by *VCM*. ■

5.4.2 Finding the Optimal Vectors

In this section, we show how to limit the set of candidate vectors. Let u_1 and u_2 be two candidate vectors and θ be the angle between them. We search l_{1min}, l_{1max} (respectively l_{2min}, l_{2max}) the lower and upper bounds of the length of u_1 (respectively u_2).

• Considering a h-hop coloring, the vectors u_1 and u_2 must be valid. According to Lemma 17 from Appendix B, we have:

$$|u_1| > (R - \sqrt{2})\ h \text{ and } |u_2| > (R - \sqrt{2})\ h. \tag{5.2}$$

• As previously said, in order to reduce the set of candidate generator vectors, we reduce the size of these vectors by using the lattice reduction algorithm of Gauss. A consequence of Gauss property (System 5.1) is that: $|\cos\theta| \leq \frac{1}{2}$, hence:

$$|\sin\theta| \geq \frac{\sqrt{3}}{2}. \tag{5.3}$$

• As shown in Theorem 2,

$det(u_1, u_2) \leq S_h = \frac{\sqrt{3}}{2}h^2R^2 + 2hR + (2 + hR)\sqrt{2}.$

It results: $|u_1||u_2||\sin\theta| \leq S_h$. Using Inequality 5.3, we get:

$$\frac{\sqrt{3}}{2}|u_1||u_2| \leq |u_1||u_2||\sin\theta| \leq S_h \tag{5.4}$$

And as $|u_1| \leq |u_2|$, from 5.4 we have:

$$\frac{\sqrt{3}}{2}|u_1|^2 \leq S_h. \tag{5.5}$$

We now distinguish the cases where $R > \sqrt{2}$ and $R \leq \sqrt{2}$

5.4.2.1 Case $R > \sqrt{2}$

Using 5.2, from 5.5 we get:

$$\frac{\sqrt{3}}{2}|u_1|(R-\sqrt{2})h < \frac{\sqrt{3}}{2}|u_1|^2 \leq S_h. \tag{5.6}$$

And using 5.2 in 5.4:

$$\frac{\sqrt{3}}{2}|u_2|(R-\sqrt{2})h < S_h. \tag{5.7}$$

To summarize, the two generator vectors should verify for $R > \sqrt{2}$:

$$l_{1min} < |u_1| \leq l_{1max} \text{ and } l_{2min} < |u_2| < l_{2max} \tag{5.8}$$

with:

$$\begin{cases} l_{1min} = h(R-\sqrt{2}) \\ l_{1max} = \sqrt{\frac{2}{\sqrt{3}}S_h} \\ l_{2min} = h(R-\sqrt{2}) \\ l_{2max} = \frac{2}{\sqrt{3}}\frac{S_h}{h(R-\sqrt{2})} \end{cases} \tag{5.9}$$

In practice, to compute the two generator vectors, we determine the upper and the lower bounds of the coordinates of u_1, and u_2 using the System (5.9). Notice that we can search the vectors in the half plane ($y \geq 0$), because if u_1 and u_2 are generator vectors, then their symmetric vectors with respect to ($y = 0$) axis are also generator vectors. Consequently, we have:

$$\begin{cases} -l_{1max} \leq x_1 \leq l_{1max} \\ 0 \leq y_1 \leq l_{1max} \end{cases} \tag{5.10}$$

and

$$\begin{cases} -l_{2max} \leq x_2 \leq l_{2max} \\ 0 \leq y_2 \leq l_{2max} \end{cases} \tag{5.11}$$

5.4.2.2 Case $R \leq \sqrt{2}$

We set $l_{1min} = 0$. In addition, we can use the bound implied by the square generator vectors as proposed in Remark 5. Let $S_s = (hR+1)^2$, which replaces S_h for the computation of l_{1max}. Then, instead of a fixed bounds for l_{1max} and l_{2max}, we propose a bound depending on u_1, by using (5.4) we have: $l_{2max} = \frac{2\sqrt{3}S_s}{3|u_1|}$, and because $|u_2| \geq |u_1|$ we have: $l_{2min} = |u_1|$. Hence:

$$\begin{cases} 0 < |u_1| \leq hR+1 \\ |u_1| \leq |u_2| \leq \frac{2\sqrt{3}(hR+1)^2}{3|u_1|} \end{cases} \tag{5.12}$$

To find the optimal vectors, we define the Method OVS.

Method OVS:
1. The first step is to search S_1 the initial set of generator vectors u_1 and u_2. S_1 is the set of vectors having as coordinates the integers (x_1, y_1) and (x_2, y_2) verifying System (5.10) and System (5.11) if $R > \sqrt{2}$ or otherwise System (5.12).
2. Now, the set S_1 should be filtered to keep only reduced and valid vectors. Indeed, for each couple of vectors (u_1, u_2) in S_1, we should verify:
2.1. (u_1, u_2) are reduced, that is they verify System (5.1).
2.2. to check the validity of the coloring, we use one of the methods that will be explained in Section 5.6. Indeed, two cases are possible:
2.2.1 if $R > \sqrt{2}$ apply VC2.
2.2.2. otherwise, apply VC1.
3. After the step 2, we obtain the set of valid reduced vectors. Now, the optimal vectors are then the vectors having the smallest absolute value of their determinant.
Notice that the search of the optimal vectors can be performed by a central unit that will then disseminate these two vectors to all nodes. It is also possible that each node in the grid computes the two generator vectors.

5.5 VCM: Node Color Computation (NCC)

5.5.1 Assigning Colors to Nodes

The node color computation (NCC) component of the VCM method takes as parameters two generator vectors u_1,u_2 (computed as in Section 5.4). Let (x_1, y_1) and (x_2, y_2) be their coordinates and let $d = \det(u_1, u_2)$. Here, we only provide one method VCM-NCC1, we can find another method in [127]. An example of computation is provided in Section 5.5.3.

5.5.1.1 Method NCC1

Method VCM-NCC 1: *VCM* assigns the color of a point W based on its coordinates $w = (x, y)$ by computing first the integer quantities $c_1(w), c_2(w)$ as in System (5.13),

$$\begin{cases} c_1(w) = (xy_2 - yx_2) \text{ modulo } d \\ c_2(w) = (-xy_1 + yx_1) \text{ modulo } d \end{cases} \tag{5.13}$$

and then using a bijective mapping between the couple (c_1, c_2) and a color $\in \{0, 1, 2, \ldots |d| - 1\}$.

Remark 6. *In the remaining of this report, we will assume that $d > 0$ without loss of generality: indeed, if $d < 0$, it is sufficient to use the vectors $(-u_1, u_2)$ instead of (u_1, u_2) and the results are similar ; notice that $c_1(w, u_1, u_2) = c_1(-w, -u_1, u_2)$, etc. (hence change of sign of u_1 is equivalent to an origin symmetry of the coloring). This avoids minor technicalities*

on the definition of the modulo, integer part, fractional part, when numbers are negative (for which definitions are not universal).

Property 6. *With the previous coloring VCM-NCC1, it is indeed possible to define a bijection from* $(c_1(w), c_2(w))$ *to* $\{0, 1, 2, \ldots |d| - 1\}$. *Moreover, the coloring verifies principles* **P3** *and* **P4** *defined in Section 5.3.*

Proof: We assume that $d > 0$ without loss of generality (see Remark 6). We need to prove that the set $\left\{(c_1(w), c_2(w)) \mid w \in \mathbb{Z}^2\right\}$ has cardinality d and that the set of nodes with the same color is exactly the lattice $\Lambda(u_1, u_2)$ translated at w.

Let W be a grid point of coordinates $w = (x, y)$. Performing a change of vector basis in $\mathbb{R}^2$ from $\{(1, 0), (0, 1)\}$ to $\{u_1, u_2\}$, the new coordinates $(\alpha, \beta) \in \mathbb{R}^2$ of W in $\Lambda(u_1, u_2)$ verify $w = \alpha u_1 + \beta u_2$ and

$$\begin{cases} \alpha = \dfrac{\det(w, u_2)}{d} \\ \beta = \dfrac{\det(u_1, w)}{d} \end{cases} \tag{5.14}$$

with $d = det(u_1, u_2)$.

Let α' and β' be the integer parts of α, β, i.e., $\alpha' = \lfloor \alpha \rfloor$ and $\beta' = \lfloor \beta \rfloor$. For arbitrary nonzero integers λ, μ, with also $\mu > 0$, we have the identity: $\frac{\lambda}{\mu} = \lfloor \frac{\lambda}{\mu} \rfloor + \frac{\lambda \text{ modulo } \mu}{\mu}$. Thus, 5.14 becomes:

$$\begin{cases} \alpha = \alpha' + \dfrac{\det(w, u_2) \text{ modulo } d}{d} = \alpha' + \dfrac{c_1(w)}{d} \\ \beta = \beta' + \dfrac{\det(u_1, w) \text{ modulo } d}{d} = \beta' + \dfrac{c_2(w)}{d} \end{cases} \tag{5.15}$$

Let W' the point with coordinates $w' = (\alpha', \beta')$. W' is on the lattice since α', β' are integers, and observe that W is in fact inside the parallelogram of the lattice $\Lambda(u_1, u_2)$ placed at node W' (i.e. inside the parallelogram defined by the 4 points of the lattice: w', $w' + u_1$, $w' + u_2$, $w' + u_1 + u_2$). Then (5.15) means simply that $(\frac{c_1(w)}{d}, \frac{c_2(w)}{d})$ are the coordinates of W relative to this parallelogram (with the basis vectors u_1, u_2).

Since there is a bijection between the set of coordinates of nodes in a parallelogram of $\Lambda(u_1, u_2)$ and the nodes themselves; and since $(\frac{c_1(w)}{d}, \frac{c_2(w)}{d})$ are these coordinates, we have the two properties: 1) there are exactly d possible values of (c_1, c_2) (because there are exactly d nodes in the parallelogram), and 2) no two nodes inside the parallelogram have the same values c_1, c_2 since these are their coordinates, relative to one vertex of the parallelogram. ■

Lemma 14. *With the color computation given by System* 5.13*, the color of the node U is repeated at nodes W with coordinates verifying:* $w = \alpha u_1 + \beta u_2$*, for some* $(\alpha, \beta) \in \mathbb{Z}^2$.

Proof: Actually, by construction, the color of a node is given by its coordinates relative to the parallelogram it belongs to. Hence, the color of any node U is reused at nodes W, such that $U\vec{W} = \alpha u_1 + \beta u_2$ for all $(\alpha, \beta) \in \mathbb{Z}^2$, which have the same relative coordinates. ■

We deduce that *VCM-NCC 1* provides a coloring that is really consistent with the principles of the periodic coloring as described in Section 5.3.

5.5.1.2 Example of Bijection for NCC1

A bijection between $\{(c_1(w), c_2(w)) \mid w \in \mathbb{Z}^2\}$ and the set of colors $\{0, 1, \ldots, d-1\}$ can be constructed by computing the values of $(c_1(w), c_2(w))$ for any node in $\mathcal{P}(u_1, u_2)$ in a list, sorting the list by lexicographical order and finally, setting the color associated with a couple $(c_1(w), c_2(w))$ to be its index in the sorted list minus 1.

Example: if $(0, 0)$ appears as the 1^{st} item of the sorted list [as it can be proved it will], the color assigned to that couple is 0. Then, for instance, for the point W of coordinates $w = u_1$, we have $(c_1(w), c_2(w)) = (0, 0)$ and therefore the color assigned to W is 0.

Note that, from a pure implementation point of view, it may be difficult to enumerate exactly the points of $\mathcal{P}(u_1, u_2)$, but then, instead, it is sufficient to enumerate all the nodes in a superset, the *bounding box* of $\mathcal{P}(u_1, u_2)$, itself computed from its four vertices $O = (0, 0)$, $O + u_1$, $O + u_2$ and $O + u_1 + u_2$. Computing the set of values $(c_1(w), c_2(w))$ for the points in the bounding box, will yield all possible values for any $w \in \mathbb{Z}^2$.

5.5.2 Computing the Number of Colors

The number of colors used in a periodic h-hop coloring is given by the next Property.

Property 7. *For any node U, the color pattern defined by the two generator vectors u_1 and u_2 meeting the aforementioned principles contains exactly $|x_1y_2 - x_2y_1|$ colors where (x_1, y_1) and (x_2, y_2) are the coordinates of u_1 and u_2.*

Proof: By definition, no two nodes within the parallelogram defined by u_1 and u_2 use the same color. Hence the number of colors is equal to the number of nodes in this parallelogram. Moreover, as we said, the number of nodes in $\mathcal{P}(u_1, u_2)$, called *lattice determinant*, is equal to the absolute value of $\det(u_1, u_2)$ [124]. Hence the property. ■

5.5.3 Example of Color Calculation

In this section, we illustrate the color calculation, using the following example (see Figure 5.4):

- R=3 and h=2;
- The OVS method determines the optimal vectors which are $u_1 = (6, -1)$ and $u_2 = (3, 5)$;
- Consider the node $w = (5, 3)$, we want to determine its color.

Applying VCM-NCC 1 with the generator vectors u_1, u_2, we get:

- Number of colors = 33;
- $(c_1(w, u_1, u_2), c_2(w, u_1, u_2)) = (16, 23)$
- Using the example bijection of Section 5.5.1.2; $c_1(w), c_2(w)$ is the 17^{th} value in the sorted list of possible values, hence color$(w) = 16$.

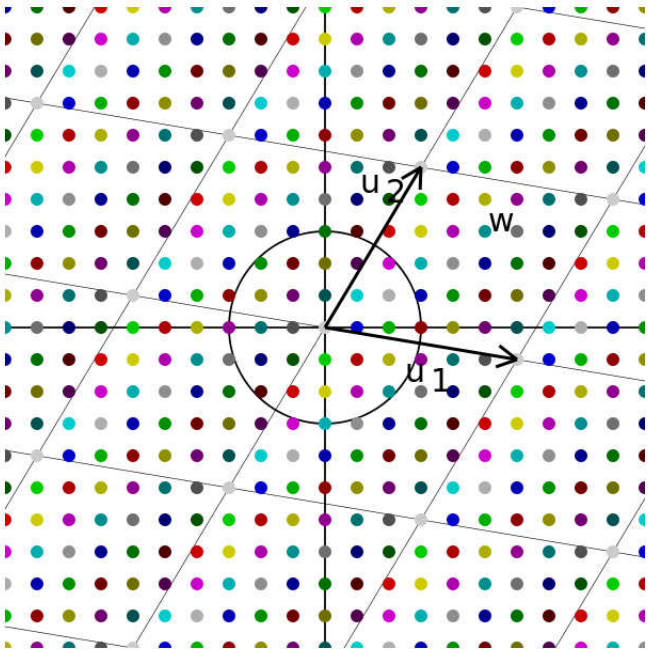


Figure 5.4: An example of coloring computation based on VCM-NCC1.

5.6 VCM: Validity Check (VC)

As defined previously, a h-hop coloring algorithm is valid if and only if no two nodes that are at less than or equal to h-hop from each other use the same color. The node color computation algorithm of VCM (described in Section 5.5) takes as input two generator vectors u_1, and u_2, and gives the color of each node. In this section, we will assume that such two vectors are given and fixed, and we present two methods for checking beforehand whether the coloring induced by these vectors is a valid coloring.

5.6.1 Method VC1: Verification around Origin

Method VC1: For each node W in the h-hop neighborhood of the origin node O, we compute the color of this node based on the given generator vectors u_1 and u_2. If W has the same color as O, then we conclude that the vectors u_1 and u_2 do not provide a valid coloring. Otherwise, if the color of O is not repeated at any point W in its h-hop neighborhood, then the coloring is valid.

The idea of Method VC1 is based on the following fact proved in this section: if there is a color conflict between any two nodes V_1 and V_2 in $\Lambda(u_1, u_2)$ (V_1 and V_2 have the same color despite they are at less than or equal to h hops), there will be a color conflict in the h-hop neighborhood of the origin O.
We set $d = \det(u_1, u_2)$.

Lemma 15. *If two nodes V_1 and V_2 with coordinates v_1, v_2 in $\mathbb{Z}^2$ have the same color, then the color of the origin node is repeated at the node W of coordinates $v_2 - v_1$.*

Proof: The functions c_1, c_2 computed from System 5.13 are actually linear modulo d. That is, if W is the node with coordinates $v_1 - v_2$, and w is the vector of nodes extremities the origin and W, we get: $c_1(W) = c_1(V_1) - c_1(V_2)$ modulo d. Hence, if $c_1(V_1) = c_1(V_2)$ we have $c_1(W) = c_1(O)$. This is true also for c_2, hence the lemma. ■

Theorem 5. *If none of the nodes inside the h-hop neighborhood of the origin node $O = (0, 0)$ has the same color as O itself, then the coloring is valid.*

Proof: By contradiction: assume that the coloring is invalid, which implies that two nodes V_1, V_2 at less than or equal to h hops have the same color. Then from Lemma 15, the node W such as $O\vec{W} = V_1\vec{V}_2$ has the same color as O. Notice that the distance in terms of hop number between O and W is the same as the distance between V_1 and V_2. Hence we have found a color conflict between O and a node W which is at less than h hops from O. Hence the theorem. ■

Theorem 5 proves that Method VC1 is a correct method for checking whether two generator vectors yield a valid h-hop coloring.

5.6.2 Method VC2: Verification in a Few Points

Method VC1 requires $\Theta(R^2)$ verifications when $R \to \infty$. In the following, we propose Method VC2, usable when $R > \sqrt{2}$ and requiring only a bounded number of verifications. Method VC2 performs a check on a few nodes on the lattice $\Lambda(u_1, u_2)$ to guarantee that the h-hop coloring associated to u_1, u_2 is valid. This method is based on Gauss lattice reduction [121]: u_1 and u_2 should be first reduced, and hence verify System (5.1).

Method VC2: The nodes with the same color as the origin are on the lattice $\Lambda(u_1, u_2)$: this method verifies that these nodes are at least $(h+1)$-away from O, in which case the coloring is valid. However, not all grid nodes need to be checked. It is sufficient to check only nodes W in $\Lambda(u_1, u_2)$ with coordinates α, β on the basis $\{u_1, u_2\}$, such that $|\alpha|$ and $|\beta| < \mu(R)$, with $\mu(R) = \frac{2\sqrt{3}R}{3(R-\sqrt{2})}$. The coloring is valid if and only if these nodes are strictly more than h hops from the origin node.

This method is based on the following theorem.

Theorem 6. *For $R > \sqrt{2}$, the coloring provided by two reduced vectors u_1, u_2 is valid if and only if:*
for all α, β integers verifying $|\alpha| < \mu(R)$, and $|\beta| < \mu(R)$, the node with coordinates (α, β) on the basis $\{u_1, u_2\}$ is at strictly more than h hops from the origin node O, where $\mu(R) = \frac{2\sqrt{3}R}{3(R-\sqrt{2})}$.

Proof: The property comes from the fact that the points on the lattice $\Lambda(u_1, u_2)$ are "far" from the origin node, because the vectors u_1, u_2 are reduced.
Indeed, Lemma 20 (see Appendix B) means that any node on the lattice with coordinates

$\alpha u_1 + \beta u_2$, with $|\alpha|$ or $|\beta| \geq \mu(R)$ can reuse the color of the origin node O because they are at strictly more that h-hop from O (provided that the points of coordinates u_1 or u_2 are themselves strictly more than h-hop away from O). Hence, to check the validity of the coloring provided by *VCM*, it is necessary and sufficient to check that for all $|\alpha|$ and $|\beta| < \mu(R)$, nodes of coordinates α, β in the lattice $\Lambda(u_1, u_2)$ are strictly more than h hops away from the origin of the lattice. This check includes checking the validity of u_1 and u_2 themselves (cases $(\alpha, \beta) = (1, 0)$ and $(\alpha, \beta) = (0, 1)$)

Notice that for dense grids ($R \to \infty$), $\mu \to 1.15\ldots$. This small bound reduces the set of nodes to be checked in order to verify the validity of the coloring for given vector candidates.

In fact for $R > \frac{3\sqrt{2}}{3-\sqrt{3}}$, that is for $R > 3.3461$, we have $\mu < 2$, hence only 4 points need to be checked (considering symmetries): u_1 (with $\alpha = 1, \beta = 0$), u_2 (with $\alpha = 0, \beta = 1$), $u_1 + u_2$ (with $\alpha = 1, \beta = 1$), $u_1 - u_2$ (with $\alpha = 1, \beta = -1$).

However, Method VC1 is applicable for any radio range R, whereas Method VC2 requires $(R > \sqrt{2})$.

5.7 Complexity of VCM

We can now evaluate the complexity of *VCM* that lies in the generator vectors computation and in their validity check.

Theorem 7. *Depending on* VC *method,* VCM *complexity is in* $\Theta(R^6)$ *for Method VC1 and* $\Theta(R^4)$ *for Method VC2.*

Proof: The vector search space is in $\Theta(R^2)$ for each vector. The time complexity of the validity check is in $\Theta(R^2)$ for Method VC1 and $\Theta(1)$ for Method VC2. The theorem follows. ■

Since Method VC2 can be used as soon as $R > \sqrt{2}$, we deduce:

Corollary 3. *VCM complexity is* $\Theta(R^4)$

5.8 Summary: How to Apply VCM in Practice

In practice, to apply *VCM*, we start from a set of sensor nodes arranged as a two-dimensional lattice (identified by their integer coordinates). The input of the algorithm are R and h. Then, each node proceeds as follows:

1. Find the optimal valid vectors u_1 and u_2 using the Method OVS.

2. Each node computes its color using for instance the bijection defined in Section 5.5.1.2.

2.1. It computes the values (c_1, c_2) for each grid node in $\mathcal{P}(u_1, u_2)$ using System (5.13). Sorting the set of the values (c_1, c_2) yields an example of mapping between every possible value of (c_1, c_2) and a unique integer in $[0, |d| - 1]$.

2.2. Knowing its coordinates in the grid, each node deduces its two components c_1 and c_2

according to System (5.13) and then its color from the previous mapping.
We can notice that *VCM* allows each node to know its color in a single round.

5.9 Coloring Results with VCM

5.9.1 Examples of Vectors

Table 5.2: VCM vectors generating the optimal number of colors.

Radio range	2-hop coloring			3-hop coloring		
	v_1	v_2	colors	v_1	v_2	colors
1	(2,1)	(-1,2)	5*	(2,2)	(-2,2)	8*
1.5	(-3,0)	(0,3)	9*	(4,0)	(0,4)	16*
2	(3,2)	(-2,3)	13*	(4,3)	(-3,4)	25*
2.5	(4,3)	(-1,5)	23*	(5,5)	(-7,2)	45*
3	(5,3)	(-1,6)	33*	(7,5)	(-8,4)	68*
3.5	(5,4)	(-6,3)	39*	(8,5)	(-8,5)	80*
4	(7,3)	(-6,5)	53*	(8,8)	(-11,3)	112*
4.5	(9,2)	(-6,7)	75*	(13,3)	(-9,10)	157*
5	(9,4)	(-1,10)	94*	(14,4)	(3,15)	198*
5.5	(9,6)	(-1,11)	105*	(16,0)	(8,14)	224*
6	(11,4)	(-9,8)	124*	(17,4)	(-12,13)	269*
6.5	(13,1)	(-7,11)	150*	(-19,0)	(9,17)	323*
7	(10,9)	(-4,13)	166*	(15,13)	(-19, 7)	352*

Table 5.2 gives for different radio ranges two vectors v_1 and v_2 generating the optimal periodic pattern as well as the minimal number of colors obtained by a periodic pattern, for both a 2-hop coloring and a 3-hop coloring. The '*' symbol highlights the optimality of the number of colors used.

We observe that 2-hop coloring of the grid with radio range $R = 3$ is not equivalent to 3-hop coloring of a grid with $R = 2$ in terms of the number of colors (33 vs. 25). We conclude that the optimal number of colors is not determined only by the product $h * R$, but also the values of h and R taken separately.

5.9.2 Comparison with Other Methods

Table 5.3 depicts the simulation results obtained with *VCM* for various grids. The method is compared to OSERENA algorithm using line/column as priority assignment heuristics. Results are given for 3-hop coloring. For high radio range values, the number of nodes should be high enough to allow the reproduction of the color pattern.

We observe that *VCM* provides an optimal 3-hop coloring, for any radio range. This is not true for any other priority assignment tested. Moreover, the number of colors does not

Table 5.3: Number of colors obtained for 3-hop coloring.

Radio range	Grid size	Colors	
		VCM	OSERENA (line/column)
1	10x10	8*	8
	20x20	8*	8
	30x30	8*	8
1.5	10x10	16*	16*
	20x20	16*	16*
	30x30	16*	16*
2	10x10	25*	30
	20x20	25*	33
	30x30	25*	33
3	20x20	68*	80
	30x30	68*	83
3.5	20x20	80*	91
	30x30	80*	91

depend on the grid size, provided that the grid size is high enough to allow the repetition of the color pattern defined by the generator vectors.

5.10 VCM in Real Wireless Networks

So far, *VCM* has been described for grid topology since this topology is used by real applications. Notice however, that wireless communication may differ from what is expected by the theory that often uses simplified models: radio links may be asymmetric, a radio link may exist even if the remote node is at a distance higher than the transmission range R or conversely not exist even if the remote node is in the theoretical radio range. However, notice that a valid h-hop coloring for a given R, is also a valid h-hop coloring for $R' < R$ (although likely non-optimal). That is why, the first step in *VCM* is to select R such that two nodes that are at a distance greater than R are not neighbors (maybe using measurements or neighborhood detection).

Now, another real aspect in wireless ad hoc and sensor networks is the late arrival (mobility or in case of late start-up) and disappearance of nodes (a node is out of battery for instance). What is the impact of such impairments on *VCM*? We distinguish two categories:

1. Radio links disappearance: in this case, *VCM* always provides a valid coloring. The periodic coloring may still be optimal, as long as the percentage of missing radio links is below a given threshold L_1.

2. Radio links appearance: in this case, nodes that should not be neighbors (or heard nodes in case of asymmetric links) are. As a consequence, nodes having the same color may interfere

because of these additional radio links. The periodic coloring provided by *VCM* may still be perfectly acceptable by the application as long as the percentage of additional radio links is below a given threshold L_2.
As a further work, we will evaluate the thresholds L_1 and L_2 and study how random topologies can be mapped onto grids.

Notice that SERENA and OSERENA rely on the routing protocol to know the real neighbors of any node and do not require the use of the same radio range for all network nodes.

Conclusion

In this chapter, we have presented a new method called *VCM*, the Vector-Based Coloring Method, able to provide an optimal periodic h-hop coloring of any grid, with h an integer ≥ 1, for any radio range R. This method is easy to use: a single round is needed. It suffices to compute the two generator vectors. Knowing its coordinates within the grid, each node deduces its color from a simple computation. We have shown that this h-hop node coloring is optimal in terms of colors and rounds. We determined also an upper and a lower bound for the number of colors needed to color an infinite grid. *VCM* provides the optimal number of colors compared to all possible coloring including non periodic ones. Finally, we discussed how to apply *VCM* in real wireless networks. As future directions, we plan to extend *VCM* to random topologies. Our methodology encompasses the following steps: (1) coloring grids with holes, (2) coloring grids of cells, (3) using the first two steps, we map random topologies on grids and we color them with the optimized number of colors.

Chapter 6

Traffic Aware Time Slot Assignment in Data Gathering Applications

Introduction

A typical application in WSNs is data gathering. Sensor nodes are deployed in the region of interest to periodically collect and report sensed data to a sink node. To achieve this many-to-one communication, sensor nodes form a data gathering tree rooted at the sink. Nodes have different traffic demands. For instance, nodes close to the sink forward more data than leaf nodes in the data gathering tree. This is the "funneling effect" [128]. Consequently, nodes may need heterogeneous numbers of time slots. Indeed, two cases are possible.

1. **First case**: Nodes can aggregate data received from their children and are able to transmit their aggregated data in a single time slot. Consequently, all nodes need the same number of slots.
2. **Second case**: Data aggregation is not possible. We speak about *raw-data convergecast* because every packet is forwarded individually. Intermediate nodes in a data gathering tree simply apply the store and forward strategy, without processing the received packets. This may happen when for instance (a) the traffic demand is high, (b) the packets size is high, (c) or when nodes are supposed to transmit each packet individually without processing it. In this case, traditional TDMA-based MAC protocols tend to grant nodes equal channel access. This may lead to congestion, packet loss, and inefficient use of the bandwidth. Consequently, channel access should be proportional to the sensor demand.

In this chapter, we work under the assumptions of the second case to deal with the Time Slot Assignment problem, denoted *TSA* regarding the application requirements. Our aim is to build an efficient TDMA schedule of minimum length. We propose *TRASA*, *TRaffic Aware Slot Assignment algorithm* for WSNs. Assuming a sensor network where each node has a specific number of packets to transmit to its parent in the data gathering tree, *TRASA* assigns each node a number of slots proportional to its traffic demand, and schedules its activity. Moreover, *TRASA* allows the allocation of slots to nodes with heterogeneous demands (i.e

different numbers of packets that a node has to transmit like sensors of different types). Consequently, the algorithm addresses the funneling effect and ensures a fair medium access.

We start this chapter by discussing the applicability of the coloring in case of heterogeneous needs of medium access time in Section 6.1. In Section 6.2, we give a state of the art about traffic aware time slot assignment methods in data gathering applications. Then, we define in Section 6.3 the *TSA* problem and prove its NP-completeness in Section 6.4. In Section 6.5, we determine theoretical bounds on the optimal TDMA cycle length. In Section 6.6, we describe *TRASA* algorithm and present an evaluation of its performance.

6.1 Adaptation of the Coloring to Traffic Demand

So far, we have studied the scheduling algorithm when nodes can aggregate data, and we have proved that the coloring technique is an efficient method to build a TDMA schedule. The basic idea is to sort colors determined by the strategic mode of OSERENA according to the decreasing order, and for each color, we schedule nodes having this color. The question now is, based on these colors, how time slots can be efficiently assigned to nodes having heterogeneous traffic demands and when this demand may vary in time? We discuss 3 intuitive solutions and compare their performance on the network of Figure 6.1 where each node generates only one packet and has to forward packets of its children (integers in this figure represent the colors).

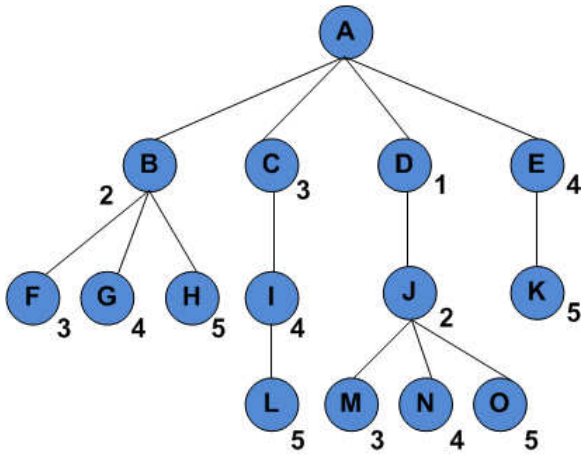


Figure 6.1: Example of a network colored via OSERENA.

6.1.1 Solution 1: Larger Slots

The first intuitive solution is to build a TDMA cycle of c superslots, where c is the total number of colors. Besides, each superslot is composed of s slots where s is the highest traffic demand of nodes scheduled during any superslot. Applying this solution to network of Figure 6.1, node D has the highest subtree size (4 descendants), so what we need is a superslot of 5 slots. The corresponding slot assignment is given by Figure 6.2 and contains 25 slots.

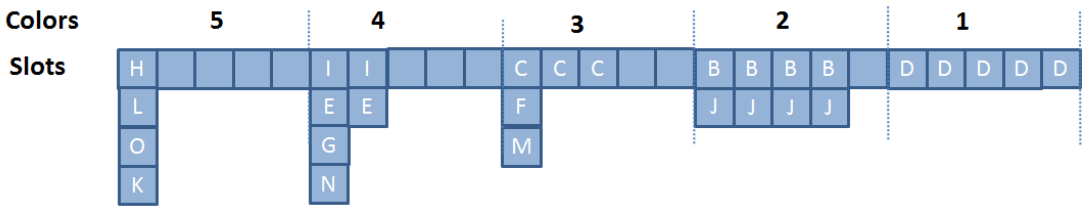


Figure 6.2: Slot assignment based on Solution 1.

We notice also that only 15 slots are busy among the 25 slots. This means that such a solution does not ensure an optimal bandwidth use, especially when few nodes have high traffic demand.

6.1.2 Solution 2: Heterogeneous Number of Granted Slots

A first enhancement of the first solution is to have superslots with heterogeneous sizes. Each superslot contains a number of slots equal to the highest traffic demand of nodes scheduled during this superslot. This leads to the scheduling depicted in Figure 6.3.

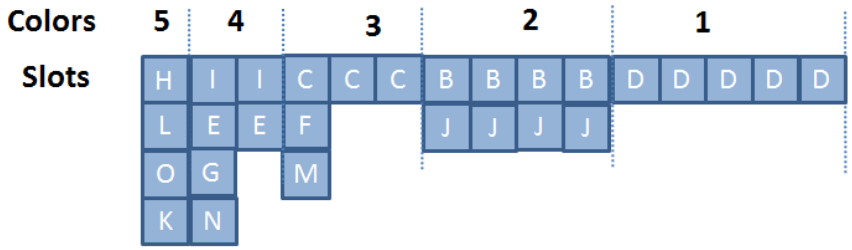


Figure 6.3: Slot assignment based on Solution 2.

This solution produces 15 slots which are all busy. This constitutes an enhancement of Solution 1 but still not sufficient as this solution does not provide an optimal spatial reuse of the bandwidth.

6.1.3 Solution 3: Multiple Colorings

In this solution, while there are nodes having packets to transmit, we repeat the following operations: (1) color the nodes having data to transmit and sort these colors according to the decreasing order, (2) match each color to one slot, (3) schedule nodes in these slots.

Applying this algorithm on the example of Figure 6.1 gives the scheduling of Figure 6.4.

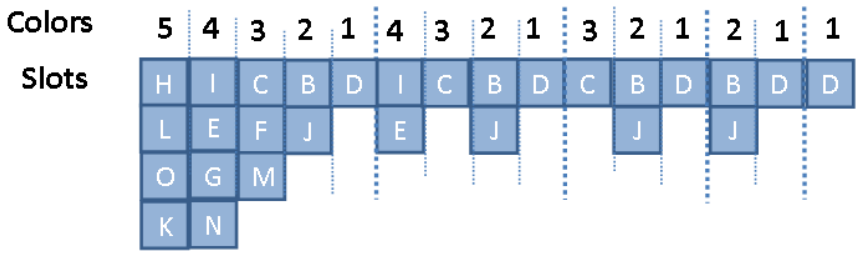


Figure 6.4: Slot assignment based on Solution 3.

This solution produces 15 slots like Solution 2. Notice that the slots allocated to a node are not contiguous. However, the drawback is the induced overhead of recoloring the network. There are k coloring where k is the highest slot demand in the network.

Moreover, we compare Solution 2 and Solution 3 for the case where nodes have 1 or more initial packets to transmit: [B:3,C:1,D:1,E:1,F:2,G:1,H:1,I:2,J:2,K:1,L:1,M:1,N:1,O:2].
Results are: 22 slots for Solution 2, and 23 slots for Solution 3. This difference can be neglected. However, as nodes in Solution 3 has non contiguous slots, realizing a duty cycle implies that they should perform more than awake/sleep switch like in Solution 1.

From the above study, we can draw the following remarks. A color is assigned to all non interfering nodes, regardless the heterogeneity in their traffic demand. For instance, two nodes with packet demand respectively 10 and 2 are scheduled simultaneously. Thus, in 8 slots, there is only one transmitter. That is why, mapping a color to one or multiple slots usually may lead to slot underuse. Moreover, when the traffic varies in time, the coloring must be updated. For these reasons, we do not perform coloring, and we design a time slot assignment algorithm that is aware about the traffic. We first give a state of the art about these algorithms.

6.2 State of the Art: Traffic Aware Time Slot Assignment

Despite the existence of a variety of scheduling schemes, few of them allocate a number of slots proportional to node demand. In this section, we present a state of the art of existing schemes classified depending on their awareness of the traffic demand. This classification is illustrated in Figure 6.5.

◦ **Protocols that do not depend on the traffic demand, or assume that nodes aggregate all the data they have to transmit in a single time slot**:
With these protocols, any node receives exactly one slot. It is the class of scheduling algorithms already presented in Chapter 2.

◦ **Traffic aware protocols:**
In these protocols, the number of slots received by a node depends on its traffic demand. We

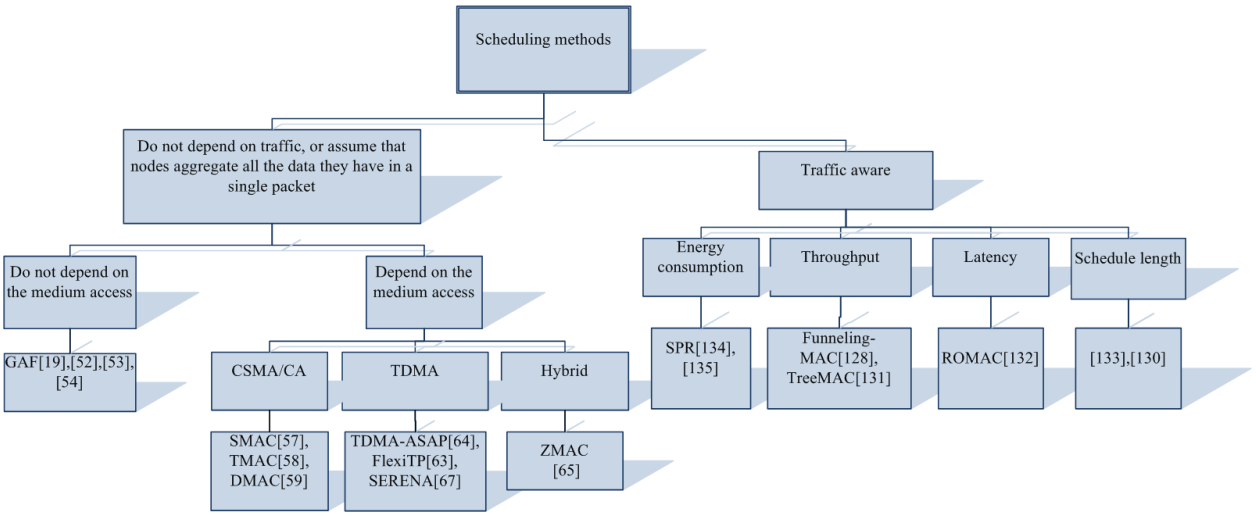


Figure 6.5: Taxonomy of time slot assignment techniques.

present some examples of TSA algorithms classified according to their main objective. Notice that one algorithm may satisfy more than one objective.

1) Maximizing throughput of received data at the sink:

In [128], Gahng et al. presented Funneling-MAC to mitigate the funneling effect and boosting application fidelity in WSNs. Funneling-MAC is hybrid, TDMA is used within a small number of hops from the sink (called the intensity region), and CSMA/CA is used elsewhere. The sink is responsible of the scheduling of the nodes in this intensity region. Funneling-MAC allows the slot reuse between nodes distant more than 2 hops away.

The main goal of TreeMAC [131] is to achieve high throughput in real-time high-data-rate WSNs. In TreeMAC, the TDMA cycle in divided into frames, and each frame is composed of 3 slots. Slots are assigned from the root to the leaves: each node assigns slots to its children such that they do not have the same frame. This is not suitable for large-scale networks. Further, this algorithm allows a slot reuse between nodes belonging to the same tree branch, but not between the subtrees. Each node has a pseudo-level equal to its distance to the sink minus 1 modulo 3. Hence, the number of slots is equal to $3(N-1)$, where N is the number of nodes. TreeMAC achieves at least one-third of the optimum throughput assuming reliable links. From energy point of view, authors show that TreeMAC outperforms CSMA and Funneling-MAC [128]. However, nodes in TreeMAC are assigned disjoint time slots for transmission. This means that if the node can sleep between its transmitting slots, it should awake up many times to transmit and receive data. The transitions between radio states are energy-consuming, so should be reduced.

2) Minimizing latency:

ROMAC [132] is a localized MAC protocol that targets data reliability and timeliness in sensor networks. Like TreeMAC, ROMAC divides the TDMA cycle into frames, each frame being composed of three slots, and allocates slots to nodes proportionally to their traffic demand.

Compared to TreeMAC, ROMAC is more localized: Each node can locally determine its frame and its time slots without relying on its parent like in TreeMAC. The frame node with identifier i transmits its own data in the frame i, and transmits the packets of any node j in its subtree in the j^{th} frame. Unlike TreeMAC and Funneling-MAC, ROMAC automatically adapts to routes changes, any node updates its set of frames every TDMA cycle. Consequently, ROMAC achieves lower delivery latency than these two protocols as the network size increases.

3) Minimizing schedule length:

Incel et al. [133] aimed at reducing the delays of data collection by minimizing the schedule length. They studied scheduling nodes where each node generates a packet at the beginning of the TDMA frame. In this work, only tree interference links are considered. Authors proved that the lower bound on the schedule length is given by: (1) the maximum node degree when packet aggregation at each intermediate node is considered, and (2) $max(2n_k - 1, N)$ where n_k is the maximum number of descendants of the sink children, otherwise. For this second case, authors proposed *Local-TimeSlotAssigment* algorithm in which the sink schedules an edge having the highest remaining number of packets. Furthermore, any parent node with an empty buffer selects one child whose buffer is not empty at random respecting tree interfering links. Consequently, they ensure parallel transmissions among multiple branches of the tree, and keep the sink busy in receiving packets. In [130], authors proposed algorithms based on coloring. Two centralized solutions are described. First, in the node-based scheduling, any slot is shared between nodes with the same color and any other node that does not conflict with them. Second, in the level-based scheduling, conflicts are defined between levels: the same color is assigned to two levels if they do not contain any couple of conflicting nodes. For each color, non conflicting nodes from the levels having this color share the same slot. Besides other nodes that do not conflict with already scheduled nodes are scheduled simultaneously. This algorithm suffers from the energy waste because of the radio switches between active/sleep states.

4) Minimizing the energy consumption:

In [134], Turau et al. proposed SPR, to schedule each path in the routing tree separately. Indeed, SPR considers the routing tree as an overlay of the paths from each leaf to the sink. The slots assigned to any node are the union of its slots in each path. Hence, the spatial reuse of time slots is restricted to a common path, which makes SPR not efficient in terms of schedule length for networks where the average number of children is high. Another example in this category is given in [135] where the energy efficiency is achieved by reducing the number of switches between the active and sleep states.

In most applications, there is not always a single objective, but often multiple and sometimes conflicting objectives. Some examples include minimizing communication cost and delay, minimizing energy consumption and completion time of data collection, maximizing capacity and minimizing energy, etc. Scheduling algorithms need to address the optimal

trade-offs in satisfying these conflicting objectives. In our work, the primary goal is minimizing the schedule length. Other performance criteria are also taken into account. In the next section, we will present a formal definition of the *TSA* problem.

6.3 The Time Slot Assignment Problem

In this section, we define the Time Slot Assignment problem, denoted *TSA*.

6.3.1 Assumptions

◦ **A1. Data gathering applications and sink tree:** In data gathering applications, a node, called sink or gateway, is in charge of collecting data sent by all other nodes. Hence, the typical traffic pattern is many-to-one routing and leads to a spanning tree T rooted at the sink node.
◦ **A2. Application data:** In each data gathering cycle, each node except the sink has its own data to transmit to its parent in addition to the data received from its children. Some nodes (for example, the children of the sink), need more than one slot to transmit their data.
◦ **A3. Time slot:** The time slot duration must allow the transmission of at least one packet.
◦ **A4. Conflicting nodes:** Two nodes are said *conflicting* if and only if they cannot transmit in the same time slot. For any given node u, the set of nodes conflicting with u is an input for the time slot assignment problem.
◦ **A5. No message loss and no node failure.**

6.3.2 Problem Statement

The time slot assignment problem, *TSA*, under the assumptions introduced in Section 6.3.1, consists in assigning slots to network nodes, such that no two conflicting nodes are scheduled in the same slot while minimizing the schedule length. Besides, this scheduling must ensure that each node transmits towards the sink, both its own packets and the packets generated in its subtree. To summarize, our aim is to build a minimal valid scheduling.

Definition 11 (**Valid scheduling**). *A scheduling is said valid if and only if:*

- *any node is assigned a number of slots sufficient to transmit all its traffic.*
- *any node is assigned a slot if and only if it has data to transmit during this slot.*
- *any two conflicting nodes do not transmit in the same time slot.*

Definition 12 (**Minimal scheduling**). *A valid scheduling is said minimal if and only if no other valid scheduling has a smaller number of time slots.*

In our work, we target **the minimization of the number of slots in the TDMA cycle** as a primary goal because it contributes to improve the network performances. First, the

maximum packet delay being equal to one TDMA cycle, reducing the TDMA cycle duration helps to meet the time constraints of packets delivery. This property is crucial for applications with strong time constraints. Second, the throughput measured at the sink is the number of slots granted to the children of the sink to send their packets divided by the TDMA cycle duration in slots and multiplied by the network capacity. Decreasing the TDMA cycle by a factor multiplies the throughput by the same factor. Third, given a TDMA cycle composed of activity and inactivity periods, minimizing the schedule length reduces the activity period of nodes and allows them to save more energy.

6.4 Complexity of TSA Problem

It has been proved in [129] and [130] that the *TSA* problem is NP-complete. In this chapter, we generalize the study and prove that the h_TSA problem is NP-complete for any positive integer h. The problem h_TSA is defined as follows.

Definition 13. *h_TSA consists in assigning time slots to nodes such that no two nodes that are less than or equal to h hops away are scheduled simultaneously, while minimizing the number of slots used.*

Theorem 8. *The h_TSA problem, for any positive integer $h \geq 1$ is NP-complete.*

Proof: The decision problem of *h_TSA* is given by: *Can nodes in a graph G be assigned S time slots (S is a positive integer) during which they can transmit their data to a sink node, such that any two nodes that are p-hop away, with $1 \leq p \leq h$, are not assigned the same time slot?*
To prove that the decision problem of *h_TSA* is NP-complete, we use the knowledge that the h-hop coloring problem in the strategic mode is NP-complete (proved in Chapter 3). We note *h_Color* this problem and recall that it is defined as: *coloring a graph with the smallest number of colors (a color is represented by an integer) such that any two nodes that are p-hop away with $1 \leq p \leq h$, do not have the same color. In addition, the color assigned to any node is smaller than the color assigned to its parent in the data gathering tree.*
We need to prove that finding a solution to *h_TSA* is equivalent to finding a solution to *h_Color*.

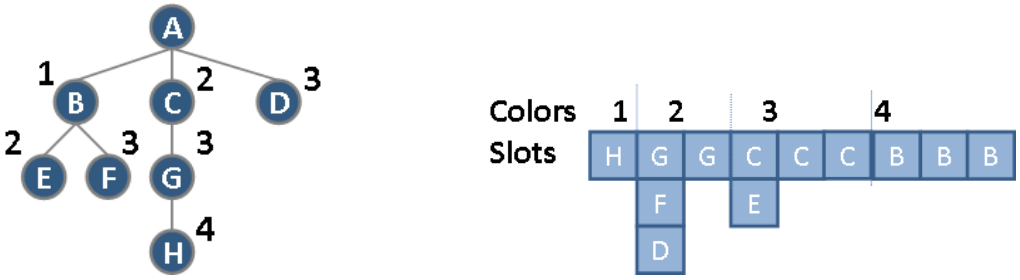
Let G be a connected, undirected graph, and T its spanning tree. Notice that the construction of T can be done in polynomial time, so it does not add a high complexity. Each node u in G has traffic demand d_u. Let C_G be a coloring of G solving the *h_Color* problem and using the colors $c_1, c_2, \ldots, c_{max}$. Let N_i be the set of nodes having the color c_i. We can build a slot assignment for nodes in G as follows. We sort the colors by decreasing order. For each color c_i, we add to the cycle a number of slots equal to the maximum traffic demand of nodes in N_i, denoted max_{di}. Then, nodes in N_i are scheduled during these slots, each one has a number of slots equal to its traffic demand. Consequently:

1. This scheduling is conflict-free, nodes that are scheduled simultaneously have the same color, and hence do not interfere.

2. The scheduling allows each node to transmit all the data it has since it is assigned a contiguous number of slots equal to its demand. Besides, this scheduling ensures that each parent node accesses the medium after all its children, because it has a higher color than them, and slots are assigned to nodes having the smallest colors first.

3. The number of slots used is equal to $S = \sum_{i=1}^{c_{max}} max_{di}$.

Figure 6.6 illustrates the proof. Figure 6.6(a) is an example of a colored tree, and Figure 6.6(b) is the corresponding time slot assignment.



(a) Example of colored tree.

(b) The corresponding time slot assignment.

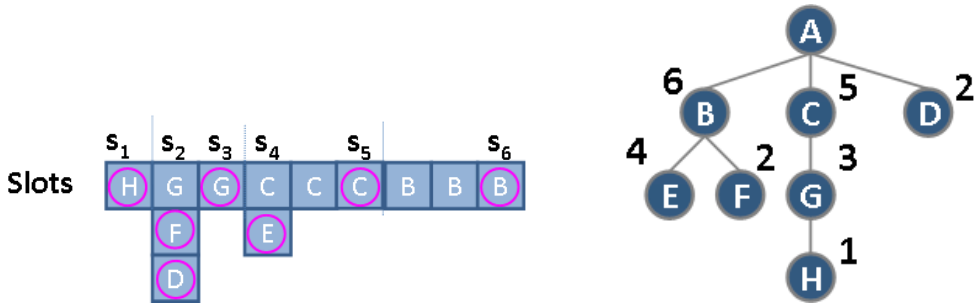
Figure 6.6: Example illustrating the TDMA cycle construction based on the coloring.

Inversely, given a scheduling of nodes in G, composed of S slots, we need to color the graph respecting the principles of the *h_Color* problem. We denote $S_s = s_1, s_2, \ldots, s_s$ the sequence of slots such that during each slot s_i at least one node is scheduled for the last time in the TDMA cycle (like circled nodes in Figure 6.7). Then, for each slot s_i in S_s we assign the color i to all uncolored nodes scheduled for the last time in s_i. Consequently, we get a number of colors bounded by the number of slots. The same color is assigned to nodes scheduled in the same slot, so the coloring is collision-free. Further, since the last slot of each parent cannot be scheduled before the last slot of any of its children, each parent has a color greater than the color of its children.
Hence the theorem.

■

6.5 Theoretical Bounds on the Number of Slots

In this section, we focus on theoretical lower bounds of cycle length. As we will see, these bounds allow us to measure the distance between the number of slots in *TRASA* and the optimal number of slots. We focus on three different topologies: linear, multi-linear and tree. The two first ones are special topologies of the third one and are representative of WSNs



(a) The node schedule. (b) The corresponding colored tree.

Figure 6.7: Example illustrating the colors assignment based on the time slot assignment.

deployed in confined areas such as the airplane fuselage (linear) or mines with several galleries (multi-linear).

6.5.1 Additional Assumptions

For simplicity reasons, we assume that in each time slot the transmitter sends only a single packet. Nodes are randomly deployed in the 2-dimensional plane. Two nodes u and v are 1-hop neighbors if and only if their distance is lower than or equal to the transmission range R. For any integer $h > 1$, any two nodes u and v are h-hop neighbors if and only if u is $(h-1)$-hop away from a 1-hop node of v.
In this chapter, we consider that interferences are limited to 2 hops. Consequently, we assume that any two nodes u and v within 2-hop neighborhood from each other do not transmit in the same time slot. Based on this assumption, we can define the label of a node.

Definition 14 (**Node label**). *For any network node u different from the sink, we define label(u) = (distance(u,sink)-1) modulo 3.*

In line networks, the assigned labels (starting by the sink child) are respectively 0,1,2,0,1,2,etc. It follows that only nodes having the same label can be assigned the same slot.

6.5.2 Number of Slots in Linear Networks

Theorem 9. *In linear networks, a lower bound on the number of slots is Max($N-1$, $3N-6$), where N is the number of nodes including the sink.*

Proof: Consider a linear network with $N \geq 1$ nodes, where u_0 is the sink node and any node u_i is at a distance $i < N$ from the sink. It is clear that for $N = 1$, 2 or 3 the theorem is true. Now let us assume $N \geq 4$. Assuming each node needs one slot to transmit its own data, the sink needs $N-1$ slots to receive data from all nodes. Besides, let u_1, u_2 and u_3 the three closest nodes to the sink. No two nodes among these three nodes can transmit data simultaneously. Consequently, the number of slots occupied by these nodes is the sum of the number of packets they have to transmit, that is $(N-1)+(N-2)+(N-3)$. Hence the theorem. ■

6.5.3 Number of Slots in Multi-line Networks

Theorem 10. *In multi-line networks, a lower bound on the number of slots is Max($N-1$, $3n_k-3$), where N is the number of nodes including the sink, and n_k is the maximum number of nodes in a line starting with a child of the sink (this child is not included).*

Proof: The sink requires $N-1$ time slots to receive all the packets generated in the network. Moreover, from Theorem 9, at least $3(n_k+1)-6 = 3n_k-3$ slots are required to transmit data to the sink. Hence the theorem. ■

6.5.4 Number of Slots in Tree Networks

Theorem 11. *In tree networks, a lower bound on the number of slots is Max($N-1$, $2n_k-1$, $3n_j-3$), where N is the number of nodes including the sink, n_k is the maximum number of nodes in a subtree rooted at a child of the sink and n_j is the maximum depth of nodes.*

Proof: The sink requires $N-1$ time slots to receive all the packets generated in the network. Moreover, let us consider u_k the child of the sink with the highest number of descendants. Let n_k-1 be this number. At least n_k slots are needed by u_k to transmit its packets and at least n_k-1 slots are needed by u_k to receive the packets from its children. Since all these transmissions are sequential, at least $2n_k-1$ slots are needed. If now we consider the longest line in the network. Let n_j be the depth of the deepest node. According to Theorem 9, at least $3n_j-3$ slots are needed. Hence, the number of slots is at least $\max(N-1,\ 2n_k-1,\ 3n_j-3)$. ■

6.6 TRASA: TRaffic-Aware Time slot Assignment

The main objective of TRASA algorithm is to achieve a time minimal scheduling while ensuring a fair medium access where any node is granted a number of slots proportional to its packet demand.

6.6.1 Principles

TRASA is based on the following rules:

1. Any node has a *priority* and a set of conflicting nodes.
2. Nodes compete for the current time slot if and only if they have data to transmit.
3. For any slot, the first scheduled node is the node having the highest priority among all the nodes having data to transmit.
4. Any node can be scheduled in any time slot if it does not interfere with nodes already scheduled in this slot.

6.6.2 Algorithm Presentation

In this section, we present a centralized version of TRASA given by Algorithm 1.

Algorithm 1 TRASA algorithm.

1: **Input**: a spanning tree T, where each node u has d_u packets to transmit and a set of conflicting nodes *Conflict(u).*
2: **Output**: The scheduling of nodes in the TDMA cycle
3: $t = 1$ /* current time slot */
4: **while** $\sum_u d_u$ **do**
5: N = List of nodes having data to transmit sorted according to their priority
6: u = node with the highest priority in N
7: **if** "oneSlot" **then**
8: $nbSlot = 1$
9: **end if**
10: **if** "manySlots" **then**
11: $nbSlot = d_u$
12: **end if**
13: **while** $N \neq \emptyset$ **do**
14: u=node with the highest priority in N
15: $nbAssignSlot = \min(d_u, nbSlot)$
16: assign slots t to $t + nbAssignSlot - 1$ to node u
17: d_u -= $nbAssignSlot$
18: $d_{parent(u)}$ += $nbAssignSlot$
19: $N = N \setminus (\{u\} \cup Conflict(u))$
20: **end while**
21: $t+ = nbSlot$
22: **end while**

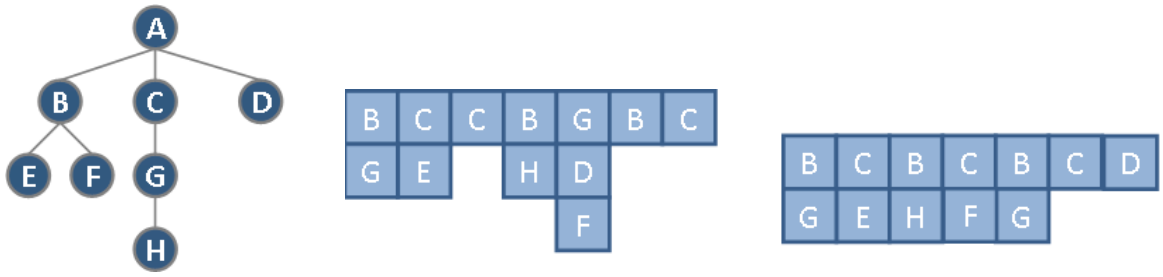
The algorithm iterates over N the set of nodes having data to transmit and sorted according to their priority. In each iteration, the algorithm determines the set of nodes scheduled in the current time slot starting at t, and the number of slots allocated to each of them. The node u with the highest priority is scheduled first (line 6). Further, any other node in the sorted set N is given the same time slot if and only if it does not conflict with nodes already scheduled in this slot (see the while loop of line 13). TRASA ends when all packets generated in the network are transmitted to the sink.
Two versions of TRASA are simulated. Indeed, at any iteration, when a node is scheduled, it is allocated either:
◦ only one time slot: this version is denoted *oneSlot*;
◦ as many time slots as required by the node with the highest priority: this version is denoted

manySlots. Intuitively, the *manySlots* version allows a node to transmit its packets successively, avoiding the energy and time wasted in switching between the sleep and the awake states.

Concerning the definition of the priority, we evaluate TRASA for two heuristics:
∘ *prio=descNb*: The priority of any node is given by its number of descendants. Intuitively, a node with a high number of descendants will have a high number of packets to transmit.
∘ *prio=remPckt*parentDem*: *remPckt* means the number of packets the node has in its buffer at the current iteration. *parentDem* is the total number of packets the parent of the node has to receive in a cycle. The idea behind this heuristic is to reduce the number of buffered packets by favoring nodes having packets to transmit to a parent with a high number of descendants.

6.6.3 Example of TRASA Slot Assignment

Figure 6.8 illustrates an example of TRASA coloring applied on the tree depicted in Figure 6.8(a). Both *manySlots* and *oneSlot* version are applied. We notice that, for this example, the number of time slots is the same for both versions.



(a) Example of a graph. (b) TRASA manySlots version. (c) TRASA oneSlot version.

Figure 6.8: Example of TDMA schedule obtained by TRASA.

6.6.4 Properties: Bounds on the Number of Slots

In this section, we present the theoretical properties of TRASA with $prio = descNb$, assuming that:

A6. Each source node generates exactly one packet per TDMA cycle.

A7. For any node u, the only nodes conflicting with u are its parent, its children, its grandparent, its brothers and its grandchildren.

6.6.4.1 TRASA for Linear Networks

Property 8. *Applied to a linear network of N nodes, TRASA schedules these nodes according to the following sequence of labels: (010)-(210)*. That is: (1) All nodes with label* 0 *are scheduled simultaneously, followed by all nodes with label* 1*, etc. (2) The sequence (210) is repeated a number of times equal to N div* 3*, where div is the integer division operator.*

Proof: In TRASA, nodes are scheduled in the order of their number of descendants. Hence, all nodes with label 0 do not conflict according to assumptions $A2$, they are assigned the first time slot. Then, nodes with label 1 have the second highest priority and still have their own packet to transmit, they are assigned the second time slot and transmit their packet to their parent which has the label 0 by definition. Consequently, nodes with label 0 have a packet to transmit, they will occupy the next time slot. After that, nodes with label 2 have the highest priority among nodes having data to transmit, they are scheduled in the fourth slot. After this transmission, the parents of these nodes, which have label 1 have a packet to transmit, since they have the highest priority, they occupy the fifth time slot. Then, their parents having label 0 have a packet to transmit, since they have the highest priority, they occupy the sixth time slot. In that way, data progress towards the sink. This last sequence 210 is repeated until all nodes have transmitted all the data they have. It is clear that the sequence 210 is repeated a number of times equal to the number of nodes with label 2 and hence to N *div* 3. ■

Theorem 12. *TRASA ensures a* ***time optimal*** *scheduling using* $Max(N-1, 3N-6)$ *slots for any linear network of* N *nodes.*

Proof: Consider a linear network of N nodes, where u_1, u_2, u_3 are the three closest nodes to the sink u_0. From property 8, each time slot is occupied by one of these nodes. Consequently, any other node u_i can be scheduled with one of these nodes depending on its label. It means that the number of slots in the TDMA cycle is equal to the number of slots required by u_1, u_2 and u_3 which is $(N-1)+(N-2)+(N-3) = 3N-6$. We have proved in Theorem 9, that any valid scheduling requires at least $Max(N-1, 3N-6)$ slots. As a consequence, TRASA that reaches this lower bound is optimal for linear networks. ■

6.6.4.2 TRASA for Multi-line Networks

Property 9. *For any multiline network, let* u_i *be the child of the sink with the highest number of descendants denoted* n_i*, TRASA requires at least* $Max(N-1, 3n_i-3)$ *slots if there is no other child of the sink with the same number of descendants* n_i *and* $Max(N-1, 3n_i-2)$ *otherwise.*

Proof: Let u_i be the child of the sink with the highest number of descendants denoted n_i. Let u_j be the child of the sink with the second highest number of descendants denoted n_j. Since nodes u_i and u_j are 'brothers' and u_i has a priority higher than u_j, u_i will occupy the first slot in parallel with any descendant of u_i having label 0 and any descendant of u_j having label 1. The second slot will be assigned to any descendant of u_i having label 1 and any descendant of u_j with label 0, etc. It results that the TDMA cycle has the label sequence *(010)-(210)** for the descendants of u_i and the label sequence *(102)-(012)** for the descendants of u_j. Similarly to Property 8, the sequence *(012)* relative to u_j is repeated n_j *div* 3 times. Consequently:
◦ If the two branches contain the same number of nodes, the node u_j will send its last packet

after the node u_i and hence, one additional slot to $3n_j - 3$ is required.
∘ If $n_i > n_j$, all descendants of u_j will be able to share slots with those used by the descendants of u_i. ■

Theorem 13. *Applied to a multi-line network, the number of slots n used by TRASA verifies:* $n \geq Max(3n_k - 3, N - 1)$*, where N is the total number of nodes including the sink and* n_k *is the highest number of descendants of the sink children.*

Proof: Assuming each node has only one packet to transmit and hence requires one time slot, at least $N - 1$ slots are needed by the sink to receive data from these nodes. From Theorem 12, if we consider the child of the sink with the highest number of descendants n_k, $3(n_k + 1) - 6$ slots are needed to schedule nodes on this branch. ■
Notice here that since the children of the sink cannot share the same slot, the cycle length is strictly higher than this bound in some scenarios as explained in Property 9.

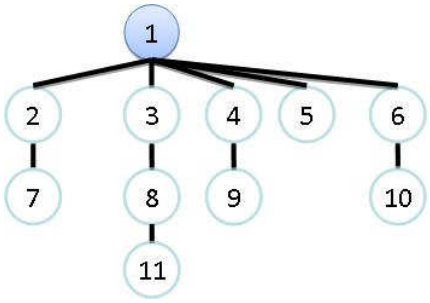
6.6.5 Performance Evaluation

6.6.5.1 Comparison with the Optimal Results

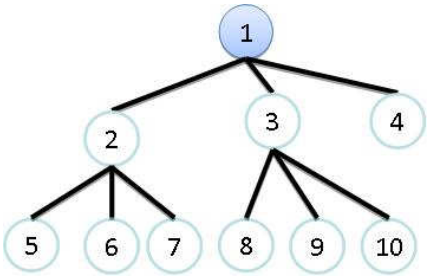
In [136], we worked with colleagues who provided a formulation of *TSA* problem as an Integer Linear Programming (ILP) optimization problem. Then, based on the proposed model, they obtained an optimal time slot assignment using the GLPK solver (GNU Linear Programming Kit) [137]. However, for small problem sizes (few nodes, each source generates a single packet), results are obtained within an acceptable duration of time. Nevertheless, when the WSN becomes large and even for moderate network sizes (e.g. 30 nodes) and simple topologies (e.g multiline networks), the time required to compute the optimal solution can reach one day.

Figure 6.9 illustrates the number of slots and the maximum size of buffers obtained by the model and TRASA with its two heuristics *prio=descNb* and *prio=remPckt*parentDem*. The caption of each subfigure follows the pattern $\{Sa\}\{SbBb\}\{ScBc\}$, where Sa stands for the optimal number of slots, Sb and Bb (Sc and Bc respectively) are the number of slots and the maximum number of buffers obtained by TRASA with $prio = descNb$ ($prio = remPckt * parentDem$ respectively). Notice that the TRASA algorithm provides the optimal number of slots in all these topologies.

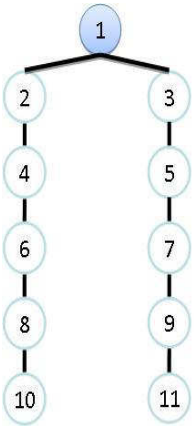
Besides, for tree topologies, we have proved in Theorem 11 that the lower bound on the number of slots is the maximum of three terms. For each of them, we can find a topology such that TRASA reaches this term. Hence, TRASA is optimal on these topologies. For example, with a branch factor of 3 and 20 nodes, we get $2n_k - 1$= 2*11-1=21 slots, which is optimal according to Theorem 11. For 50 nodes and a branch factor of 3, we get 49=$N - 1$ slots, which is optimal according to Theorem 11. For 10 nodes where a sink child is the head of a line with 4 nodes and the other sink child has 3 children, TRASA achieves the optimal value by reaching the lower bound of $3n_j - 3$= 3*5-3=12 slots.



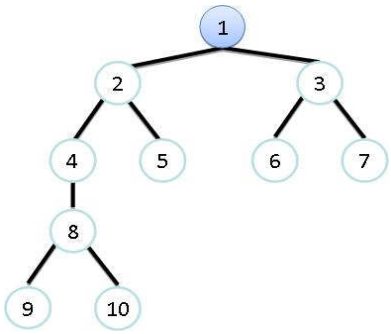
(a) $\{S10\}$ $\{S10B2\}$ $\{S10B2\}$



(b) $\{S9\}$ $\{S9B2\}$ $\{S9B2\}$



(c) $\{S13\}$ $\{S13B3\}$ $\{S13B3\}$



(d) $\{S13\}$ $\{S13B3\}$ $\{S13B2\}$

Figure 6.9: Examples of tree and multi-line topologies

6.6.5.2 Simulation Results

We developed a Java based simulation tool and performed simulations with the two versions and two heuristics of TRASA. We compare the TRASA performance with a slot assignment where the priority is given by the number of remaining packets, denoted *remPckt* (i.e the number of packets present in the buffer of the node considered). We generate random graphs deployed in a given area ($100m$x$100m$), where the number of nodes ranges from 20 to 100. We build trees where the maximum number of children is 3. Unlike the previous section where the only existing links are those in the tree, we assume that a link exists between two nodes if and only if their distance is less than or equal to the radio range ($30m$). Consequently, additional links to the tree links are considered. In the following, each result is an average of 20 runs for small topologies and 50 runs for large topologies.

We first evaluate the total number of slots for the two heuristics of TRASA (see Figure 6.10). Both heuristics of TRASA give the same number of slots, and outperform the heuristic *remPckt*. This result justifies the heuristic $remPckt * parentDem$ that takes into account not only the number of remaining packets but also the number of packets received by the parent. Further, the slot number is not impacted by the number of slots assigned to any node in each iteration: *OneSlot* and *manySlots* versions of TRASA use almost the same

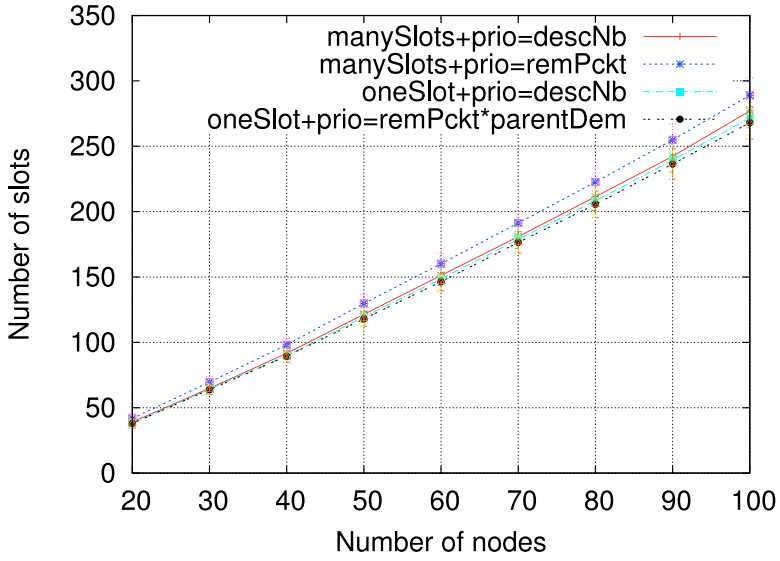
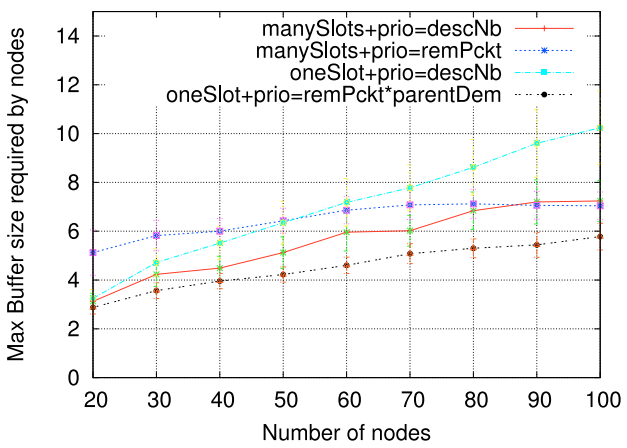


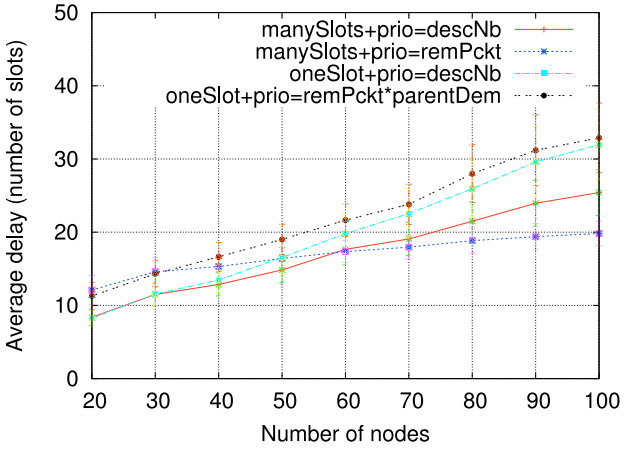
Figure 6.10: The number of slots.

schedule length.

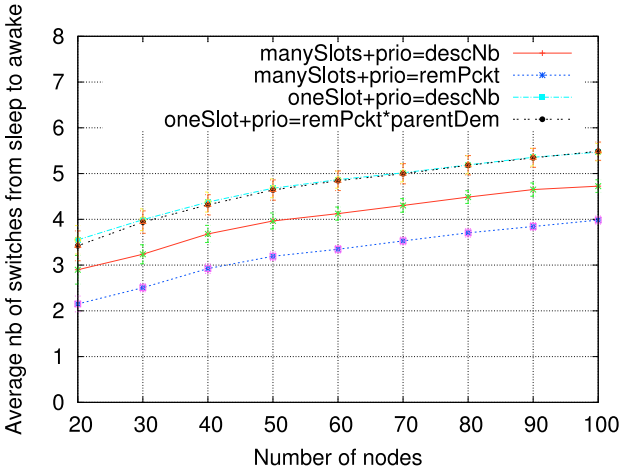
Simulation results show that $OneSlot + remPckt * parentDem$ ensures the smallest buffer size as illustrated in Figure 6.11(a) which is explained by the pipeline effect favored by this heuristic. The opposite of one might think, assigning the highest priority to a node having the highest number of remaining packets does not accelerate the buffer release, as illustrated by the result of $manySlots + remPckt$. We define the average data delivery delay as the average number of slots that one packet takes to reach the sink once it is transmitted by its source. Figure 6.11(b) shows that TRASA achieves the smallest delays for the heuristic $manySlots + descNb$. This is explained by the fact that if the priority is given by the number of descendants, nodes close to the sink have the highest priority, and hence the probability that the sink receives data in a time slot is high, which reduces the data gathering delays. In addition to maximizing the sleep duration, reducing the radio state switches contributes also in energy saving as these switches are energy consuming. Figure 6.11(c) shows that, as expected, the $manySlots$ version allows to reduce the number of radio state switches since nodes are allowed to send their packets in consecutive time slots.



(a) Maximum buffer size.



(b) Average delay.



(c) Number of radio switches.

Figure 6.11: TRASA performance.

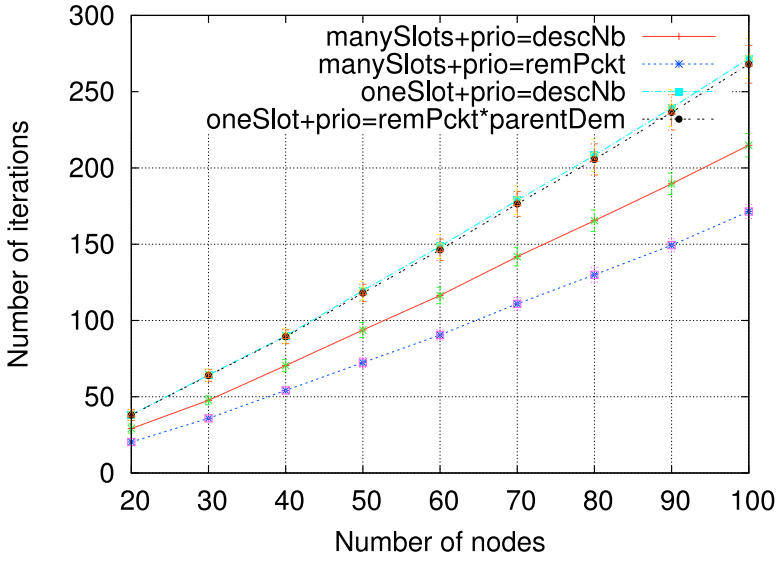


Figure 6.12: The number of iterations.

Similarly, the number of iterations is reduced with the *manySlots* version as illustrated in Figure 6.12.

6.7 Conclusion

In this chapter, we focused on data gathering applications which are a typical type of applications supported by WSNs. Assuming a slotted medium access, we investigated the raw convergecast problem looking for a minimal schedule length. A smaller schedule length improves the end-to-end delays and reduces the energy consumption. We proved that the *h-TSA* problem is NP-complete. We focus more particularly on specific topologies such as linear or multi-linear which are well adapted to confined environments and compute lower bounds for the time slot assignment problem. We proposed a traffic aware time slot assignment algorithm called *TRASA*. We proved that *TRASA* is optimal for linear topologies. Furthermore, applying *TRASA* to particular different tree topologies, we show that the optimal schedule length is reached. *TRASA* heuristics outperform solutions only based on the number of remaining packets. As a perspective for this work, it is judicious to dynamically adapt the schedule of nodes to occasional traffic demands, that is when some or many nodes require more time slots for a specific number of cycles. Another perspective would be to design a distributed version of *TRASA*.

Chapter 7

Conclusion and Perspectives

7.1 Conclusion

Sensor networks offer a powerful combination of distributed sensing, computing and communication. They support countless applications and, at the same time, pose numerous challenges due to their peculiarities. Primary issue is the power scarcity to which battery-operated sensors are typically subject. Thus, **optimizing energy consumption and bandwidth use** constituted the main objectives of our work. Realizing these objectives required overcoming practical problems that we faced in the OCARI project.

Among the many techniques identified in the literature, we focused on **node activity scheduling** that allocates time slot to nodes. Energy efficiency is achieved by avoiding collisions and allowing nodes to sleep as long as possible. To achieve this objective, the adopted strategy in our work is to design **auto-adaptive protocols and algorithms**. Indeed, to obtain high performance degree, any designed protocol cannot neglect the type of the application, the unreliable nature of wireless links, etc. In our work, this adaptivity that can be obtained by cross layer concerns:

- **The type of the application and communications supported**. To schedule the nodes activities, we use the graph coloring technique where a color is shared between non interfering nodes and mapped to a time slot. We define two modes of coloring: the first for the general applications and the second for data gathering applications. We proved that being aware of the types of communications is very beneficial: algorithms pay only what the application needs. More precisely, a careful study of possible interference cases taking into account the types of communications supported allows us to improve the parallelism in the transmissions. Thus, less colors are needed and data transmission delays may be decreased.

- **The density of nodes**. Scalability of protocols running on dense WSNs is challenging because of the limited memory and bandwidth. To cope with it, we designed OSER-ENA, a scheduling algorithm based on node coloring and adapted to dense networks.

OSERENA is scalable as it uses a message whose size depends neither on the density nor on the number of nodes.

- **The interferences and the unreliability** are characteristics of wireless communications. In our work, interference cases regarding the communications supported are finely determined. That is why, the coloring problem definition is generalized and has as parameter the maximum distance between two interfering nodes. This makes our work more general than previous works which made the assumption: interferences are limited to 2 hops. Besides, to deal with the unreliable wireless links, we adapted the known coloring algorithm SERENA to such environments and enhanced its robustness in case of data gathering applications. Therefore, after topology changes, nodes can keep their initial colors without creating interferences: the coloring remains valid.

- **The regularity of the network topology**. Focusing on grid topologies, we determined bounds on the optimal number of colors. Furthermore, we developed a coloring method VCM that takes advantage of the regularity of these topologies to have an optimal periodic coloring.

- **The type of the traffic**. Nodes do not have the same traffic to transmit. For instance, nodes close to the sink have much more traffic to relay than others. We generalized our study considering that nodes require different numbers of slots. As the best of our knowledge, few research papers adopt this assumption. The contribution was the traffic aware time slot assignment TRASA that grants to each node a medium access time proportional to its demand.

7.2 Perspectives

With regard to the contributions of this thesis, we can identify the following perspectives:

1. *Adapting OSERENA and TRASA to occasional traffic demands of nodes.* This occasional demand may occur if two possible scenarios: (1) alarms that are in general prioritized, but should not prevent the transmissions of periodic traffic, (2) when traffic periodicities are longer than the whole cycle length.

2. *Adapting OSERENA and TRASA to multichannel assignment protocols.* Multichannel communication mitigates interferences, jamming and congestion. It also increases the network throughput. In addition, the current generation of wireless sensor motes provides multiple channels available for use. For instance, Micaz motes can communicate on multiple frequencies as specified in the IEEE 802.15.4 standard. So, how to extend the results of our thesis to fit this context? We can consider two possible problems:

    - *First problem: How to allocate frequencies to wireless sensors?* We can use OSERENA in case of homogeneous traffic demand or via TRASA otherwise. Intuitively,

this adaptation would be straightforward: Each frequency is mapped to one time slot and nodes having the same color transmit at the same frequency.
However, the difficulties are: (1) one channel can be preferred to another because of interferences external to the sensor network, (2) we have a predetermined number of channels, while the TDMA cycle in the slot allocation problem can be not strictly defined by the application. Thus the challenging objective is: given a number of channels, efficiently schedule nodes on these channels while maximizing the network throughput for instance.

- *Second problem: How to schedule transmissions of wireless sensors whose transmission frequencies are already assigned?* We could adapt TRASA and OSERENA to allocate time slots to these nodes. Any node should be able to receive data from its neighbors that transmit in another frequencies. So, how to schedule this node activity?

3. *Exploiting results of coloring grids in random topologies coloring.* This work is in progress and the methodology followed starts by adapting VCM to coloring grids of cells. More precisely, a cell is assimilated to one grid node in VCM. Then, we design a method to color the nodes inside one cell.

4. *Extending the energy efficient routing protocol EOLSR.* This protocol was designed for data gathering applications, where routes are built from any node to a sink node. We plan to extend this work as follows:

   - *Allow any node to communicate with any other node.* A specific case can be considered where the sink can communicate in point-to-point with any "pollable node".
   - *Supporting multiple sinks.* Having multiple sinks can be imposed by the application in to cases: first, when sensed data are of different types for example; second, to enhance robustness.

5. *For the OCARI project, we could improve the adaptivity of OSERENA and EOLSR*, mainly by tuning control messages periodicity according to the applications requirements and the wireless sensors capacities.

List of Publications

International Journals

1. Ichrak Amdouni, Pascale Minet, Cédric Adjih, *OSERENA: a Coloring Algorithm Optimized for Dense Wireless Networks*, to appear in the International Journal of Networked and Distributed Computing, IJNDC 2012.

2. Saoucène Mahfoudh, Pascale Minet, Ichrak Amdouni, *Energy Efficient Routing and Node Activity Scheduling in the OCARI Wireless Sensor Network*, Journal Future Internet vol. 2, pp. 308-340, August 2010.

3. Saoucène Mahfoudh, Gerard Chalhoub, Pascale Minet, Michel Misson, Ichrak Amdouni, *Node Coloring and Color Conflict Detection in Wireless Sensor Networks. Journal Future Internet*, vol. 2, pp. 469-504, October 2010.

International conferences

1. Cédric Adjih, Ichrak Amdouni, Pascale Minet, *VCM: The Vector-Based Coloring Method for Grid Wireless Ad Hoc and Sensor Networks*, MSWIM 2012, Paphos, Cyprus, October 21-25, 2012.

2. Ichrak Amdouni, Ridha Soua, Erwan Livolant, Pascale Minet, *Delay Optimized Time slot Assignment for Data Gathering Applications in wireless Sensor Networks*, ICWCUCA 2012, Clermont-Ferrand, France, August 28-30, 2012.

3. Ichrak Amdouni, Pascale Minet, *TRASA: TRaffic Aware Slot Assignment Algorithm in Wireless Sensor Networks*, ICCIT 2012, Hammamet, Tunisia, June 26-28, 2012.

4. Ichrak Amdouni, Pascale Minet, Cédric Adjih, *Node coloring in wireless networks: complexity results and grid coloring*, WMNC 2011, Toulouse, France, October 26-28, 2011.

Research Reports

1. Ichrak Amdouni, Cédric Adjih, Pascale Minet, *On the coloring of Grid Wireless Sensor Networks: the Vector-Based Coloring Method*, Inria Research Report 7756, October 2011.

2. Ichrak Amdouni, Pascale Minet, Cédric Adjih, *Oserena an Optimized Coloring Algorithm for Dense or Large Scale Wireless Networks*, Inria Research Report 7785, November 2011.

3. Ichrak Amdouni, Pascale Minet, Cédric Adjih, *Node coloring for dense wireless sensor networks*, Inria Research Report 7588, March 2011.

Bibliography

[1] Anastasi G.; Conti M.; Di Francesco M.; Passarella A., *Energy conservation in wireless sensor networks: a survey*, Journal Ad Hoc Networks, vol. 7, pp. 537-568, May 2009.

[2] Xing G.; Sha M.; Hackmann G.; Klues K.; Chipara O.; Lu C., *Towards unified radio power management for wireless sensor networks*, Journal Wireless Communications and Mobile Computing, vol. 9, pp. 313-323, 2009.

[3] *An estimation of sensor energy consumption*, Journal Progress In Electromagnetics Research, vol. 12, pp. 259-295, 2009.

[4] Heinzelman W.R.; Kulik J.; Balakrishnan H., *Adaptive protocols for information dissemination in wireless sensor networks*, The 5th ACM/IEEE conference on Mobile computing and networking, MobiCom'99, Seattle, Washington, United States, August 1999.

[5] Intanagonwiwat C.; Govindan R.; Estrin D.; Heidemann J., Silva F., *Directed diffusion for wireless sensor networking*, Journal IEEE/ACM Transactions on Networking (TON) archive, vol. 11, pp. 2-16, February 2003.

[6] Braginsky D.; Estrin D., *Rumor routing algorthim for sensor networks*, 1st ACM international workshop on Wireless sensor networks and applications, WSNA'02, Atlanta, Georgia, USA, September 2002.

[7] Yong Y.; Gehrke J., *The cougar approach to in-network query processing in sensor networks*, SIGMOD, vol. 31, pp. 9-18, September 2002.

[8] Chu M.; Haussecker H.; Zhao F., *Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks*, International Journal of High Performance Computing Applications, vol. 16, 2002.

[9] Sadagopan N.; Krishnamachari B.; Helmy A., *The ACQUIRE Mechanism for Efficient Querying in Sensor Networks*, In IEEE International Workshop on Sensor Network Protocols and Applications, SNPA03, Anchorage, AK, May 2003.

[10] Heinzelman W.; Chandrakasan A.; Balakrishnan H.; *Energy-efficient communication protocol for wireless sensor networks*, International Conference System Sciences, Hawaii, January 2000.

[11] Loscri V.; Morabito G.; Marano S., *A Two-Level Hierarchy for Low-Energy Adaptive Clustering Hierarchy*, The IEEE 62nd Semi-annual Vehicular Technology Conference, Dallas, Texas, USA, September 2005.

[12] Ye M.; Li C.; Chen G.; Wu J.; Ye M., *EECS: An energy efficient clustering scheme in wireless sensor networks*, The IEEE International Performance Computing and Communications Conference, IPCCC 2005, Phoenix, Arizona, April 2005.

[13] Lindsey S.; Raghavendra C.S., *PEGASIS: power efficient gathering in sensor information systems*, The IEEE Aerospace Conference, Big Sky, Montana, March 2002.

[14] Manjeshwar A.; Dharma P. A., *TEEN: A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks*, The 15th International Parallel & Distributed Processing Symposium, IPDPS '01, Washington, DC, USA, April 2001.

[15] Manjeshwar A.; Dharma P. A., *APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks*, The 16th International Parallel and Distributed Processing Symposium, IPDPS'02, Washington, DC, USA, April 2002.

[16] Gautam N.; Lee W.; and Pyun J., *Track-sector clustering for energy efficient routing in wireless sensor networks.* The 9th IEEE International Conference on Computer and Information Technology, Xiamen, China, October 2009.

[17] Ming L.; Jiannong C.; Guihai C.; Xiaomin W., *An Energy-Aware Routing Protocol in Wireless Sensor Networks*, Sensors 2009, vol. 9, pp. 445-462, January 2009.

[18] Yu Y.; Govindan R.; Estrin D., *Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks*, UCLA Computer Science Department UCLA-CSD TR-01-0023, May 2001.

[19] Xu Y.; Heidemann J.; Estrin D., *Geography-informed Energy Conservation for Ad-hoc Routing*, ACM MOBICOM 2001, Rome, Italy, July 2001.

[20] Basagni S.; Chlamtac I.; Syrotiuk V.; Woodward B., *A Distance routing effect algorithm for mobility (DREAM)*, MobiCom 1998, TX, USA, October 1998.

[21] Chen B.; Jamieson K.; Balakrishnan H.; Morris R., *SPAN: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks*, MobiCom 2001, Rome, Italy, July 2001.

[22] Hea T.; Stankovica J.A.; Lub C.; Abdelzahera T., *SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks*, International Conference on Distributed Computing Systems, ICDCS 2003, Providence, Rhode Island, USA, May 2003.

[23] Luo J.; Hubaux J.P., *Joint mobility and routing for lifetime elongation in wireless sensor networks*, The 24th IEEE INFOCOM, Miami, USA, March 2005.

[24] Chatzigiannakis I.; Kinalis A.; Nikoletseas S., *Sink mobility protocols for data collection in wireless sensor networks*, The 4th ACM international workshop on Mobility management and wireless access, MobiWac'06, Terromolinos, Spain, October 2006.

[25] Shah R.C; Roy S.; Jain S., *Data MULEs: modeling and analysis of a three-tier architecture for sparse sensor networks*, Journal Ad Hoc Networks, vol. 1, pp. 30-41, May 2003.

[26] Kee-Young S.; Junkeun S.; JinWon K.; Misun Y.; Pyeong S.M., *REAR: Reliable Energy Aware Routing Protocol for Wireless Sensor Networks*, The 9th International Conference on Advanced Communication Technology, February 2007.

[27] Chandane M.M; Bhirud S.G; Bonde S.V, *Distributed Energy Aware Routing Protocol for Wireless Sensor Network*, Journal of Computer Applications, vol. 34, pp. 6-11, November 2011.

[28] Senouci S. M.; Pujolle G., *Energy efficient routing in wireless ad hoc networks*, 2004 IEEE International Conference on Communications ICC'04, Paris, France, June 2004.

[29] Scott K.; Bambos N., *Routing and Channel Assignment for Low Power Transmission in PCS*, the 5th IEEE International Conference on Universal Personal Communications, Cambridge, MA, USA, September 1996.

[30] Mahfoudh S.; Minet P., *Eolsr: an Energy Efficient Routing Protocol in Wireless Ad Hoc and Sensor Networks*, Journal of Interconnection Networks JOIN'08, vol. 9, pp. 389-408, December 2008.

[31] Jain S.; Kaushik P.; Jyoti S., *Energy Efficient Maximum Lifetime routing for wireless Sensor Network*, Journal Of Advanced Smart Sensor Network Systems, IJASSN, vol. 2, January 2012.

[32] Wang W.S.; O'Keeffe R.; Wang N.; Hayes M.; O'Flynn B.; O'Mathuna C., *Practical wireless sensor networks power consumption metrics for building energy management applications*, The 23rd European Conference Forum Bauinformatik 2011, Construction Informatics, Cork, Ireland, September 2011.

[33] Chen Y.P.; Liestman A.L.; Jiangchuan L., *Energy-Efficient Data Aggregation Hierarchy for Wireless Sensor Networks*, The 2nd International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks, Lake Buena Vista, FL, USA, August 2005.

[34] Kuo T.W.; Tsai M.J., *On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: NP-completeness and approximation algorithms*, INFOCOM 2012, Orlando, Florida USA, March 2012.

[35] Ding M.; Cheng X.; Xue G., *Aggregation Tree Construction in Sensor Networks*, The 58th IEEE Vehicular Technology Conference, Orlando, Florida, USA, October 2003.

[36] Lee M.; Wong V.W.S., *An Energy-Aware Spanning Tree Algorithm for data aggregation in wireless sensor networks*, IEEE PacRim 2005, Victoria, BC, Canada, August 2005.

[37] Eskandari Z.; Yaghmaee M.H.; Mohajerzadeh, A.H., *Energy Efficient Spanning Tree for Data Aggregation in Wireless Sensor Networks*, The 17th International Conference on Computer Communications and Networks, ICCCN'08, St Thomas U.S. Virgin Islands, August 2008.

[38] Fei H.; Xiaojun C.; Carter M., *Optimized Scheduling for Data Aggregation in Wireless Sensor Networks*, The International Conference on Information Technology: Coding and Computing, ITCC'05, Las Vegas, Nevada, USA, April 2005.

[39] Majumder K.; Sarkar S.K., *Clustered Chain based Power Aware Routing (CCPAR) Scheme for Wireless Sensor Networks*, International Journal on Computer Science and Engineering, IJCSE, vol. 2, pp. 2953-2963, 2010.

[40] Zhuang Y.; Pan J.; Wu G., *Energy-Optimal Grid-Based Clustering in Wireless Microsensor Networks*, IEEE ICDCS Workshop on Wireless Ad hoc and Sensor Networking (WWASN), 2009. The 29th IEEE International Conference on Distributed Computing Systems Workshops, ICDCSW'09, Montreal, Quebec, Canada, June 2009.

[41] Yu L.; Wang N.; Zhang W.; Zheng W., *GROUP: A Grid-Clustering Routing Protocol for Wireless Sensor Networks*, Wireless Communications, Networking and Mobile Computing, WiCOM'06, Wuhan City, China, September 2006.

[42] Wu J.; and Li H., *On calculating connected dominating set for efficient routing in ad hoc wireless networks*, The 3rd workshop on Discrete algorithms and methods for mobile computing and communications, DIALM'99, Seattle, Washington, United State, August 1999.

[43] Garey M.R.; Johnson D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co. 1979.

[44] Adjih C.; Clausen T.; Jacquet P.; Laouiti A.; Minet P.; Muhlethaler P.; Qayyum A.; Viennot L., *Optimized Link State Routing Protocol*, RFC 3626, IETF, October 2003.

[45] Clausen T; Herberg U., *Comparative study of RPL-enabled optimized broadcast in Wireless Sensor Networks*, The 6th Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSIP, Brisbane, Australia, December 2010.

[46] Winter T.; Thubert P.; Brandt A.; Hui J.; Kelsey R.; Levis P.; Pister K.;, Struik R.; Vasseur JP.; Alexander R., *RPL: IPv6 Routing Protocol for Low power and Lossy Networks*, RC 6550, IETF, June 2010.

[47] Sze-Yao N.; Yu-Chee T.; Yuh-Shyan C.; Sheu, Jang-Ping, *The Broadcast Storm Problem in a Mobile Ad Hoc Network*, The 5th annual ACM/IEEE international conference on Mobile computing and networking, MobiCom '99, Seattle, Washington, United States, August 1999.

[48] Haa Z.; Halpern J.Y.; Li L., *Gossip-Based Ad Hoc Routing*, Journal of IEEE/ACM Transactions on Networking (TON), vol. 14, pp. 479-491, June 2009.

[49] Levis P.; Clausen T.; Hui J.; Gnawalli O.; Ko J., *The Trickle Algorithm, RFC 6206, MARS 2011.*, `http://tools.ietf.org/html/rfc6206`

[50] Pei G.; Gerla M.; Chen T.W., *Fisheye state routing in mobile ad hoc networks*, The ICDCS Workshops, Taipei, Taiwan, April 2000.

[51] Nguyen D.; Minet P., *Scalability of the OLSR Protocol with the Fish Eye Extension*, The 6th international Conference on Networking, ICN'07, Sainte-Luce, Martinique, April 2007.

[52] Cardei M.; Du D., *Improving wireless sensor network lifetime through power aware organization*, ACM Journal of Wireless Networks, May 2005.

[53] Cardei M.; Thai M.; Li Y.; Wu W., *Energy-efficient target coverage in wireless sensor networks*, IEEE INFOCOM 2005, Miami, Florida, March 2005.

[54] Carle J.; Simplot-Ryl D., *Energy-Efficient Area Monitoring for Sensor Networks*, Computer, vol. 37, pp. 40-46, February, 2004.

[55] Ye F.; Zhong G.; Lu S.; Zhang L., *A Robust Energy Conserving Protocol for Long-lived Sensor Networks*, The 23rd International Conference on Distributed Computing Systems, ICDCS'03, Providence, RI, USA, May 2003.

[56] Cho J.; Kim G.; Kwon T.; Choi Y., *A Distributed Node Scheduling Protocol Considering Sensing Coverage in Wireless Sensor Networks*, The 65th Vehicular Technology Conference, VTC'07, Dublin, Ireland, April 2007.

[57] Ye W.; Heidmann J.; Estrin D., *An Energy-Efficient MAC Protocol for Wireless Sensor Networks*, IEEE INFOCOM, New York, USA, June 2002.

[58] Dam T. V.; Langendoen K., *An adaptive energy-efficient MAC protocol for wireless sensor networks*, ACM SenSys'03, November 2003.

[59] Lu G.; Krishnamachari B.; Raghavendra C., *An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks*, Parallel and Distributed Processing Symposium, April 2004.

[60] Lu G.; Krishnamachari B.; Raghavendra C., *O-MAC: An Organized Energy-Aware MAC Protocol for Wireless Sensor Networks*, IEEE ICC, Glasgow, UK, June 2007.

[61] Rajendran V.; Obraczka K.; Garcia-Luna-Aceves J.J., *Energy-efficient, collision-free medium access control for wireless sensor networks*, Sensys'03, Los Angeles, California November 2003.

[62] Rajendran V.; Garcia-Luna-Aceves J.J.; Obraczka K., *Energy-efficient, application-aware medium access for sensor networks*, IEEE MASS 2005, Washington, November 2005.

[63] Winnie L.L.; Amitava D.; Rachel C.O., *FlexiTP: A Flexible Schedule-Based TDMA Protocol for Fault-Tolerant and Energy-Efficient Wireless Sensor Networks*, IEEE Transactions on Parallel and Distributed Systems, vol. 19, pp. 851-864, June 2008.

[64] Gobriel S.; Mosse D.; Cleric R., *TDMA-ASAP: sensor network TDMA scheduling with adaptive slot stealing and parallelism*, ICDCS 2009, Montreal, Canada, June 2009.

[65] Rhee I.; Warrier I.; Aia M.; Min J., *Z-MAC: a hybrid MAC for wireless sensor networks*, SenSys'05, San Diego, California, November 2005.

[66] Rhee I.; Warrier A.; Xu L., *Randomized dining philosophers to TDMA scheduling in wireless sensor networks*, Technical Report TR-2005-21, Dept of Computer Science, North Carolina State University, April 2005.

[67] Minet P.; Mahfoudh S., *SERENA: SchEduling RoutEr Nodes Activity in wireless ad hoc and sensor networks*, IWCMC 2008, IEEE International Wireless Communications and Mobile Computing Conference, Crete Island, Greece, August 2008.

[68] Ingelrest F.; Simplot-Ryl D.; Stojmenovic I., *Optimal Transmission Radius for Energy Efficient Broadcasting Protocols in Ad Hoc and Sensor Networks*, In IEEE Transactions on Parallel and Distributed Systems, vol. 17, pp. 536-547, June 2006.

[69] De-yun G.; Lin-juan Z.; Hwang-cheng W., *Energy saving with node sleep and power control mechanisms for wireless sensor networks*, The Journal of China Universities of Posts and Telecommunications, vol. 18, pp. 49-59, February 2011.

[70] Correia L. H.A.; Macedo D.F.; Dos Santos A.L.; Loureiro A.A.F.; Nogueira J.M.S., *Transmission power control techniques for wireless sensor networks*, In journal Computer Networks, vol. 51, pp. 4765-4779, December 2007.

[71] Zayane M. A.; Dhaou R.; Beylot A.L., *Routing and Power Adaptation for Mostly-on Wireless Sensor Network Applications*, IFIP Wireless Days, Dublin, Ireland, November 21-23, 2012.

[72] West D. B., *Introduction to Graph Theory*, 2nd edition Prentice Hall, 2001.

[73] Johansson O., *Simple distributed ($\Delta+1$) -coloring of graphs*, Information processing Letters, 70, 1999.

[74] Finoccho I.; Panconesi A.; Silvestri R., *Experimental analysis of simple distributed vertex coloring algorithms*, SODA 2002, San francisco, California, January 2002.

[75] Busch C.; Magdon-Ismail M.; Sivrikaya F.; Yener B., *Contention-free MAC protocols for wireless sensor networks*, DISC 2004, Amsterdam, Netherlands, October 2004.

[76] Brelaz D., *New methods to color the vertices of a graph*, Communications of the ACM, 22(4), 1979.

[77] Hansen J.; Kubale M.; Kuszner L.; Nadolski A., *Distributed largest-first algorithm for graph coloring*, EURO-PAR 2004, Dresden, Germany, August 2004.

[78] Glover F., *Tabu search-part I*, ORSA Journal on Computing, vol. 1, pp. 190-205, 1989.

[79] Bermond J.C.; Havet F.;Huc F.; Linhares-Sales C., *Improper colouring of weighted grid and hexagonal graphs*, Discrete Mathematics, Algorithms and Applications, vol. 2, pp. 395-411, 2010.

[80] Cargiannis I.; Fishkin A.; Kaklamanis C.; Papaioannou E., *A tight bound for online coloring of disk graphs*, Theoretical Computer Science, 2007.

[81] Kierstead H. A.; Qin J., *Coloring interval graphs with First-Fit*, Discrete Math, vol. 144, pp. 47-57, September 1995.

[82] Vizing V., *On an estimate of the chromatic class of a p-graph*, Diskret. Analiz., vol. 3, pp. 23-30, 1964.

[83] Shanon C.E, *A theorem on coloring the lines of a network*, Journal Maths. Physics, vol. 28, pp. 148-151, 1949.

[84] `http://en.wikipedia.org/wiki/Edge_coloring`.

[85] Durand D.; Jain R.; Tseytlin D., *Distributed Scheduling Algorithms to Improve the Performance of Parallel Data Transfers*, Proc. IPPPS'94 Workshop on Input/Output in Parallel Computer Systems, April 1994.

[86] Durand D.; Jain R.; Tseytlin D., *Applying Randomized Edge Coloring Algorithms to Distributed Communication: An Experimental Study.*, ACM Symposium of Parallel Algorithms and Architectures, 1995.

[87] Marathe M. V.; Panconesi A.; Risinger L.D., *An Experimental Study of a Simple, Distributed Edge-Coloring Algorithm*, ACM Journal of Experimental Algorithmics, vol. 9, 2004.

[88] Herman T.; Pemmaraju S.; Pirwani, I., *Oriented Edge Colorings and Link Scheduling in Sensor Networks*, The 1st International Conference on Communication System Software and Middleware, Comsware'06, Delhi, India, January 2006.

[89] Cheng M.; Yin L., *Transmission Scheduling in Sensor Networks via Directed Edge Coloring*, The International Conference on Communications, ICC'07, Glasgow, Scotland, 2007.

[90] Hilgemeier M.; Drechsler N.; Drechsler R., *Fast heuristics for the edge coloring of large graphs*, Euromicro Symposium on Digital system design, DSD'03, IEEE computer Society, 2003.

[91] Avanthay C.; Hertz A.; Zufferey N., *A variable neighborhood search for graph coloring*, Journal European Journal of Operational Research, vol. 151, pp. 379-388, 2003.

[92] Tzeng C.H.; Jehn-Ruey J.; Shing-Tsaan H., *A self-stabilizing $(\Delta + 4)$-edge-coloring algorithm for planar graphs in anonymous uniform systems*, Journal Information Processing Letters, vol. 101, pp. 168-173, February 2007.

[93] Galinier P.; Hertz A., *A survey of local search methods for graph coloring*, Computers & Operations Research, vol. 3, pp. 2547-2562, September 2006.

[94] Wu Q.; Hao J.K;, *Coloring large graphs based on independent set extraction*, to appear in Computers and Operations Research, Elsevier, 2011.

[95] Even G.; Lotker Z.; Ron D.; Smorodinsky S., *Conflict-Free Colorings of Simple Geometric Regions with Applications to Frequency Assignment in Cellular Networks*, SIAM J. Comput, vol. 33, pp 94-136, 2004.

[96] Subramanian A.P.; Gupta H.; Das S.R., *Minimum Interference Channel Assignment in Multi-Radio Wireless Mesh Network*, The 4th Annual IEEE Communications Society Conference on Mesh and Ad Hoc Communications and Networks SECON'07, San Diego, California, USA, June 2007.

[97] Chowdhury K.R.; Chanda P.; Agrawal D.P.; Zeng Q.A., *DCA- A Distributed Channel Allocation Scheme for Wireless Sensor Networks*, The 16th International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC'05, Berlin, Germany, September 2005.

[98] Wang W.; Liu X., *List-coloring based channel allocation for open-spectrum wireless networks*, The 62nd Vehicular Technology Conference, VTC'05, Dallas, Texas, USA, September 2005.

[99] Zheng H.; Peng C., *Collaboration and fairness in opportunistic spectrum access*, IEEE International Conference on Communications, ICC'05, Seoul Korea, March 2005.

[100] Gandham S.; Dawande M.; Prakash R., *Link scheduling in sensor networks: distributed edge coloring revisited*, The 24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM'05, Miami, March 2005.

[101] Cheng M.; Yin L., *Transmission Scheduling in Sensor Networks via Directed Edge Coloring*, IEEE International Conference on Communications, ICC'07, Glasgow, Scotland, June 2007.

[102] Schneider J.; Wattenhofer R., *Coloring unstructured wireless multi-hop networks*, The 28th ACM symposium on Principles of distributed computing, PODC'09, USA, 2009.

[103] Fertin G.; Godard E.; Raspaud A., *Acyclic and k-distance coloring of the grid* Journal Information Processing Letters, vol. 87, pp. 51-58, July 2003.

[104] Mahfoudh S.; Minet P.; Amdouni I., *Energy Efficient Routing and Node Activity Scheduling in the OCARI Wireless Sensor Network*, Journal Future Internet, vol. 2, pp. 308-340, September 2010.

[105] Brelaz D., *New methods to color the vertices of a graph*, Commun. ACM, pp. 251–256, 1979.

[106] Hansen J.; Kubale, M.; Kuszner, L.; Nadolski, A. *Distributed largest-first algorithm for graph coloring*. In Proceedings of EURO-PAR 2004, Pisa, Italy, August 2004.

[107] Mahfoudh S.; Chalhoub G.; Minet P.; Misson M.; Amdouni I., *Node Coloring and Color Conflict Detection in Wireless Sensor Networks*, Journal Future Internet 2010, vol. 2, pp. 469-504, October 2010.

[108] Melodia T.; Vuran C. M.; Pompili D., *The State of the Art in Cross-layer Design for Wireless Sensor Networks*, The Second international conference on Wireless Systems and Network Architectures in Next Generation Internet, EURO-NGI'05, Villa Vigoni. Italy, July 2005.

[109] Wang Y.H.; Chao C.F., *Dynamic backup routes routing protocol for mobile ad hoc networks*, Journal Information Sciences, vol. 176, pp. 161-185, January 2006.

[110] Lee S.J.; Gerla M., *AODV-BR: Backup Routing in Ad hoc Networks*, Wireless Communications and Networking Conference, WCNC'02, Chicago, USA, September 2000.

[111] Ganesan, D.; Govindan, R.; Shenker, S.; Estrin, D. *Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks*, Journal Mobile Computing and Communications Review, vol. 5, pp. 11-25, October 2001.

[112] Hsu V.S; Kahn J.M; Pister K.S, *Wireless communications for smart Dust*, Electronics Research Laboratory Technical Memorandum Number M98/2, University of California, Berkely, January 1998.

[113] Clark B.; Colbourn C.; Johnson D., *Unit disk graphs*, Discrete Mathematics, vol. 86, December 1990.

[114] Xiaoyuan T.; Guoqiang M.; Anderson B.D.O., *On the Probability of K-hop Connection in Wireless Sensor Networks*, IEEE Communications Letters, vol. 11, pp. 662-664, August 2007.

[115] Baumgartner T.; Fekete S.P.; Kamphans T.; Kröller A., Pagel M., *Hallway Monitoring: Distributed Data Processing with Wireless Sensor Networks*, REALWSN, December 2010.

[116] Chakrabarty K.; Iyengar S. S.; Qi H. R. and Cho E., *Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks*, IEEE Transactions on Computers, vol. 51, pp. 1448-1453, 2002.

[117] Poe W.Y.; Schmitt J.S., *Node deployment in large wireless sensor networks: coverage, energy consumption, and worst-case delay*, Asian Internet Engineering Conference (AINTEC'09), Thailand, November 2009.

[118] Panichpapiboon S.; Ferrari G.; Tonguz O.K., *Sensor networks with random versus uniform topology: MAC and interference considerations*, 59th Vehicular Technology Conference (VTC'04), May 2004.

[119] McCulloch J.; Mccarthy P.; Siddeswara M. G.; Wei, P.; Hugo D.; Terhorst, A., *Wireless sensor network deployment for water use efficiency in irrigation*, Proc. of the workshop on Real-world wireless sensor networks, March 2008.

[120] Amdouni I.; Minet P.; Adjih C., *Node coloring in wireless networks: complexity results and grid coloring*, WMNC 2011, Toulouse, France, October 26-28, 2011.

[121] Vallée,B.; Vera A., *Lattice reduction in two dimensions: analyses under realistic probabilistic models*, DMTCS'07, January 2007.

[122] Amdouni, I.; Minet, P.; Adjih, C., *Node coloring for dense wireless sensor networks*, INRIA Research Report 7588, Paris-Rocquencourt, France, March 2011.

[123] Vector-Based Coloring Method page, `http://hipercom.inria.fr/SensorNet/VCM/`, INRIA, 2011.

[124] Wubben D.; Seethaler D.; Jaldén J.; Matz G., Lattice Reduction, A survey with applications in wireless communications, *IEEE Signal Processing Magazine*, vol. 29, May 2011.

[125] Thue A., *Über die dichteste Zusammenstellung von kongruenten Kreisen in einer Ebene*, Norske Vid. Selsk. Skr. pp. 1-9, 1910.

[126] Tóth L.F.; *Über die dichteste Kugellagerung*, Math. Z. vol. 48, pp. 676-684, 1943.

[127] Amdouni I.; Adjih C.; Minet P., *On the Coloring of Grid Wireless Sensor Networks: the Vector-Based Coloring Method*, Inria Research Report, RR-7756, September 2011.

[128] Ahn G.S.; Miluzzo E.; Campbell A. T.; Hong S.G.; Cuomo F., *Funneling-mac: A localized, sink-oriented mac for boosting fidelity in sensor networks*, The 4th ACM Conference on Embedded Networked Sensor Systems (SenSys 2006), Boulder, CO, USA, November 2006.

[129] Choi H.; Wang J.; Hughes E. *Scheduling for information gathering on sensor network*, Wireless Networks, 2009.

[130] Ergen S.C.; Varaiya P., *TDMA scheduling algorithms for wireless sensor networks*, Wireless Networks, May 2010.

[131] Song W. Z.; Huang R.; Shirazi B.; LaHusen R., *TreeMAC: Localized TDMA MAC Protocol for Real-time High-data-rate Sensor Networks*, Journal of Pervasive and Mobile Computing, Percom'09, vol. 5, pp. 750-765, December 2009.

[132] Huang R., Song W.Z., Xu M., Shiraz B., *Localized QoS-Aware Media Access Control in High-Fidelity Data Center Sensing Networks*, The 1st International Green Computing Conference, IGCC'10, Chicago, USA, August 2010.

[133] Incel D.O.; Ghosh A.; Krishnamachari B.; Chintalapudi K., *Fast data collection in treebased wireless sensor networks*, IEEE Transactions on Mobile Computing, vol. 1, pp. 86-99, January 2009.

[134] Turau V.; Weyer C.; Renner C., *Efficient Slot Assignment for the Many-to-One Routing Pattern in Sensor Networks*, The 1st International Workshop on Sensor Network Engineering, IWSNE'08, Santorini Island, Greece, June 2008.

[135] Mao J.; Wu Z.; Wu X., *A TDMA scheduling scheme for many-to-one communications in wireless sensor networks.* Computer Communications. vol. 30, pp. 863-872, February 2007.

[136] Amdouni I.; Soua R.; Livolant E.; Minet P., *Delay Optimized Time Slot Assignment for Data Gathering Applications in Wireless Sensor Networks*, Third International Conference on Wireless Communications in Unusual and Confined Areas, Clermont-ferrand, France, August 28-30, 2012.

[137] http://www.gnu.org/software/glpk/

Appendix A

Mathematical Results Related to OSERENA Algorithm

In this Annex, we group mathematical results that are useful to prove the properties of OSERENA, the coloring algorithm presented in Chapter 4.

A.1 Correctness of OSERENA

Lemma 7. *With OSERENA, any node u colors itself if and only if it has the highest priority among all the uncolored nodes in $\mathcal{N}(u)$.*

Proof: Let us show that if any node u is coloring itself, then it has the highest priority among the uncolored nodes up to 3-hop. According to the Rule R'1, if u is coloring itself then we have from the following Equation:
$priority(u) = max(max\_prio1(u), max\_prio2(u), max\_prio3(u))$.
From this equation, we have:
(1) $priority(u) \geq max(max\_prio1(u))$ leads to the fact that no uncolored one-hop neighbor has a priority higher than u.
(2) $priority(u) \geq max(max\_prio2(u))$. Hence, no uncolored two-hop neighbor has a priority higher than u, otherwise we would have the following contradiction:
$priority(u) \geq max(max\_prio2(u)) > priority(u)$.
(3) $priority(u) \geq max\_prio3(u)$. Hence, no uncolored three-hop neighbor in $\mathcal{N}(u)$ has a priority higher than u, otherwise we would have the following contradiction: $priority(u) \geq max\_prio3(u) > priority(u)$.
Hence, node u has the highest priority among the uncolored nodes in $\mathcal{N}(u)$.

Conversely, if node u has the highest priority among its uncolored neighbors up to 3 hops, then:

- all its uncolored one-hop neighbors have a smaller priority. Hence, for any v uncolored one-hop neighbor of u, we have $priority(v) < priority(u)$. Hence,

$max(max\_prio1(u)) = max\ _{v \in 1hop(u)}(priority(v)\ for\ v\ uncolored\ ) < priority(u)$;

- all its uncolored two-hop neighbors have a smaller priority. Let us consider w the node with the highest priority in $max\_prio2(u)$. It denotes the highest priority of an uncolored node w that is one-hop neighbor of v, itself one-hop neighbor of u. Consequently, we have the following cases:
  - node w is the node u itself and has priority $priority(u)$;
  - node w is a one-hop or two-hop neighbor of node u. In which case, we have by assumption: $priority(w) < priority(u)$.

  Hence, $max(max\_prio2(u)) = priority(u)$.

- and all its uncolored three-hop neighbors in $\mathcal{N}(u)$ have a smaller priority. By definition, $max\_prio3(u)$ is the maximum priority of uncolored nodes q that are one-hop neighbors of w, itself one-hop neighbor of v, one-hop neighbor of u. Consequently, we have the following cases:
  - node q is the node u itself and has priority $priority(u)$;
  - node q is a one-hop, two-hop or three-hop neighbor of node u. In which case, we have by assumption: $priority(q) < priority(u)$.

  Hence, $max\_prio3(u) = priority(u)$.

Finally, $priority(u) = max(max\_prio1(u), max\_prio2(u), max\_prio3(u))$.
Hence, node u is coloring itself with OSERENA. ■

Lemma 8. *With OSERENA, when node u colors itself, it knows all the colors taken in* $\mathcal{N}(u)$ *with a higher priority.*

Proof: The exchange of $Color$ messages allows any node u to know any uncolored node in $\mathcal{N}(u)$ having a higher priority than itself. Node u also knows the colors of already colored nodes in $\mathcal{N}(u)$ by means of $bitmap1$, $bitmap2$ and $bitmap3$. Thus, when u colors itself, it takes the smallest color unused in these bitmaps, and hence unused in $\mathcal{N}(u)$. ■

Lemma 9 *OSERENA coloring ends when all nodes are colored.*

Proof: Rule R5 defines when any node should stop sending color messages. Here we prove that with this rule, OSERENA ends when all nodes are colored. If u is colored and $max\_prio1(u) = \varnothing$, then node u and all its one-hop neighbors are colored. Moreover, if node u receives a $Color$ message from any one-hop neighbor v with $max\_prio1(v) = max\_prio2(v) = \varnothing$, it means that all the one-hop neighbors of v and all the one-hop neighbors of its one-hop neighbors are already colored. Hence, all nodes up to three-hop from u and belonging to $\mathcal{N}(u)$ are colored. The coloring algorithm ends when node u as well as all its 1-hop, 2-hop and 3-hop neighbors are colored. ■

Lemma 10. *In a wireless network meeting assumptions A0, A1, A2 and A3 in the absence of message loss and node failure, all nodes color themselves with OSERENA and stop sending their Color message.*

Proof: Let us consider any node u. The nodes in $\mathcal{N}(u)$ color themselves according to their priority. As soon as u becomes the uncolored node with the highest priority, it colors itself according to rules R'1 and R'2. From Lemma 9, OSERENA ends when all nodes are colored. Hence, it is useless for u to send its *Color* message insofar as any information contained in its message is already known by its one-hop, two-hop and three-hop neighbors in $\mathcal{N}(u)$ and these nodes are already colored. ■

A.1.1 Equivalence of OSERENA to a Centralized Algorithm

Lemma 11. For any node u, for any given priority assignment, nodes $\in \mathcal{N}(u)$ color themselves in the same order with OSERENA and First Fit.

Proof: Let us consider any node u that is coloring itself in OSERENA. According to Lemma 8, we have:

- any node $v \in \mathcal{N}(u)$ such that $priority(v) > priority(u)$ is already colored in OSERENA, otherwise u could not color itself now;
- any node $v \in \mathcal{N}(u)$ such that $priority(v) < priority(u)$ is not colored in OSERENA, because it is constrained by node u that is not yet colored.

Hence, in $\mathcal{N}(u)$ the coloring order in OSERENA is compliant with the priority order that is by definition followed by First Fit. In conclusion, both coloring algorithms follow the priority order to color nodes in a given neighborhood $\mathcal{N}(u)$. ■

A.1.2 Convergence Time of OSERENA

In this section, we compute the probability P_1, P_2 and P_3 of the occurrence of the events leading to coloring delays. Recall that these events are:

- E_1*:* $\exists$ v_1, v_2 and v_3, three nodes that are 2-hop away from u and 4-hop away from each other.
- E_2*:* these three nodes v_1, v_2 and v_3 have a priority higher than u.
- E_3*:* v_1, v_2 and v_3 are colored simultaneously.

A.1.2.1 Estimation of an Upper Bound of P_1

On Figure A.1, a bound of P_1 corresponds to the probability for v_3 to belong to the hatched area.

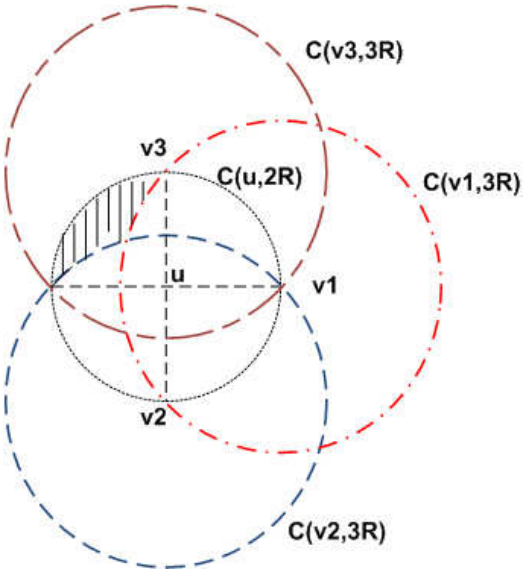


Figure A.1: Possible zone for node v_3.

The computation of the probability P_1 is illustrated in Figure A.2.

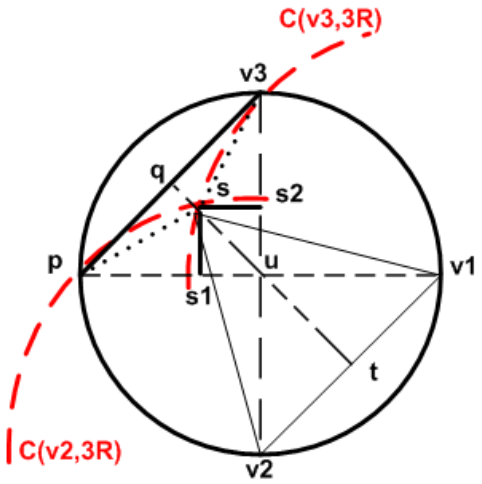


Figure A.2: Computation of P_1.

Nodes v_i, for $i \in [1,3]$, should belong to $\mathcal{D}(u,2R) \setminus \mathcal{D}(u,R)$ and should be at a distance belonging to $]3R,4R]$ from each other. To maximize the number of possible nodes v_3, we take $v_1 \in \mathcal{C}(u,2R)$. The choice of v_1 done, we increase the number of possible nodes v_3 by taking $v_2 \in \mathcal{C}(u,2R) \cap \mathcal{C}(v_1,3R)$, approximating $3R+\varepsilon$ by $3R$. We make v_1 and v_2 closer increasing again the possibilities for v_3 by transforming the triangle (v_1,u,v_2) in a right triangle. We then have $d(v_1,v_2) = 2R\sqrt{2}$ computed as the hypotenuse in the triangle (u,v_1,v_2). We now select v_3 that belongs to $\mathcal{D}(u,2R) \setminus \mathcal{D}(u,R) \setminus \mathcal{D}(v_1,2R\sqrt{2}) \setminus \mathcal{D}(v_2,2R\sqrt{2})$, corresponding to the hatched area depicted in Figure A.1. We compute S_P the surface of this area.

We notice that we have $S_P \leq S_D - 2S_T - S_C$, where S_D is the surface of the disk quarter $\mathcal{D}(u, 2R)$, S_T is the surface of the triangle formed by s, u and v_3 and S_C the surface of the square (s, s_1, u, s_2) whose diagonal is y. We first compute $d(u, q)$ in the right triangle (u, q, v_3). We get $2R^2 + d(u, q)^2 = 2^2R^2$. Hence, $d(u, q) = R\sqrt{2}$. In the isocel triangle (v_1, v_2, s), we compute $d(q, t$. We have: $(d(s, t))^2 + 2R^2 = 8R^2$. Since $d(s, t) = d(s, u) + d(u, q)$, we get $d(s, u) = d(s, t) - d(u, q) = (\sqrt{6} - \sqrt{2})R$. We then get:
$S_D = \Pi R^2$ and $S_C = y^2/2 = (4 - 2\sqrt{3})R^2$.
$S_T = (\sqrt{4 - 2\sqrt{3}} - 2 + \sqrt{3})R^2$.
We deduce $S_P = (\Pi - 2\sqrt{4 - 2\sqrt{3}})R^2$.
Hence, $P_1 = \frac{number\ of\ favorable\ cases}{number\ of\ possible\ cases} = \frac{S_P}{4\Pi R^2}$.
Finally, we get $P_1 = \frac{1}{4} - \frac{\sqrt{4-2\sqrt{3}}}{2\Pi}$.

A.1.2.2 Estimation of an Upper Bound of P_2

For any node u, let us compute P_2 the probability of event E_2: there exists three nodes two-hop away from u with a priority higher than u. This event E_2 can be considered as the intersection of two events E_{21} and E_{22}, where E_{21} means that there exists three nodes in $\mathcal{D}(u, 2R)$ with a priority higher than u. Event E_{22} means that three nodes in $\mathcal{D}(u, 2R)$ do not belong to $\mathcal{D}(u, R)$. We do not have event E_21 if and only if in $\mathcal{D}(u, 2R)$, 1) u has the highest probability, or 2) u has the second highest probability or 3) u has the third highest probability. Let M denote the number of nodes that are exactly one-hop away from u. The average number of nodes in $\mathcal{D}(u, 2R)$ is equal to $4M + 1$. We compute P_{21} the probability of event E_{21}. We have $P_{21} = 1 - \frac{3}{4M+1}$.
We can now compute P_{22} the probability of event E_{22}. We get P_{22}=probability that none of these three nodes in $\mathcal{D}(u, 2R)$ belong to $\mathcal{D}(u, R)$. Since the nodes are independent, we get $P_{22} = (1 - \frac{\Pi R^2}{4\Pi R^2})^3 = (3/4)^3$. Since events E_{21} and events E_{22} are independent, we get $P_2 = P_{21} \cdot P_{22}$, leading to $P_2 = \frac{27}{64}(1 - \frac{3}{4M+1})$.

A.1.2.3 Estimation of an Upper Bound of P_3

For any node u, we select the last three nodes v_1, v_2 and v_3, two-hop away from u that color themselves just before u. We want to compute P_3 the probability that event E_3 occurs that is: these three nodes color themselves simultaneously. We can bound P_3 by 1.

Appendix B

Mathematical Results Related to VCM Method

In this annex, we group mathematical results and grid properties that are useful to study the validity and the optimality of *VCM*, the coloring method presented in Chapter 5. These results can be applied to *VCM*, or any other algorithm.

B.1 Relation between Number of Hops and Actual Distance

Results in this section are inequalities, establishing links between number of hops and actual distance.

Lemma 16. *For any point V of $\mathbb{R}^2$, there exists a node V' of the grid $\mathbb{Z}^2$ such that $d(V,V') \leq \sqrt{2}/2$.*

Proof: In the worst case, the node V occupies the center of a grid cell. It is at equal distance of two grid nodes that are diagonally opposed. Hence, its distance to one of them is equal to $\sqrt{2}/2$. ■

Lemma 17. *For any transmission range $R > \sqrt{2}$, for any grid node U, any node V that meets $d(U,V) \leq (R-\sqrt{2})h$ is at most h-hop away from U.*

Proof: We consider the $h-1$ points of $\mathbb{R}^2$ that allow us to divide the segment $[U,V]$ in h equal parts.
Let W_i be these nodes, with $i \in [1, h-1]$.
For any $i \in [1, h-1]$, let W'_i the grid point the closest to W_i. For simplicity reason, we denote $W'_0 = U$ and $W'_h = V$. We have $d(U,V) \leq \sum_{i=0}^{h-1} d(W'_i, W'_{i+1})$.
We have $d(W'_i, W'_{i+1}) \leq d(W'_i, W_i) + d(W_i, W_{i+1}) + d(W_{i+1}, W'_{i+1})$. According to Lemma 16, we have $d(W_i, W'_i) \leq \sqrt{2}/2$. Hence, we get $d(W'_i, W'_{i+1}) \leq \sqrt{2} + d(W_i, W_{i+1})$. By construction, $d(W_i, W_{i+1}) = d(U,V)/h$.
If $d(U,V) \leq (R-\sqrt{2})h$, then $d(W'_i, W'_{i+1}) \leq R$. Hence, nodes W_i for $i \in \{1, 2, \ldots, h-1\}$ constitute a h-hop path between U and V. ■

Lemma 18. *For any transmission range R, for any two grid nodes U and V, in h-hop coloring, if $d(U,V) > hR$ then U and V are at least $(h+1)$-hop away.*

Proof: By contradiction assume that, U and V are h-hop away or less. Let W_i be the $k-1$ nodes constituting the k-hop path between U and V, with $k \leq h$. Let $W_1 = U$, and $W_h = V$. Since nodes W_i are 1-hop neighbors, we have:

$$|\vec{UV}| = |\sum_{i=1}^{h} W_i\vec{W}_{i+1}| \leq \sum_{i=1}^{h} d(W_i, W_{i+1}) \leq hR.$$

Hence the contradiction. ■

Let U, V be two points of $\mathbb{Z}^2$ and define $\mathcal{H}(U,V)$ as the number of hops between U and V (it is an integer). For any $R > 0$ (some inequalities are trivially true when $R \leq \sqrt{2}$), the lemma 17 and lemma 18 can be summarized as:

$$d(U,V) \leq (R-\sqrt{2})h \implies \mathcal{H}(U,V) \leq h \tag{B.1}$$

$$d(U,V) > (h-1)R \implies \mathcal{H}(U,V) \geq h \tag{B.2}$$

$$\mathcal{H}(U,V) \geq h \implies d(U,V) > (R-\sqrt{2})(h-1) \tag{B.3}$$

$$\mathcal{H}(U,V) \leq h \implies d(U,V) \leq hR \tag{B.4}$$

B.2 Bounds on Distance and Number of Hops of Points on a Lattice

Lemma 19. *If u_1 and u_2 are reduced generator vectors of a lattice $\Lambda(u_1,u_2)$, with $|u_1| \leq |u_2|$, then for any vector w such that $w = \alpha u_1 + \beta u_2$, and $\alpha, \beta \in \mathbb{Z}^2$, we have $|w| \geq \frac{3}{4}\alpha^2|u_1|^2$, and $|w| \geq \frac{3}{4}\beta^2|u_1|^2$.*

Proof: Let W the node of coordinates (α, β). We have:

$$\begin{aligned} |\vec{UW}|^2 &= \alpha^2|u_1|^2 + \beta^2|u_2|^2 + 2\alpha\beta\ (u_1 \cdot u_2) \\ &\geq \alpha^2|u_1|^2 + \beta^2|u_2|^2 - 2|\alpha||\beta|\ |u_1 \cdot u_2|. \end{aligned}$$

Since u_1 and u_2 are reduced vectors, we can use the property given in the System 5.1, we get:

$$\begin{aligned} |\vec{UW}|^2 &\geq \alpha^2|u_1|^2 + \beta^2|u_1|^2 + |\alpha||\beta||u_1|^2. \text{ Hence,} \\ |\vec{UW}|^2 &\geq ((|\alpha| - |\beta|)^2 + |\alpha||\beta|)|u_1|^2. \end{aligned}$$

Notice that this quantity does not change if we change the sign of α or of β. Thus, we assume $(\alpha \geq 0)$, $(\beta \geq 0)$, and let $f(\alpha,\beta) = (\alpha-\beta)^2 + \alpha\beta$.
By a change of variable $\beta = \frac{\alpha}{2} + \lambda$, we get:
$f(\alpha,\beta) = \frac{3}{4}\alpha^2 + \lambda^2 \geq \frac{3}{4}\alpha^2$. Similarly, we have $f(\alpha,\beta) \geq \frac{3}{4}\beta^2$. Hence the lemma. ■

Lemma 20. *Consider any transmission range $R > \sqrt{2}$, two reduced generator vectors u_1 and u_2 of the lattice $\Lambda(u_1, u_2)$, and a node W with $U\vec{W} = \alpha u_1 + \beta u_2$ for some α and β in $\mathbb{Z}$. Assume also that the point V_1 such that $U\vec{V}_1 = u_1$ is at strictly more than h hops from U. Then:*
if $|\alpha| \geq \mu(R)$ or $|\beta| \geq \mu(R)$ where $\mu(R) = \frac{2\sqrt{3}R}{3(R-\sqrt{2})}$, we have: W is strictly more than h hops away from U.

Proof: Since u_1 and u_2 are reduced, we can apply Lemma 19 and obtain:

$$\begin{aligned} |U\vec{W}|^2 &\geq f(\alpha,\beta)|u_1|^2 \\ &\geq \frac{3}{4}\alpha^2|u_1|^2, \text{ and as } \alpha \geq \mu(R) \\ &\geq \frac{3}{4}(\frac{2\sqrt{3}R}{3(R-\sqrt{2})})^2|u_1|^2 \\ &\geq (\frac{R}{R-\sqrt{2}})^2|u_1|^2 \end{aligned}$$

Since the point V_1 is strictly more than h-hop away from U, the lemma 17 implies by contradiction that $|u_1| = |U\vec{V}_1| > (R-\sqrt{2})h$. It follows that:

$$|U\vec{W}|^2 > R^2h^2$$

Applying Lemma 18, we obtain the result. ■