Thesis

Energy-Ef cient Self-Organization for Wireless Sensor Networks

by

Thomas WATTEYNE

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

Doctor of Philosophy in

Computer Science

at the

Institut National des Sciences Appliquées de Lyon

Advisors Isabelle AUGÉ-BLUM and Stéphane UBÉDA
INRIA ARES project, CITI Laboratory, INSA Lyon

Doctoral School: EDIIS

Defended on 4th november 2008 in front of the following jury

Jury:

Anne-Marie Kermarrec (Reviewer)
INRIA Senior Researcher, IRISA/INRIA-Rennes

Marcelo Dias de Amorim (Reviewer)
CNRS Researcher, University Pierre et Marie Curie (Paris 6)

Isabelle Guérin-Lassous (Examiner)
Professor at the University Lyon 1

David Simplot-Ryl (Examiner)
Professor at the University of Lille 1

Isabelle Augé-Blum (Thesis Supervisor)
Associate Professor at INSA-Lyon

Stéphane Ubéda (Thesis Supervisor)
Professor at INSA-Lyon

Dominique Barthel (Industrial Mentor)
Principal Research Scientist at Orange Labs, Grenoble
Abstract

A wireless sensor is a small electronic device capable of measuring a physical value (temperature, light, etc.) and communicating wirelessly. Because of the network’s large size (100+ nodes), communicating with the collecting node – called sink – must be done in an ad-hoc multi-hop manner, as a single sensor can only communicate with a small number of in-range sensors.

The major challenges faced by this type of networks are energy-efficiency (sensors are powered by on-board batteries) and self-organization. Self-organization protocols enable sensors to communicate with the sink while continuously adapting to topological changes because of the wireless link’s dynamic nature and because nodes may (dis)appear. The radio chip accounts for most of the energy consumed on a wireless sensor. Reducing the use of this chip does reduce the sensor’s energy consumption, yet it makes communication between sensors more complex. As a result, there is a trade-off between energy-efficiency and ease of communication – thus efficiency of the self-organization scheme.

The contributions of the thesis work are:

- We propose a medium access control protocol which avoids maintaining neighborhood tables at each sensor (this consumes energy). The proposed protocol only builds this table on demand, which provides robustness against topological changes. We combine this protocol which an energy-efficient medium access technique borrowed from literature to form a robust and energy-efficient communication protocol.

- We use virtual coordinates as a basis for self-organizing a wireless sensor network. Each sensor contains virtual coordinates which a routing protocol uses to find a multi-hop path to the sink, locally. By continuously updating these virtual coordinates, the proposed technique is robust against changes in the environment. We show that this technique has the self-organization characteristics of interest.

- We combine proposed protocols to form a cross-layered communication architecture which enables the network to self-organize in an energy-efficient way.

Results are based on analytical, simulation-based and experimental studies. They show the energy-efficiency of our proposals within the application domain of interest. Two implementations of the complete communication architecture are also presented.

Key words: ad-hoc wireless multi-hop network, wireless sensor network, MAC protocol, self-organization, energy-efficiency, analytical and experimental study.
# Contents

## Acronyms

## Notations

## Acknowledgments

### 1 Introduction

1.1 Wireless Sensor Networks ........................................ 1  
1.2 Applications and Market Opportunities .......................... 2  
1.3 Self-Organization ............................................... 4  
1.4 Energy-Efficiency .............................................. 4  
1.5 Organization of this Document ................................. 5

### 2 State of the Art

2.1 State of the Art in Medium Access Control Protocol ........ 7  
2.1.1 Preliminary Discussion ..................................... 7  
2.1.2 Taxonomy of MAC Approaches ................................. 9  
2.1.3 Framed MAC Protocols for Periodic & High-Load Traffic ... 9  
2.1.4 Contention-Based MAC Protocols with Common Active Periods 14  
2.1.5 Preamble Sampling Protocols ............................... 17  
2.2 State of the Art in Routing Techniques ....................... 21  
2.2.1 Flooding-Based Routing .................................... 21  
2.2.2 Using Geographical Information for Routing ................. 23  
2.2.3 Real, Approximate and Relative Coordinates ................ 27  
2.3 Clustering and Virtual Backbones ............................. 31  
2.3.1 Clustering Techniques .................................... 31  
2.3.2 Quantifying Organization by Means of Entropy ............... 32  
2.4 Bio-Inspired Self-Organization ................................ 33  
2.5 Open Issues .................................................. 34  
2.6 Summary ..................................................... 35

### 3 Assumptions and Models

3.1 Urban Wireless Sensor Networks ............................... 37  
3.2 WSN System Model ............................................ 38  
3.2.1 Centralized, Distributed, Localized ......................... 38  
3.2.2 Propagation Models ....................................... 38  
3.2.3 Graph Representation ...................................... 40  
3.3 Analytical, Simulation and Experimental Apparatus ........... 43  
3.3.1 Analytical Model .......................................... 43
4 1-hopMAC, Pushing Neighborhood Discovery to MAC 50
4.1 Analytical Comparison of MAC Approaches 50
4.2 Needs and Design-Drivers 57
4.3 Electing a Neighbor Based on a Metric 59
4.3.1 Description of the Different Variants 59
4.3.2 Dynamic Switching Between Variants 61
4.4 Reducing Collision Probability in Backoff-Based Election Mechanisms 63
4.4.1 Motivation 63
4.4.2 Calculating $P$ 64
4.4.3 Improving $P$ with non-Uniformly Distributed Values of $x_i$ 67
4.5 Retrieving the Neighbor List 70
4.5.1 Overview of 1-hopMACv2 70
4.5.2 Tuning the Contention Window Size 71
4.6 Summary 73
5 Geographic Routing over Virtual Coordinates 75
5.1 An Analytical Study on Clustering 75
5.2 Practicality of Geographic Routing Techniques 78
5.2.1 Impact of Loose Positioning 80
5.2.2 Impact of non-UDGs 80
5.2.3 Discussion 82
5.3 The 3-rule Routing Protocol 83
5.3.1 Distributed Exhaustive Search in an Arbitrary Graph 84
5.3.2 Introducing Geographic Information to Lower Hop Count 85
5.3.3 Simulation and Analysis 86
5.4 From Real to Virtual Coordinates 88
5.5 2D Virtual Positions 90
5.5.1 Initialization and Iterative Convergence Process 90
5.5.2 Demonstrating Network Convergence 93
5.5.3 Simulation Results and Discussion 96
5.6 1D Virtual Heights 99
5.6.1 Initialization and Iterative Convergence Process 99
5.6.2 Demonstrating Network Convergence 100
5.6.3 Simulation Results and Discussion 100
5.7 Summary 104
6 Cross-Layer Integration and Experimental Studies 106
6.1 Interaction Between MAC and Routing Layers 106
6.2 The WiFly Experiment 107
6.2.1 Overview of the Experiment 107
6.2.2 1-hopMACv1 and 2D Virtual Positions 109
6.2.3 Preliminary Analytical and Simulation Results 110
6.2.4 Experimental Results 112
6.2.5 Lessons Learnt 116
6.3 The Sense&Sensitivity Experiment 116
6.3.1 1-hopMACv2 and 1D Virtual Heights 116
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3.2</td>
<td>Overview of the Experiment</td>
<td>118</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Experimental Results</td>
<td>120</td>
</tr>
<tr>
<td>6.3.4</td>
<td>Lessons Learnt</td>
<td>133</td>
</tr>
<tr>
<td>6.4</td>
<td>Summary</td>
<td>134</td>
</tr>
<tr>
<td></td>
<td><strong>Conclusions and Future Work</strong></td>
<td>136</td>
</tr>
<tr>
<td>6.5</td>
<td>Contributions of This Thesis Work</td>
<td>136</td>
</tr>
<tr>
<td>6.6</td>
<td>Future Work</td>
<td>137</td>
</tr>
<tr>
<td></td>
<td><strong>Appendix</strong></td>
<td>141</td>
</tr>
<tr>
<td>A.1</td>
<td>List of Publications</td>
<td>141</td>
</tr>
<tr>
<td>A.2</td>
<td>Abstract [French]</td>
<td>144</td>
</tr>
</tbody>
</table>
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE</td>
<td>emergent Algorithm for highly uniform Cluster formation</td>
</tr>
<tr>
<td>AOA</td>
<td>Angle Of Arrival</td>
</tr>
<tr>
<td>AODV</td>
<td>Ad hoc On-demand Distance Vector routing protocol</td>
</tr>
<tr>
<td>BER</td>
<td>Bit Error Rate</td>
</tr>
<tr>
<td>BMAC</td>
<td>Berkeley MAC</td>
</tr>
<tr>
<td>CAGR</td>
<td>Compound Annual Growth Rate</td>
</tr>
<tr>
<td>CCA</td>
<td>Clear-Channel-Assessment</td>
</tr>
<tr>
<td>CDMA</td>
<td>Code Division Multiple Access</td>
</tr>
<tr>
<td>CENSAM</td>
<td>Center for Environmental Sensing And Modeling</td>
</tr>
<tr>
<td>CI</td>
<td>preamble-sampling Check Interval</td>
</tr>
<tr>
<td>CLDP</td>
<td>Cross-Link Detection Protocol</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CR</td>
<td>Communication Request</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>CSMA</td>
<td>Carrier Sense Multiple Access</td>
</tr>
<tr>
<td>CTS</td>
<td>Clear-To-Send</td>
</tr>
<tr>
<td>DMA</td>
<td>Direct Memory Access</td>
</tr>
<tr>
<td>DSR</td>
<td>Dynamic Source Routing protocol</td>
</tr>
<tr>
<td>DTN</td>
<td>Delay Tolerant Network</td>
</tr>
<tr>
<td>DYMO</td>
<td>DYnamic Mobile On-demand routing protocol</td>
</tr>
<tr>
<td>EtE</td>
<td>End-to-End routing process</td>
</tr>
<tr>
<td>ETSI</td>
<td>European Telecommunications Standards Institute</td>
</tr>
<tr>
<td>F-MAC</td>
<td>Frequency MAC</td>
</tr>
<tr>
<td>FDMA</td>
<td>Frequency Division Multiple Access</td>
</tr>
<tr>
<td>FMAC</td>
<td>Framelet MAC</td>
</tr>
<tr>
<td>FPA</td>
<td>Fast Path Algorithm</td>
</tr>
<tr>
<td>G-MAC</td>
<td>Gateway MAC</td>
</tr>
<tr>
<td>GBR</td>
<td>Gradient-Based Routing</td>
</tr>
<tr>
<td>GFG</td>
<td>Greedy-Face-Greedy routing protocol</td>
</tr>
<tr>
<td>GG</td>
<td>Gabriel Graph</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>GPSR</td>
<td>Greedy Perimeter Stateless Routing for wireless networks</td>
</tr>
<tr>
<td>GRAN</td>
<td>GRAdient Broadcast</td>
</tr>
<tr>
<td>GSpring</td>
<td>Greedy Embedding Spring Coordinates</td>
</tr>
<tr>
<td>GTIM</td>
<td>Gateway Traffic Indication Message</td>
</tr>
<tr>
<td>GTSNetS</td>
<td>Georgia Tech Sensor Network Simulator</td>
</tr>
<tr>
<td>HART</td>
<td>Highway Addressable Remote Transducer</td>
</tr>
<tr>
<td>HEED</td>
<td>Hybrid, Energy-Efficient, Distributed clustering</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>I-EDF</td>
<td>Implicit Earliest Deadline First</td>
</tr>
<tr>
<td>IEEE</td>
<td>today, IEEE simply stands for the letters I-E-E-E</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>ISA</td>
<td>today, ISA simply stands for the letters I-S-A</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LEACH</td>
<td>Low-Energy Adaptive Clustering Hierarchy</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>LMAC</td>
<td>Lightweight MAC</td>
</tr>
<tr>
<td>LOS</td>
<td>Line-Of-Sight</td>
</tr>
<tr>
<td>LPL</td>
<td>Low Power Listening</td>
</tr>
<tr>
<td>LPM</td>
<td>Low Power Mode</td>
</tr>
<tr>
<td>MAC</td>
<td>Medium Access Control</td>
</tr>
<tr>
<td>MANET</td>
<td>Mobile Ad hoc NETworking IETF working group</td>
</tr>
<tr>
<td>MFP</td>
<td>Micro-Frame Preamble sampling</td>
</tr>
<tr>
<td>MMSN</td>
<td>Multi-Frequency Media Access Control for Wireless Sensor Networks</td>
</tr>
<tr>
<td>MPR</td>
<td>Multi-Point Relay</td>
</tr>
<tr>
<td>MSK</td>
<td>Minimum Shift Keying</td>
</tr>
<tr>
<td>nLOS</td>
<td>non Line-Of-Sight</td>
</tr>
<tr>
<td>NS-2</td>
<td>Network Simulator</td>
</tr>
<tr>
<td>OLSR</td>
<td>Optimized Link-State Routing protocol</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection</td>
</tr>
<tr>
<td>PDF</td>
<td>Probability Density Function</td>
</tr>
<tr>
<td>PEDAMACS</td>
<td>Power-Efficient and Delay Aware Medium Access Control protocol</td>
</tr>
<tr>
<td>PER</td>
<td>Packet Error Rate</td>
</tr>
<tr>
<td>PKI</td>
<td>Public Key Infrastructure</td>
</tr>
<tr>
<td>PLL</td>
<td>Phase Locked Loop</td>
</tr>
<tr>
<td>ppm</td>
<td>parts per million</td>
</tr>
<tr>
<td>PR-MAC</td>
<td>Path-oriented Real-time MAC protocol</td>
</tr>
<tr>
<td>RFC</td>
<td>Request-For-Comments</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio Frequency IDentifier</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality-of-Service</td>
</tr>
<tr>
<td>RISC</td>
<td>Reduced Instruction Set Computer</td>
</tr>
<tr>
<td>ROLL</td>
<td>Routing Over Low-power Lossy links</td>
</tr>
<tr>
<td>RSS</td>
<td>Received Signal Strength</td>
</tr>
<tr>
<td>RSSI</td>
<td>Received Signal Strength Indicator</td>
</tr>
<tr>
<td>RTS</td>
<td>Request-To-Send</td>
</tr>
<tr>
<td>RX</td>
<td>Reception</td>
</tr>
<tr>
<td>SDO</td>
<td>Standards Developing Organizations</td>
</tr>
<tr>
<td>SINR</td>
<td>Signal-to-Interference and Noise Ratio</td>
</tr>
<tr>
<td>SMAC</td>
<td>Sensor MAC</td>
</tr>
<tr>
<td>SMACS</td>
<td>Self-organizing Medium Access Control protocol for Sensor networks</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>STEM-B</td>
<td>Sparse Topology and Energy Management MAC protocol, Beacon mode</td>
</tr>
<tr>
<td>STEM-T</td>
<td>Sparse Topology and Energy Management MAC protocol, Tone mode</td>
</tr>
<tr>
<td>TC</td>
<td>Topology Control</td>
</tr>
<tr>
<td>TDMA</td>
<td>Time Division Multiple Access</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>TI</td>
<td>Texas Instruments, Inc.</td>
</tr>
<tr>
<td>TinyOS</td>
<td>Tiny Operating System</td>
</tr>
<tr>
<td>TMAC</td>
<td>Timeout MAC</td>
</tr>
<tr>
<td>TOA</td>
<td>Time Of Arrival</td>
</tr>
<tr>
<td>ToF</td>
<td>Time-of-Flight</td>
</tr>
<tr>
<td>TRAMA</td>
<td>TRaffic-Adaptive Medium Access protocol</td>
</tr>
<tr>
<td>TSMP</td>
<td>Time Synchronized Mesh Protocol</td>
</tr>
<tr>
<td>TX</td>
<td>Transmission</td>
</tr>
<tr>
<td>U-WSN</td>
<td>Urban Wireless Sensor Network</td>
</tr>
<tr>
<td>UDG</td>
<td>Unt Disk Graph</td>
</tr>
<tr>
<td>UWB</td>
<td>Ultra-Wide Band</td>
</tr>
<tr>
<td>VCap</td>
<td>Virtual Coordinate Assignment Protocol</td>
</tr>
<tr>
<td>WOR</td>
<td>Wake On Radio</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Networks</td>
</tr>
</tbody>
</table>
Notations

General Variables:

- $D_{mf}$: duration of a micro-frame
- $T_{mf}$: duration between two successive micro-frames (period)
- $D_{cca}$: duration of a Clear-Channel-Assessment
- $D_{data}$: duration of a message, in seconds.
- $D_{rtscts}$: duration of the RTS/CTS phase is SMAC
- $D_{sync}$: duration of synchronization phase is SMAC
- $D_{turn}$: time a radio takes for switches between transmission and reception
- $T_{sync}$: period of time after which nodes need to be resynchronized
- $D_{max}$: maximum allowed de-synchronization between nodes
- $\theta$: frequency tolerance of the time-base crystal
- $N$: number of nodes in the network
- $d$: density, i.e. average number of neighbors of a node
- $R$: communication radio range of a node
- $\eta$: radio duty-cycle
- $\{x\}$: 1D coordinate of a node
- $\{x, y\}$: 2D coordinates of a node

Variables used in the comparison of MAC approaches (Section 4.1):

- $L$: total length of the message, including the preamble
- $x$: distance – in meters – to the sink nodes
- $r(x)$: average number of retries for a message before sent successfully
- $M_{ps}$: total number of transmission attempts over the network in one second
- $E_{msg}$: amount of radio on-time for a 1-hop neighborhood ($d + 1$ nodes)
- $P_{idle}$: radio on-time per second and per node when a node sits idle
- $\alpha$: total number of message generated on the network during one second
- $h$: minimum hop count to the sink, i.e. topological distance
- $N(h)$: number of nodes of height $h$
- $\rho(x)$: rate of messages crossing the co-centric circle of radius $x$
- $\rho(h)$: rate of messages relaying by all the nodes at height $h$
- $\bar{m}$: rate of messages sent in a 1-hop neighborhood
- $\lambda$: message arrival rate in a buffer model
- $\mu$: message transmission rate in a buffer model
- $T_{active}$: Duration separating two successive active periods in SMAC
- $\Delta$: average delay a message experiences between source and sink
Variables used in the description of $1-hopMAC_{v1}$ (Section 4.3):

- $TR$: total radio on-time of a 1-hop neighborhood running 1-hopMAC
- $\beta$: metric contained in a node
- $\beta_i$: metric of node $i$
- $\beta_{\text{min}}$: smallest possible metric
- $\beta_{\text{max}}$: largest possible metric
- $\beta_{\text{first}}$: metric of the node which answers first
- $\beta_{\text{thresh}}$: threshold metric used to switch between 1-hopMAC variants
- $N$: number of nodes in the 1-hop neighborhood (=10)
- $D$: length of the contention window
- $d$: message length (=10)
- $x_i$: node $i$’s sending instant
- $x_{\text{first}}$: smallest value among all $x_i$
- $\Delta t$: small duration used to concert a metric in a backoff time ($x = \beta \cdot \Delta t$)
- $P$: collision probability
- $D_{x_i}$: probability distribution function of $x_i$ in $[0 \ldots D]$
- $D_{x_{\text{first}}}$: probability distribution function of $x_{\text{first}}$ in $[0 \ldots D]$
- $\mathcal{F}$: mapping function, $x_i = \mathcal{F}(\beta_i)$
- $c$: constant value

Variables used in the description of $1-hopMAC_{v2}$ (Section 4.5):

- $CW$: Duration of a Contention Window
- $CT$: Total contention time

Variables used when quantifying organization by means of throughput (Section 5.1):

- $M$: number of edges in the network
- $C$: number of clusters in the network
- $\Theta$: throughput
- $\xi$: ratio between the data rate of clusterhead-clusterhead links and clusterhead-node links
- $W$: physical bandwidth
- $\bar{L}$: average distance between a node and the sink node
- $\bar{r}$: average distance between neighbor nodes
- $\bar{h}$: average number of hops between a node and the sink node
Acknowledgments

The thesis work was made possible thanks to a research contract between the Centre d’Innovation en Télécommunications et Intégration de Services (CITI), INSA Lyon and France Telecom R&D/Orange Labs, Grenoble, France, partially funded by a CIFRE subsidy from the French ANR.

I thank Mrs. Anne-Marie Kermarrec, INRIA Senior Researcher at IRISA/INRIA-Rennes and Mr. Marcelo Dias de Amorim, CNRS Researcher at the University Pierre et Marie Curie (Paris 6) for having accepted to review my thesis.

I thank Mrs. Isabelle Guérin-Lassous, Professor at the University Lyon I and Mr. David Simplot-Ryl, Professor at the University of Lille 1 for serving as examiners of my thesis, and for their interest in my work. I thank David Simplot-Ryl for the very inspiring discussions on geographic and other exotic routing techniques throughout my thesis.

I thank Dr. Isabelle Augé-Blum and Prof. Stéphane Ubéda for having given me the opportunity to study in the CITI Laboratory. I deeply thank Isabelle for her accurate advice during the 4 years we have worked together.

I thank the team at France Telecom for the great working environment. I especially thank Mr. Dominique Barthel for his expert eye and constant support.

I thank Prof. Kris Pister from UC Berkeley and Prof. Christian Ibars from CTTC for hosting me at their respective laboratories.

I thank the undergraduate students I have had the privilege of supervising for their help, and especially Michaël Gauthier and Clément Burin des Roziers for their true engineering skills.

I thank Dr. Mischa Dohler, now at CTTC but previously at France Telecom R&D. They simply don’t make advisors better than this.
CHAPTER 1

Introduction

1.1 Wireless Sensor Networks

Wireless multi-hop networks are composed of nodes equipped with a wireless extension, capable of communicating together in an ad hoc fashion. This type of communication has the following characteristics. Unlike cellular networks or single hop wireless networks (including IEEE802.11 networks [Society, 2007]), they do not rely on a fixed infrastructure. This implies that the network self-organizes to form a coherent communicating entity. As nodes may move, be switched off, or enter/leave the network, the communication protocols governing the network should be able to cope with these topological changes without human intervention. Most of the nodes are too far away from the collecting node (called sink node) to communicate directly. Intermediate nodes are hence used to relay the message, in what is called ad hoc multi-hop communication.

The unreliable nature of the wireless link may cause the topology to change even when there is no change in the position/presence of nodes. Nodes may also be added or removed from the network, causing topological changes.

Sensor networks have been researched and deployed for decades already; their wireless extension, however, has witnessed a tremendous upsurge in recent years [Akyildiz et al., 2002, Akyildiz and Kasimoglu, 2004, Culler et al., 2004]. This is mainly attributed to the unprecedented operating conditions of WSNs, i.e. a potentially enormous amount of sensor nodes reliably operating under stringent energy constraints. WSNs allow for an un tethered sensing of the environment. It is anticipated that within a few years, sensors will be deployed in a variety of scenarios, ranging from environmental monitoring to health care, from the public to the private sector, etc.

They will be battery-driven and deployed in great numbers in an ad hoc fashion, requiring communication protocols and embedded system components to run in an energy efficient manner. Energy-efficiency is of utmost importance, as it is closely related to the network lifetime. Prior to large-scale deployment, however, a gamut of problems has still to be solved which relates to various issues, such as the design of suitable software and hardware architectures, development of communication and organization protocols, validation and first steps of prototyping, until the actual commercialization. In contrast to known and well understood systems, however, WSNs bear some fundamental design differences [Culler et al., 2004]:

- **Applications**: The gamut of applications is vast, thus requiring very different solutions to be developed for different applications. This problem is further enhanced due to the stringent energy constraints, requiring subtle solutions to be developed for different requirements.

- **Number of Nodes**: The number of nodes involved can be very large, where current roll-out examples include a few hundreds; however, roll-out expectations are in the range of a few thousand nodes communicating simultaneously. This is also atypical to any wireless system today, and hence poses new technological as well as social and environmental challenges.

- **Deployment**: Similar to the applications, nodes can be deployed in many different ways, from randomly dropping them from a helicopter, to carefully positioning them onto a water meter. In any case, the deployment strategy is largely driven by the application, not by the resulting communication topology. A WSN needs to be able to cope with the whole range of possible topologies.
• **Energy:** WSNs are nowadays battery powered and, because changing batteries in a few thousand nodes on a regular basis is clearly impractical, they are required to have a long lifetime and are hence considered to be highly constrained in energy. This is in contrast to other systems, where nodes are usually either powered by the mains or easily rechargeable on a regular basis.

This means that, unlike incumbent systems, wireless sensor networks need to be:

- highly scalable (protocols ought to work with arbitrary numbers of nodes);
- highly energy efficient (at all layers and functionalities); and
- highly application tailored (efficient for given task).

Although WSNs are a truly cross-disciplinary research topic, one major challenge is to enable network-wide communication. Because batteries can not be replaced, communication protocols ought to be as energy-efficient as possible. This poses a challenge as it implies turning off the radio chips most of the time. Unlike traditional wireless networks, source and destination nodes are not within communication range most of the time, and intermediate nodes need to relay the message. In this multi-hop scheme, new problems arise such as handling the increased load due to the relay process, or finding a suitable multihop path between source and destination. In most envisioned applications, newly deployed nodes need to acquire routing information before being able to participate in the relaying process. In such an ad hoc setting, self-organizing algorithms are thus of utmost importance. These algorithms must be able to recover from major topological changes due to links dynamics of nodes (dis)appearing. The cost of an individual node need to be kept low; wireless sensors therefore embed low-throughput, low-cost and ultra-low power radio chips. Radio links are extremely unreliable in nature, an extra burden the self-organization scheme needs to cope with. Finally, as networks are deployed in uncontrolled environments, potential security threats appear such as node tampering, false data injection or man-in-the-middle attacks. This thesis focuses on the communication part of a WSN, more specifically on medium access control, routing and self-organization principles.

### 1.2 Applications and Market Opportunities

The WSN market is at an early stage, and is still fragmented with a number of participants and applications. In 2005, the revenue generation by WSNs in 2005 is evaluated to $159.9 million, a number which is expected to grow to $1,853.9 million by 2012. The Compound Annual Growth Rate (CAGR, the year-over-year growth rate of an investment over a specified period of time) for the market of WSNs is evaluated to 40.1%. This high number suggests that WSNs have a large potential market impact [Viswanathan, 2006]. WSNs experience a slow rate of adoption that is expected to increase through end user awareness. Customers have a low interest in technology and standards, and are mainly interested in comfort within personal network communications.

Geographically, 35.2% of the market is dominated by North America for a total market of $56.3 million in revenues. By 2012, however, the share of North America and Europe in the WSN market is expected to decrease to 31.2% and 28.9% respectively. Production bases will most likely be shifted to the Asia Pacific region.

Wireless products are adopted as a consumer accessory in many parts of Western Europe and the Asia Pacific regions, for different reasons. Demand in the European market is being driven by regulations due to EU standards implementation. An example is the EU regulation which requires all new and renovated homes to be equipped with smoke detectors by 2012. A WSN
company may see this as an opportunity to come up with a smoke detecting WSN which would relay the alarm directly to the fire brigade through the home’s Internet Service Provider (ISP). The Asia Pacific market exhibits the highest growth rate with large projects of urban development and infrastructure construction which embed wireless monitoring of manufacturing facilities. An example is the CENSAM (Center for Environmental Sensing And Modeling) project in Singapore which works on deploying a city wide water quality monitoring system.

Before WSNs are largely adopted by the market, the following challenges need to be addressed. Today, interoperability of multi-vendor equipment thwarts market penetration. Moreover, there is a serious lack of adequate open bandwidth, which impedes market reach\(^1\). Finally, the constantly evolving standards bother wireless equipment manufacturers who may restrain from developing commercial products with standards which are not established yet [Dohler and Watteyne, 2008, Dohler et al., 2008a].

Traditionally, there are four main approaches to product development, namely, norms and standards, forums and associations, proprietary solutions and open source. These approaches trade time-to-market for development costs, obedience to regulations and intellectual property.

Standards Developing Organizations (SDOs) create normative standardization documents which are elaborated by a community publicly available without any discriminatory conditions. Vendors benefit from standards because they can access markets more easily with standard-compliant products, at the risk of blurring competitive differentiation. Customers benefit because they can access a wide range of services without the burden of being tied to a given service provider or technology. Standardization efforts pertinent to WSNs are the IEEE (link and physical layer solutions), ETSI (complete machine-to-machine solutions), ISA (regulation for control systems) and the IETF (routing and network solutions).

The IEEE usually standardizes the PHY layer of the transmitter, and medium access protocol rules. IEEE standards applicable to WSNs are IEEE802.15.4 (technology used by ZigBee and IETF 6LowPan), IEEE 802.15.1 (technology used by Bluetooth/WiBree) and IEEE 802.11x (technology used by WiFi). The Internet Engineering Task Force (IETF) comprises more than 120 active working groups organized in 8 areas. IETF working groups of interest for WSNs are IETF 6LoWPAN (focusing on end-to-end IPv6 connectivity in WSNs) and IETF ROLL (focusing on routing and self-organization).

Forums and Associations working on WSNs include ZigBee, WiBree (ultra low power communication based on Bluetooth technology), Wavenis OSA (lead by Coronis Systems, focuses on metering and urban monitoring) and WirelessHART (an extension of Highway Addressable Remote Transducer – HART, a leading technology in industrial control).

Current proprietary solutions can not inter-operate, but companies behind these solutions are lobbying in SDOs or forums. These companies include Dust Networks (based in California, specialized in highly-reliable industrial control), ArchRock (also from California, goes for home automation), CrossBow (from California, with an emphasis on asset management and tracking) and Coronis Systems (based in Montpellier, France, focuses on urban automated meter reading).

The market of WSNs is young; nevertheless, general trends can be seen by observing the entities involved. Companies have started reusing the work developed by academia and realigning the focus to simple and functional solutions. While the dream of sensor so small that they could be considered "smart dust" is in most entrepreneurs’ minds, current commercial solutions are disarmingly intuitive.

---

\(^1\)This is not true in Japan and China who have both allocated frequencies bands for wireless sensor-like networks. There are two PHY customization efforts going on in the IEEE802.15.4 effort: 802.15.4c is studying the possibility of creating a PHY targeted at the newly opened 314-316 MHz, 430-434 MHz, and 779-787 MHz bands in China; 802.15.4d is defining an amendment to the specification to support the new 950-956 MHz frequency band in Japan.
1.3 Self-Organization

In [Elson and Estrin, 2004], the authors present their vision of what a WSN will be capable of doing in 2053. That year, the authors imagine, a major earthquake hits the Mojave desert in Southern California. Luckily, a few dozen of seismometers – those closest to the epicenter – first sense the unusual acceleration in the ground and warn the population of Los Angeles about an imminent major earthquake, within a few tens of a second over a multi-hop wireless path. Buildings are evacuated and thanks to the timely alert, major human losses are avoided. To rescue those stuck under collapsed buildings, fireman disperse thousands of tiny sensors which self-organize to form a coherent network. A map of the structure appears on a screen at the rescue truck. People are visible as heat sources, chemical sensors detect abnormal traces of natural gas. Four days later, so say the authors, Southern California returns to normal. The 2053 quake came and went, thanks largely to the pervasive sensors.

Although this story can be considered a fairy tale, it stresses how important self-organization is in a wireless multi-hop environment. Communication rules need to turn the set of individual communicating sensors into a coherent wireless network. The ultimate goal is to obtain a fully autonomous network. Self-organization covers various notions, often referred to as self-* [Prehofer and Bettstetter, 2005]. A self-organizing network should offer the user a deploy-and-forget experience. An end user should be able to deploy his network without requiring specific networking knowledge. The network should hence self-configure, i.e. tune a set of parameters as a function of its environment without requiring human intervention. Wireless links are unreliable in nature, and nodes may be removed or fail due to battery exhaustion, as well as added by the user. The network should therefore be self-healing.

Self-organization will be formally defined and related work discussed in Section 2.3.

1.4 Energy-Efficiency

Different node powering mechanisms are available, such as non-rechargeable battery, rechargeable battery with regular recharging (e.g. sunlight) [Jiang et al., 2005], rechargeable battery with irregular recharging (e.g. opportunistic energy scavenging), capacitive/inductive energy provision (e.g. active RFID), always on (e.g. powered electricity meter).

The vast majority of current and envisioned roll-outs are battery powered, in which minimizing the total energy consumption increases network lifetime. In Table 1.1, we list the battery types used to power some popular WSN platforms. These platforms have between 8 and 20 kJ of energy available.

<table>
<thead>
<tr>
<th>platform</th>
<th>battery</th>
<th>energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>EM2420, eZ430-RF2500</td>
<td>2×AAA/LR03</td>
<td><a href="mailto:1600mAh@1.5V">1600mAh@1.5V</a> (2400mWh / 8.6kJ)</td>
</tr>
<tr>
<td>MICAx, Telos</td>
<td>2×AA/LR06</td>
<td><a href="mailto:4000mAh@1.5V">4000mAh@1.5V</a> (6000mWh / 21.6kJ)</td>
</tr>
<tr>
<td>WSN430</td>
<td>Varta PolyFlex</td>
<td><a href="mailto:830mAh@4.0V">830mAh@4.0V</a> (3320mWh / 12.0kJ)</td>
</tr>
</tbody>
</table>

Table 1.1: Energy source in current platforms.
CHAPTER 1. INTRODUCTION

Figure 1.1: Energy consumption of the WSN430 node in active and sleeping states.

1.5 Organization of this Document

We propose an energy-efficient and self-organizing communication architecture. We show the robustness of our solution as well as its applicability to urban WSNs through two experimental studies. The remainder of this thesis is organized as follows:

The second chapter presents the state of the art, organized by communication layer. It starts by the medium access control techniques identified in literature (framed, synchronized and preamble sampling techniques) before giving an overview of routing protocols in WSNs, with a particular focus on geographic routing. A discussion on self-organization gives the reader some insights on how protocol design for large scale WSNs can be influenced by observing large-scale biological systems.

The third chapter details the assumptions and models used in this thesis. We first describe the urban WSN application scenario, its constraints and challenges, before detailing the WSN system model used (including propagation and graph models). We present the analytical and simulation tools as well as the experimental apparatus used.

The fourth chapter presents 1-hopMAC, a MAC protocol which avoids constructing and maintaining neighborhood tables. If does so by (1) adopting a fully on-demand approach and (2) shifting neighborhood discovery down to the medium access control (MAC) layer for energy-efficiency. Two versions are proposed, pros and cons of which are evaluated against existing MAC protocols.

The fifth chapter focuses on geographic routing techniques. It starts by showing how current techniques fail when hard-to-meet assumptions are not met. It then proposes to use the sequence of already traversed nodes to help the routing algorithm decide which neighbor should be next on the multi-hop route to destination. This novel routing technique functions for any arbitrary
random graph, and guarantees delivery. We show that this protocol discovers paths with a hop count comparable to existing geographic routing solutions, while functioning over a much larger spectrum of topologies.

We propose to use geographical routing over coordinates which are not geographic, but which rather reflect the node’s location in the topology. Virtual coordinates organize in such a way that a geographic routing protocol running on top of them discovers optimal paths in terms of number of hops. We show how this virtual coordinate-based self-organization scheme allows nodes and sink nodes to be added, removed, and relocated without any configuration.

The sixth chapter integrates the contributions made into a complete self-organizing communication architecture. The resulting solution is implemented by two experimental studies. The first aims at showing the validity of the virtual coordinate solution under the extreme case of a fast moving mobile sink node. The second mimics an urban WSN deployment, and shows how the data rearranges as sink nodes are added and removed from the network.
Wireless Sensor Networks have emerged as an application domain of wireless ad hoc networks. WSNs, as ad hoc networks, use wireless multi-hop communication; the difference lies mainly in the different sets of constraints. Energy-hungry ad hoc solutions seemed unsuitable for WSNs, and research has shifted towards optimizing medium access. This communication layer controls the radio chip, hence the energy consumption of the node. Similarly, resource-hungry ad hoc routing concepts such as hierarchical and flooding-based techniques appear inefficient with WSNs. Geographical routing protocols have recently emerged as a promising candidate for routing in WSNs.

This chapter presents a comprehensive overview over the state of the art in WSN communication protocols. We start by exploring the MAC protocols before digging into routing solutions. We then show how protocols can be inspired by (self-organizing) biological systems. We end this chapter by discussing open issues.

2.1 State of the Art in Medium Access Control Protocol

Because it deals with two key constraints, the MAC layer arguably is pivotal in a WSN communication architecture [Langendoen and Halkes, 2005, Demirkol et al., 2006]. First, as detailed in Chapter 3, it controls the state of the radio chip, hence the duty cycle and the energy-efficiency of the node. Second, since the wireless medium is broadcast in nature, it is in charge of resolving any arising contention while taking into account link outages and changes of topologies due to nodes (dis)appearing.

There has hence been a growing interest in understanding and optimizing WSN MAC protocols in recent years. Research was driven primarily to reducing energy consumption because of the limited and constrained resources on-board a wireless sensor. This section provides a comprehensive state-of-the-art study by exposing the prime focus of WSN MAC protocols, design guidelines that inspire these protocols, as well as drawbacks and shortcomings of existing solutions. We also address how existing and emerging technology is influencing future solutions.

2.1.1 Preliminary Discussion

Before describing key proposed protocols, some terms and key techniques need to be introduced.

**TDMA, CDMA, FDMA.** TDMA – for Time Division Multiple Access – slices time into slots. By assigning a slot to only one node, collisions are avoided. This is the basis for the MAC protocol family presented in Section 2.1.3 we call ‘frame based mac protocols’. Frequency can also be sliced in non-interfering frequency bands, a technique called Frequency Division Multiple Access, FDMA. This is the basis for frequency-diverse MAC protocols, which are described later on. Finally, Code Division Multiple Access – CDMA – uses orthogonal codes which allow different signals to be sent at the same time/frequency while being received successfully.

**Aloha, CSMA.** In contention-based MAC protocols, multiple nodes compete to grab the same channel. If two nodes end up sending at the same time, their messages may collide at the receiver
and they need to resend. In the simplest case – called Aloha [Abramson, 1985] – a node sends a message whenever it wants, and hopes for the best. Its efficiency, however, is limited to a disappointingly low 18%\(^1\). Carrier Sense Multiple Access (CSMA) is more sophisticated in that nodes listen to the medium to see whether it is busy before sending a packet. If so, the node backs off and simply retries later.

**Hardware considerations.** Low-cost radio chips cannot listen and transmit to/on the wireless channel at the same time, a situation referred to as half-duplex communication. A chip can be either transmitting a packet, receiving a packet, idle listening to the wireless medium without receiving anything, or switched off. Its energy consumption varies greatly depending on its state. As shown in Table 2.1, switching the radio chip off reduces the energy consumption by about three orders of magnitude. As a rule of thumb, transmitting or (idle) listening to the medium consumes about the same amount of energy (see Table 2.1). PLL stands for Phase Locked Loop.

Table 2.1 shows that turning the radio on lasts in the order of 100\(\mu\)s, while turning it around lasts a few tens of \(\mu\)s. Turnaround times can become a problem when dealing with CSMA MAC protocols. Two nodes sense the channel to see whether it is free before sending a packet. If the instants they sense the channel are closer to each other than the turnaround time, they both think the channel is free and their messages may collide.

**Hidden nodes problem.** Even with null turnaround times, messages can collide. Take the simple topology in Fig. 2.1. Nodes A and C cannot hear one another. Even with CSMA, they always see the medium free, and may end up sending packets to B at the same time. This situation has been addressed and solved in IEEE802.11 [Society, 2007] by RTS/CTS handshaking. If A wants to send a message to B, it first sends a very short Request-to-Send (RTS) message to which B responds with a very short Clear-To-Send (CTS) message. Upon overhearing this CTS, B’s neighbors refrain from transmitting for the duration of the to-be-sent data message contained in the RTS/CTS messages. The channel is virtually reserved at the receivers neighbors and the hidden node problem is avoided.

Two reasons refrain WSN MAC protocols from using RTS/CTS messages. First, RTS/CTS is effective only if the data message is much longer than the RTS/CTS messages themselves. Whereas the maximum size for an IEEE802.11 packet is 2304 bytes, typical WSN applications require packet of less than 100 bytes. Second, for the algorithm to be effective, all nodes should

---

\(^1\)Slotted Aloha, a variant where messages can only be sent inside fixed sized time windows called slots, increases efficiency to about 36%.

---

<table>
<thead>
<tr>
<th></th>
<th>cc2420</th>
<th>cc2500</th>
<th>cc1100</th>
<th>cc1101</th>
</tr>
</thead>
<tbody>
<tr>
<td>power down</td>
<td>20(\mu)A</td>
<td>900nA</td>
<td>900nA</td>
<td>500nA</td>
</tr>
<tr>
<td>receiving/idle listening</td>
<td>18.8mA</td>
<td>15.7-18.8mA</td>
<td>15.1-16.4mA</td>
<td>15.6-16.9mA</td>
</tr>
<tr>
<td>transmitting at -10dBm</td>
<td>11mA</td>
<td>11.1mA</td>
<td>13.5mA</td>
<td>13.1mA</td>
</tr>
<tr>
<td>transmitting at 0dBm</td>
<td>17.4mA</td>
<td>21.5mA</td>
<td>16.9mA</td>
<td>16.8mA</td>
</tr>
<tr>
<td>PLL turn-on time</td>
<td>(&lt; 192\mu s^a)</td>
<td>88.4(\mu s)</td>
<td>88.4(\mu s)</td>
<td>88.4(\mu s)</td>
</tr>
<tr>
<td>PLL RX/TX settling time</td>
<td>(&lt; 192\mu s^a)</td>
<td>9.6(\mu s)</td>
<td>9.6(\mu s)</td>
<td>9.6(\mu s)</td>
</tr>
<tr>
<td>PLL TX/RX settling time</td>
<td>(&lt; 192\mu s^a)</td>
<td>21.5(\mu s)</td>
<td>21.5(\mu s)</td>
<td>21.5(\mu s)</td>
</tr>
</tbody>
</table>

\(^a\)192\(\mu s\) is the upper-bound the IEEE802.15.4 standard imposes for PLL settling and turn-on times. The actual value can be lower when measured directly from the hardware.
CHAPTER 2. STATE OF THE ART

Figure 2.1: The hidden node problem. Node A and C are hidden to one another; B may become exposed.

have their radios on to overhear the RTS/CTS exchange. As an always-on MAC protocol is energy-prohibitive, RTS/CTS are only used in contention-based MAC protocols with common active periods.

**Real-time communication.** Applications with timeliness constraints are called "real-time". When an event happens, the network should transmit the corresponding alarm message within a known and bounded delay. Taking more time can have a potentially (very) negative impact. Real-time applications are classified in two categories. Hard real-time applications are such that the bounded delay absolutely needs to be met. Examples include radioactive leakage detection in a nuclear power plant. Soft real-time applications are looser in that they can tolerate that a fraction of the messages do not meet their deadline. Examples include multimedia applications.

Real-time communication protocols for WSNs are scientifically at the crossroads of two distinct worlds: distributed real-time applications and wireless networking. Some real-time MAC protocols for WSNs are inspired by industrial wired networks. Because constraints of WSNs and wired LAN settings are very different, these protocols sometimes put unrealistic constraints which severely limit their applicability. Routing solutions emerging from the world of wireless networking are more functional, but they present only soft real-time characteristics.

**Frequency-diverse solutions.** Current low power radio chips enable the center frequency to be programmed within a range of several tens of MHz. As an example, on a CC2500 radio chip, the 250 kBaud MSK modulation occupies 296kHz of bandwidth [CC2500, 2007]; adjacent carrier frequencies can hence be safely positioned as close as 500kHz apart. With an operating frequency range of 2400-2483.5MHz, there is potentially room for over 150 non-overlapping channels. In single-channel MAC protocols, all nodes are configured to use a single frequency all the time. Frequency-diverse MAC protocols switch between multiple frequencies at run-time.

All channel partitioning techniques have the same effects of reducing collisions at the cost of a reducing adaptability to varying traffic. TDMA techniques have been exploited extensively in WSNs. FDMA techniques have become practical as recent radio chips are able to switch between frequency channels fast (e.g. in less than 100μs). FDMA has an advantage over TDMA in that it can be made robust against narrow-band interference.

2.1.2 Taxonomy of MAC Approaches

All energy-efficient MAC protocols switch the radio off to save energy, while switching it on every now and then to communicate. Different approaches have been taken, which we classify in three families: frame-based (scheduled) protocols, contention based protocols with common active periods and preamble sampling protocols [Bachir et al., 2008].

2.1.3 Framed MAC Protocols for Periodic & High-Load Traffic

Periodic and high-load traffic is most suitably accommodated by means of reservation-based protocols. Such protocols construct a schedule which all nodes follow and are sometimes referred
to as TDMA protocols. A schedule is a succession of slots which form a frame; the frame continuously repeats over time. Note that slots can be generalized to partition the available resource (channel) along the time, frequency of code axis, or any combination thereof. When a node needs to send a message, it waits for its slot during which it knows no other node is transmitting. This approach is attractive because, once the schedule is set up, it can significantly reduce the number of collisions, the amount of idle listening and overhearing. This approach offers bounded latency, fairness and good throughput in loaded traffic conditions, at the cost of reducing adaptability of variable traffic.

Data can be scheduled in different ways into slots. When communication links are scheduled, specific sender-receiver pairs are assigned to a given slot. This avoids both overhearing and collisions, but may decrease network throughput if traffic is variable. Slots can be assigned to senders: during its slot, a node is given the opportunity to transmit to any of its neighbors, requiring all its neighbors to listen. The opposite is also possible (i.e. scheduling receivers), in which case multiple nodes may end up competing to send, requiring contention resolution techniques.

Although framed MACs are appealing because they provide levels of quality of service other techniques can not offer, this comes at a price. WSNs are infrastructure-less, so network-wide synchronization and scheduling is a complex challenge. Protocols may rely on a central scheduling node, but this limits this technique to smaller supervised networks such as industry automation WSNs. Wireless links are unreliable, causing network topologies to change over time. With links and nodes coming and going, maintaining a network-wide consistent schedule is hard. Finally, collision-free scheduling requires a node to be aware of all nodes multiple hops away\(^2\), which uses a large memory footprint.

**Centralized scheduling at the sink.** G-MAC (Gateway MAC [Brownfield et al., 2006]) elects a node acting as a gateway for a certain time, and then rotates nodes in order to balance load. The TDMA frame of G-MAC contains three periods: the collection period, the traffic indication period and the distribution period. During the collection period, nodes contend for the channel and send packets expressing their future traffic needs. In the traffic indication period all nodes wake up and listen to the channel to receive the Gateway Traffic Indication Message (GTIM). The GTIM maintains synchronization among nodes and assign slots to nodes.

BitMAC [Ringwald and Römer, 2005] constructs a spanning tree rooted at the sink. Each node acts as an access point for its direct children. Both Power Efficient and Delay Aware Medium Access Control protocol for Sensor networks (PEDAMACS) [Sinem and Varaiya, 2006] and Time Synchronized Mesh Protocol (TSMP) [Pister, 2008]) also use a centralized scheduler.

**Distributed scheduling.** SMACS (Self-organizing Medium Access Control for Sensor networks) [Sohrabi et al., 2000] allows nodes to establish a communication infrastructure between neighboring nodes by defining transmission and reception slots using 2 phases:

- Neighbor Discovery: A node wakes up randomly and listens for a given time to invitation messages. If it does not received such a message, it invites other nodes by sending a new invitation message;

- Channel Assignment: Different links use different channels to reduce collisions.

Outside the established transmission-reception slots, nodes sleep to save energy. The main advantage is that SMACS is easy to implement as slots are formed on the fly. SMACS, however,

\(^2\)The exact number of hops depends on the characteristics of interference.
CHAPTER 2. STATE OF THE ART

suffers from a relatively high energy consumption. Moreover, broadcast messages are not taken into account natively, and are replaced by a series of unicast messages.

**TRAMA** [Rajendran et al., 2006] (TRaffic-Adaptive Medium Access protocol) determines a collision-free scheduling and performs link assignment according to the expected traffic. The protocol contains two phases: localized topology formation and scheduled channel access. The scheduled channel access allows each node to wake up only to transmit or to receive, which reduces idle listening and overhearing to zero. The main issue with TRAMA is its complexity and the assumption that nodes are synchronized network-wide.

**EMAC** [van Hoesel and Havinga, 2004a] defines three types of nodes: active, passive and dormant. Only active and passive nodes participate in communication, dormant nodes may be those that run out of energy or those that are recharging their batteries. Active nodes participate in all of the operations and the passive nodes can only exchange information with a corresponding active node.

Time is sliced up into frames, and further into slots. Each slot contains three parts: CR (Communication Request), TC (Topology Control), and DATA. In this TDMA schedule, each active node owns a slot. In the CR part of its slot, an active node listens for incoming requests from the passive nodes attached to it. In the TC part, an active node transmits acknowledgments to its passive nodes, synchronization information, and a table containing the schedule information.

A passive node attaches itself to an active node and follows it. The passive node spends its time in sleep mode to save energy; it only wakes up to transmit a CR to its corresponding active node or to receive a TC from it. As a passive node determines its corresponding active node independently of the others, more than one passive node may choose the same active node. This may result in a collision when more than one passive node sends a CR packet to the same active node at the same time. If the active node has no data to transmit during its DATA part, it allows other nodes to contend for transmitting data during its DATA part.

Transmissions from an active node are announced in the TC packets. All active nodes listen to all their neighboring active nodes. The active nodes should form a connected dominating set to ensure connectivity. A dominating set means that each node in the network is either a member of the dominating set or it has a neighbor that belongs to the dominating set.

Slot assignment is distributed: each active node transmits the slots used by itself and by its neighbors in bitmapping style to its neighbors. This allows neighbors to construct a local topology and to select slots that are not being used in the two-hop neighborhood to avoid collisions.

The first drawback of EMACs is that active nodes are permanently active and thus cannot save energy. Although EMACs envisages rotating roles, the resulting dominating sets with such forced rotation might not be optimal, thus leading to more energy consumption by more nodes.

**Real-time communication.** The Implicit Earliest Deadline First (I-EDF) protocol [Caccamo et al., 2002] assumes the network is organized in regular hexagonal cells, with a router node in the middle of each cell. FDMA is used in inter-cell communication to avoid conflicts between cells, and TDMA is adopted in intra-cell transmission. The nodes inside each cell are assumed to be fully connected, which means a node can transmit messages to any other node in a single hop. Although it guarantees bounded end-to-end delay, its main drawback is that the hexagonal cell structure is extremely constraining. Moreover, the protocol is not energy-efficient, and assumes all nodes can communicate using 7 different frequencies.

Another solution is to assume the sink node knows the full network topology, learnt for example during an initialization phase. With this information, it is able to build a network-wide TDMA schedule. Power Efficient and Delay Aware Medium Access Control protocol for Sensor networks (PEDAMACS [Sinem and Varaiya, 2006]) assumes the sink has a transmission power strong enough for its message to be heard by any node in the network. The sink node uses this link
to send the global TDMA schedule determined after learning the topology. Messages sent by the sensor nodes follow a multi-hop path to the sink.

**RT-LINK** [Mangharam et al., 2007] is TDMA-based and applicable to networks which require predictability in throughput, latency and energy consumption. Hardware-based global time synchronization is used. Two phases, namely topology-gathering and scheduling, are included in RT-LINK. A cycle is defined as the duration between two synchronization pulses, and consists of a large number of frames divided into two parts: scheduled and contention slots. Each node that wants to transmit data periodically sends Hello messages by randomly selecting a slot within the contention slots. A Hello message is transmitted in a multi-hop manner to the sink node, which is responsible for network-wide slot assignment. The node is active in its assigned slot(s).

**PR-MAC** [Chen et al., 2007] (Path-oriented Real-time MAC protocol) is used for monitoring applications where data is sent periodically. A sensor node starts by sending a message to the sink using a (non real-time) contention-based MAC protocol. This message contains a description of the sensed value, and the path taken by the message. Using the reverse path, the sink nodes send a series of control messages to the relaying nodes, which indicate the periodicity of the subsequent messages and act as resource reservation messages. Once all relaying nodes are contacted, the path is set up and the sink node can expect data message to reach it in a real-time fashion. Although I-EDF, PEDAMACS, TR-LINK and PR-MAC set up real-time communication, they are based on a central controller.

To our knowledge, only **DUAL-MODE** [Watteyne et al., 2006a] creates a schedule in a distributed way. It is designed for linear networks. The protocol switches between two modes: unprotected (contention based) and protected (contention-free). Alarm messages are transmitted in "unprotected mode" as long as there are no collisions. Each node relays a message when a backoff time proportional to its distance to the sink elapses (and no other message is heard). In case of collision, the protocol switches to protected mode, which avoids collision by channel reservation. Because the unprotected mode allows for faster transmission, the protocol switches back to this mode when possible. This protocol does not require synchronization and constructs a schedule in a fully distributed manner. Yet, besides it only focusing on linear networks, it is not energy-efficient.

**Frequency-diverse solutions.** Note that we use the term frequency and channel interchangeably.

**Multichannel LMAC** [Inccl et al., 2006] proposes to enhance the TDMA-based LMAC with multi-channel support. In **LMAC** (Lightweight MAC) [van Hoesel and Havinga, 2004b], nodes in a 2-hop neighborhood decide on a TDMA schedule in a distributed way, assigning different slots to different nodes. When the density of the network gets very high (e.g. a two-hop neighborhood is composed of tens of nodes) all slots end up assigned, and new nodes end up without slots and are unable to communicate. [Inccl et al., 2006] proposes for those nodes to pick a slot on another frequency. The number of potential slots is roughly multiplied by the number of frequency channels. This protocol hence allows more nodes to communicate in LMAC.

Multi-Frequency Media Access Control for Wireless Sensor Networks (**MMSN** [Zhou et al., 2006]) uses an initial frequency assignment phase. Frequencies are assigned evenly to the nodes of a 1-hop neighborhood, with nodes learning their neighbors’ frequency. A node uses the destination’s frequency when transmitting, and its own frequency when receiving. During network run-time, nodes are synchronized and time is sliced up into slots. A backoff-based CSMA algorithm solves contention between nodes in a given frequency/time slot. An interesting proposal is to toggle snooping/transmission, in which nodes listen/transmit on both their own and the destination’s frequency to make sure they are not receiving a packet while transmitting to a neighbor (this requires a radio chip with fast PLL RX/TX and TX/RX settling time).

**Y-MAC** [Kim et al., 2008] is primarily designed to decrease latency. Nodes are synchronized and reception slots are assigned to each node on a common base channel. In case multiple packets
need to be sent between neighbor nodes, successive packets are sent each on a different frequency following a pre-determined hopping sequence. This hopping sequence starts at the base channel. As a result, bursts of messages ripple across channels, significantly reducing latency. The presented implementation results serve as proof-of-concept for the multi-channel MAC approach.

TSMP [Pister, 2008] (Time Synchronized Mesh Protocol) is TDMA-based and hence requires network-wide synchronization. Access is controlled by means of a tunable amount of time slots which form a frame. The protocol is designed such that a node can participate in multiple frames at once allowing it to have multiple refresh rates for different tasks. In addition, TSMP employs FDMA and frequency hopping. Different links use differing frequency slots and the same link hops during its life time across different frequency slots. This yields high robustness against narrow-band interference and other channel impairments.

A traditional approach to facilitate synchronization is beaconing. Longer frames decrease synchronization refresh rate and power consumption; shorter frames invoke the opposite. TSMP nodes maintain a sense of time by exchanging resynchronization messages during active periods together with the usual data and acknowledgment packets; this invokes negligible overhead. TSMP nodes are active in three states: 1) sending a packet to a neighbor; 2) listening for a neighbor to talk; and 3) interfacing with an embedded hardware component. The duration of active periods, i.e., the duty cycling, is very flexible in TDMA; typical applications require duty cycles of less than 1% on average.

When applied, the sink typically retrieves the list of nodes, their neighbors and their requirements in terms of traffic generation. From this information, it constructs a scheduling table in both time and frequency. When implementing TSMP on IEEE 802.15.4 compatible hardware, 16 frequency channels are available. Exemplified by the scheduling table of Fig. 2.2, the TSMP link establishment and maintenance rules are simple:

- never put two transmissions in the same time/frequency slot;
- at a given time, a given node should not receive from two neighbors nor have to send to two neighbors.

Assuming that slots are 10ms long and node $H$ sends a packet following route $H \rightarrow F \rightarrow B \rightarrow G$, then $H$ send to $F$ in slot $[t_5, ch.6]$, thereafter $F \rightarrow B$ in $[t_{10}, ch.11]$, then $B \rightarrow G$ at $[t_{18},ch.8]$. Latency can in this case be reduced to 3 slots or 30ms. Fig. 2.2 shows that successive packets sent between two nodes are sent using different frequencies, following a preset hopping sequence.

We would like to stress that TSMP uses multi-channel not to increase network throughput, but to increase robustness against narrow-band interference\(^3\). Fig. 2.2 shows that successive packets sent between two nodes are sent using different frequencies, following a preset hopping sequence. [Doherty et al., 2007] presents experimental results in which 44 nodes were deployed running TSMP, including retransmission mechanisms, for 28 days in a printing facility. A delivery ratio of over 99.999% is reported.

Multi-channel hardware support largely impacts MAC layer design. It increases the network’s throughput by reducing contention; additional channels can be introduced when a common base channel becomes too crowded [Incé et al., 2006, Kim et al., 2008, Le et al., 2008]. Multi-channel can also be used to increase robustness against narrow-band long-lasting [Le et al., 2008] and transient [Pister, 2008] interference. Whereas most current multi-channel solutions assume nodes are synchronized and time is split into time slots [Pister, 2008, Incé et al., 2006, Zhou et al., 2006, Kim et al., 2008], this assumption is broken in recent solutions [Le et al., 2008] (to be described).

\(^3\)The same could be achieved by using larger band radios, but this would reduce the number of usable independent channels and hence increases latency.
Proof-of-concept experiments [Le et al., 2008, Kim et al., 2008] and commercial products [Pister, 2008] show the great potential of multi-channel MAC protocols for WSNs.

### Discussion

Table 2.2 summarizes the different framed MAC protocols described in this section. Framed MAC protocols are the family which can guarantee Quality of Service such a (hard) real-time communication or reliability. Setting up and maintaining a network-wide schedule costs energy and time. This technique is used primarily for high-traffic and relatively small WSNs. Typically applications include industrial monitoring.

#### 2.1.4 Contention-Based MAC Protocols with Common Active Periods

In a contention-based MAC protocol with common active periods, nodes define common active/sleep periods. Active periods are used for communication; sleep periods are used for saving energy. During active periods, nodes contend for the channel using Carrier Sense Multiple Access (CSMA). Note that this family of MAC protocols requires the nodes to maintain a certain level of synchronization to keep active/sleep periods common to all nodes.

The periodic nature of the active/sleep state of the networks makes these MAC protocols particularly suitable for applications in which traffic is periodic. Examples include monitoring and applications in which keep-alive messages are periodically exchanged to ensure network reliability.

The disadvantage is that nodes need to re-synchronize every now and then due to clock drift. The nodes’ clocks are based on crystals which frequency changes due to manufacturing or temperature. Typical clock drift is 10 ppm, meaning that a clock gets desynchronized by 10 time units every $10^6$ time units. As two clocks may drift in opposite directions (one going 10 ppm slow, the other 10 ppm fast), the relative clock drift between two nodes is double the absolute clock drift. Protocols such as S-MAC (to be presented) can only function if the nodes’ clock are within $500 \mu s$ from each other (as reported in [Van Dam and Langendoen, 2003]). Node hence need to resynchronize every $\frac{500 \mu s}{2 \times 10^{10} ppms} = 25 s$.

In S-MAC [Ye et al., 2002, Ye et al., 2004] (Sensor MAC), all nodes wake-up at the same time (Fig. 2.3). Upon waking up, nodes wait for a time randomly picked in $[0...D_{sync}]$ before
## Chapter 2. State of the Art

### Function Protocols

<table>
<thead>
<tr>
<th>Centralizing Scheduling at the Sink</th>
<th>G-MAC [Brownfield et al., 2006]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BitMAC [Ringwald and Römer, 2005]</td>
</tr>
<tr>
<td></td>
<td>PEDAMACS [Sinem and Varaiya, 2006]</td>
</tr>
<tr>
<td></td>
<td>TSMP [Pister, 2008]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distributed Scheduling</th>
<th>SMACS [Sohrabi et al., 2000]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DUAL-MODE [Watteyne et al., 2006a]</td>
</tr>
<tr>
<td></td>
<td>TRAMA [Rajendran et al., 2006]</td>
</tr>
<tr>
<td></td>
<td>EMAC [van Hoesel and Havinga, 2004a]</td>
</tr>
</tbody>
</table>

| Frequency-diverse solutions | Multichannel LMAC [Incet al., 2006] |
|                            | MMSN [Zhou et al., 2006] |
|                            | Y-MAC [Kim et al., 2008] |
|                            | TSMP [Pister, 2008] |

<table>
<thead>
<tr>
<th>Real-time communication</th>
<th>I-EDF [Caccamo et al., 2002]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PEDAMACS [Sinem and Varaiya, 2006]</td>
</tr>
<tr>
<td></td>
<td>RT-LINK [Mangharam et al., 2007]</td>
</tr>
<tr>
<td></td>
<td>PR-MAC [Chen et al., 2007]</td>
</tr>
<tr>
<td></td>
<td>DUAL-MODE [Watteyne et al., 2006a]</td>
</tr>
<tr>
<td></td>
<td>TSMP [Pister, 2008]</td>
</tr>
</tbody>
</table>

### Table 2.2: Framed MAC protocols for Periodic & High-Load Traffic

Sending a synchronization message. A node cancels its transmission if it hears another synchronization message. After $D_{\text{sync}}$, another window of length $D_{\text{rtscts}}$ is created for nodes to agree on data messages that will be exchanged, using Request-To-Send (RTS) and Clear-To-Send (CTS) messages. Nodes which are not participating in any message exchange go back to sleep after the RTS/CTS phase, other nodes exchange their messages before returning to sleep.

Using common active periods has some disadvantages. Determining the optimal size of the active periods requires taking into account the trade-off between idle listening and collisions. Short active periods reduce idle listening while increasing contention and thus collision rates. Long active periods do the opposite, they reduce contention at the cost of increased idle listening. SMAC uses a fixed pre-calculated size for active periods that is optimized for an expected workload. This makes the protocol rigid, by ignoring spatial or temporal imbalance. Besides rigidity, SMAC

![Figure 2.3: S-MAC, an example of a contention-based MAC protocol with common active periods.](image-url)
suffer from extra end-to-end delay because of the mandatory sleeping phases. The difficulty faced by improvements on SMAC is hence to find a suitable trade off between sleep delay and optimal active periods.

**Mitigating rigidity.** TMAC (Timeout MAC [Van Dam and Langendoen, 2003]) follows up on the basic idea introduced by SMAC that consists in using common active/sleep schedules: nodes determine their active/sleep schedules in a way similar to SMAC. TMAC alleviates SMAC’s rigidity by proposing an adaptive duty cycle in which the duration of active periods is no longer fixed but varies according to the traffic.

The key idea of TMAC consists in making a node switch its radio off before the active period ends, in case it does not expect any traffic. By downsizing active period lengths, TMAC saves more energy than SMAC. The proportion of this saving depends on the amount of time cut back on the initial active period duration. To optimize the sleep period durations, TMAC moves all communications to a burst at the beginning of active periods. A node can determine that there is no communication going on in the remainder of an active period if no activation event occurs within the duration $T_A$. An activation event may be, for instance, the reception of a frame or sensing some noise considered as collision. The minimum duration of $T_A$ should be long enough to span the maximum contention duration and the RTS/CTS exchange.

In variable workloads, TMAC saves about five times more energy than SMAC does. However, this is achieved at the cost of an increased latency and thus reduced throughput. Although TMAC improves on SMAC’s energy savings, it still suffers from the main problem of the high cost of maintaining common active/sleep schedules via exchanging SYNC packets.

**Minimizing sleep-delay.** Adaptive Listening [Ye et al., 2004] suggests the use of overhearing to reduce the sleep delay. In adaptive listening, the node that overhears its neighbor’s transmission and learns from it the duration of that transmission may sleep in the meantime and then wake up just when the transmission ends. This idea has also been proposed in nanoMAC [Ansari et al., 2007]. Here, the node wakes up after that transmission even if it might happen during its sleep period. This makes it possible for the node’s neighbor to immediately send data to it, instead of waiting for the node’s next scheduled active time.

FPA (Fast Path Algorithm [Li et al., 2005]) makes nodes wake up for an additional time, even during their pre-scheduled sleep periods, to ensure timely relaying of frames. A node uses its hop distance from the sender to estimate when its upstream neighbor will send a frame to it. Then, the node wakes up at the estimated time only to receive and potentially forward the frame to its downstream neighbor. The node sets these additional wakeup times from information piggybacked in the first data packet on that path.

**Real-time communication.** To our knowledge, F-MAC [Roedig et al., 2006] (framelet MAC) is the only contention-based MAC protocol guaranteeing bounded bandwidth and delay without requiring synchronization. F-MAC protocol uses a framelet approach: fixed sized frames are retransmitted a fixed number of times with a specific retransmission period (i.e. the amount of time between two successive retransmissions). The period of retransmission of the framelets varies from node to node. With its neighbors using different periods, a given node is able to successfully receive at least one framelet from a source even when all its neighbors are transmitting. Moreover, nodes can be grouped into clusters, each clusters having its own framelet period. The biggest drawback of this approach is its poor bandwidth utilization as the same information is sent in multiple framelets. Furthermore, the worst case delay increases exponentially with the number of nodes within the same collision domain. This protocol is not suitable for large scale and dense sensor networks.
**Table 2.3: Contention-Based MAC protocols with Common Active Periods**

<table>
<thead>
<tr>
<th>function</th>
<th>protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canonical Solution</td>
<td>S-MAC [Ye et al., 2002, Ye et al., 2004]</td>
</tr>
<tr>
<td>Mitigating Rigidity</td>
<td>TMAC [Van Dam and Langendoen, 2003]</td>
</tr>
<tr>
<td>Minimizing Sleep-Delay</td>
<td>Adaptive Listening [Ye et al., 2004]</td>
</tr>
<tr>
<td></td>
<td>FPA [Li et al., 2005]</td>
</tr>
<tr>
<td>Frequency-diverse solutions</td>
<td>Le’s solution [Le et al., 2008]</td>
</tr>
<tr>
<td>Real-time communication</td>
<td>F-MAC [Roedig et al., 2006]</td>
</tr>
</tbody>
</table>

**Frequency-diverse solutions.** Le’s solution [Le et al., 2008] is to our knowledge the only multi-frequency MAC protocol which does not assume nodes are synchronized. The protocol dynamically assigns channel to nodes, and groups nodes sharing a channel into clusters. As in [Incel et al., 2006, Kim et al., 2008], nodes all start at the same base channel. By periodically exchanging status messages measuring the loss ratio, nodes can detect when too much contention/interference is experienced on their channel. Clusterheads then take the initiative to hop to the next available channel, followed by the other nodes in their cluster. Inter-cluster communication is done by temporarily changing to the destination’s channel. The resulting mechanism is transparent to the application and routing layers and it efficiently minimizes cross-channel communication while maximizing same-channel traffic. Throughput increase of as much as 50% are reported in dense networks. Although nodes do not need to be synchronized, they need to broadcast status messages to their neighbors frequently (every 1 second in the presented experiment). Energy-efficiency may thus be an issue under this setting. Other than increasing throughput, the solution elegantly copes with narrow-band long lasting interference. In case of a permanent interference on a given channel, bad status reports cause the nodes to hop to another channel, increasing the nodes’ robustness.

**Discussion.** Table 2.3 summarizes the different contention-based MAC protocols with common active periods described in this section. Contention-based MAC protocols with common active periods are particularly attractive for application with periodic traffic. The traffic load should be high enough that synchronization messages can be piggybacked on data messages. Typical applications include home automation.

**2.1.5 Preamble Sampling Protocols**

Unlike the other two families, nodes using preamble-sampling are not synchronized. Instead, nodes periodically listen for a very short time (called Clear-Channel-Assessment, or CCA) to decide whether a transmission is ongoing. We call check interval (CI) the amount of time a node waits between two successive CCAs. The sender needs to make sure the receiver node is awake before sending data; it thus prepends a (long) preamble to its data. By having the preamble at least as long as the wake-up period, the sender is certain the receiver will hear it and be awake for receiving the data.

While this technique has been referred to in literature as Cycle Receiver [Lin et al., 2004], Low-Power Listening [Polastre et al., 2004] or Channel Polling [Ye et al., 2006], we call it preamble sampling.

Figs. 2.4, 2.4 and 2.6 are chronographs depicting the radio state of node S and its three neighbors A, B and C for different preamble-sampling variants described in this section. A box above/under a vertical line means the node’s radio is transmitting/receiving, respectively. No box means the radio is off. All nodes sample the channel for $D_{cca}$ seconds every CI seconds.
CHAPTER 2. STATE OF THE ART

Basic preamble-sampling. Fig. 2.4 depicts basic preamble-sampling functions: node $S$ sends a continuous preamble of length $CI + D_{cca}$. When nodes $A$, $B$ and $C$ sample the channel, they stay awake until the data message of length $D_{data}$ is sent.

Reducing energy-consumption at the receiver. Basic preamble-sampling requires $A$, $B$ and $C$ to listen to the remainder of the preamble, which costs energy. Micro-Frame Preambling (MFP [Bachir et al., 2006]) cuts the preamble into a series of micro-frames (see Fig. 2.5). Each micro-frame contains a counter indicating how many micro-frames still remain. A micro-frame is sent every $T_{mf}$ seconds, and lasts for $D_{mf}$. Upon sampling the channel, a node knows how many microframes are still to be sent, and hence returns to sleep until the actual data is sent.

Reducing energy-consumption at the transmitter. One major drawback of preamble-sampling is that preambles are long, which costs energy and increases collision probability. Techniques have been proposed to overcome this problem.

X-MAC [Buettner et al., 2006] uses a concept similar to MFP. The sender $S$ cuts the preamble into micro-frames, and listens between each micro-frame (see Fig. 2.6). Note that $S$ needs a time $T_{turn}$ to switch between reception and transmission modes. When the destination node (here $B$) hears the preamble, it answers with an acknowledgment message of length $D_{ack}$. This causes the length of the preamble to be, on average, half that of MFP.

WiseMAC [El-Hoiydi et al., 2004] aims at letting each node learn about its neighbors’ wakeup times; if the transmitter knows the wakeup time of the receiver, then it can timely start its trans-
mission just to meet the receiver wakeup. A passive approach is adopted: nodes piggyback their
wakeup schedule in the acknowledgment message when a transmission takes place. Each node
maintains a neighbor table containing the wake-up schedule together with a timestamp. Clock
drifts may make the transmitter lose accuracy about the receiver’s wakeup time.

The older the timestamp, the more clocks may have drift, hence the longer the preamble needs
to be. The preamble, however, never needs to be larger than $CI$ which constitutes the preamble
worst case length. Let’s assume a clock drift of $10$ ppm and $CI = 100$ ms. As clocks may drift in
opposite directions, two nodes drift $100$ ms apart if they have not communicated for
$10^3 \times 10^{-6} = 5000$ s (= 1.5 hour). Under lower loads (for example in an urban WSN where a node sends data
only every 24 hours), WiseMAC is useless as preambles will systematically be of their maximum
size, i.e. $CI$.

Avoiding unnecessary receptions. In basic preamble sampling, nodes overhear the data mes-
sage destined to other nodes, which costs energy. MFP removes overhearing by putting the desti-
nation’s address into a micro frame. Only the destination node wakes up when data is transmitted,
the other nodes just sleep through that period. MFP efficiently reduces the amount of time over-
listening.

In 1-hopMAC [Watteyne et al., 2006b], nodes answer a request with a time proportional to
some metric; the node which answers first is elected next hop node by the routing process. This
cross-layer improvement on preamble-sampling avoids the need to maintain neighbor tables. This
will be detailed in Chapter 4.

Other techniques. Other techniques have been proposed on top of preamble sampling, such
as the use of distinct wakeup and data channels to increase throughput (Sparse Topology and
Energy Management, STEM [Schurgers et al., 2002]), or improving the quality of CCA by outlier
detection (Berkeley MAC, BMAC [Polastre et al., 2004]).

Hardware support. Radio chips come in different flavors, trading flexibility for energy-efficiency.
The stronger a standard, the more likely a chip manufacturer will implement some of its func-
tionalities in hardware in order to reduce energy consumption. Radios such as the TI CC2420
comply with the IEEE802.15.4 standard; while this reduces flexibility, it greatly increases energy-
efficiency as the micro-controller can stay asleep while the radio chip handles parts of the commu-
nication algorithm in hardware.
Different types of preamble-sampling MAC protocols can be implemented on different radio chips. Simpler, more flexible radios can send a sequence of zeros and ones of an arbitrary length, making them suitable for implementing basic preamble sampling techniques where the preamble is monolithic. IEEE802.15.4 compliant radios limit the size of a packet to 128 bytes, making it hard to send long monolithic preambles. These radios are, however, very efficient at sending micro-framed preambles.

Because of its energy-efficiency, preamble-sampling has become such a popular MAC technique that industry giants such as Texas Instruments have started making radio chips which take care of the preamble sampling in hardware. Examples include CC1100, CC1101 and CC2500 chips. They can be programmed to perform a CCA periodically, and to send an interrupt to the microcontroller when they sense either power levels above a threshold, or when bits of data can be demodulated. While the radio-chip takes care of the sampling, the microcontroller can go into a deep sleep, consuming almost nothing.

Discussion. We summarize the protocols described in this section in Table 2.4.

There is an optimal value for the \( C1 \) beyond which nodes waste more energy in transmission than they save in reception. Finding this optimal value depends mainly on the traffic load on the network. As an example, let’s consider 10 nodes that are all within communication range, and sample the channel for \( 200 \mu s \) every \( C1 \). Without traffic, the average duty cycle is \( \frac{200.10^{-6}}{C1} \), so the larger \( C1 \), the more energy efficient the protocol is. Consider now that \( \alpha \) messages are sent between the 10 nodes per second. We thus have a cumulative \( \alpha \times C1 \) of transmission time and \( \alpha \times 9 \frac{C1}{2} \) of reception time (the 9 neighbors of the sender node listen to the preamble on average for \( \frac{C1}{2} \) seconds). With traffic, the average duty cycle hence becomes \( \frac{200.10^{-6}}{10C1} + \alpha C1 \frac{11}{20} \). We depict the duty cycle in Fig. 2.7 for several loads. The larger the load, the smaller the duty cycle should be.

A key to understanding preamble-sampling techniques is that it costs more to send than to receive data. This is because the sender needs to transmit a long preamble, while the receiver (especially when preamble are cut into micro-frames) only switches on its radio to receive the data.

Preamble-sampling is good at reducing the energy consumption as no control traffic is needed, making it particularly suitable for low-throughput applications. The counterpart, however, is that

---

Table 2.4: Preamble Sampling Protocols

<table>
<thead>
<tr>
<th>function</th>
<th>protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic preamble-sampling</td>
<td>Cycle Receiver [Lin et al., 2004], LPL [Polastre et al., 2004], Channel Polling [Ye et al., 2006]</td>
</tr>
<tr>
<td>Reducing energy-consumption at the receiver</td>
<td>MFP [Bachir et al., 2006]</td>
</tr>
<tr>
<td>Reducing energy-consumption at the transmitter</td>
<td>X-MAC [Buettner et al., 2006], WiseMAC [El-Hoiydi et al., 2004]</td>
</tr>
<tr>
<td>Avoiding Unnecessary Receptions</td>
<td>MFP [Bachir et al., 2006]</td>
</tr>
<tr>
<td></td>
<td>1-hopMAC [Watteyne et al., 2006b]</td>
</tr>
<tr>
<td>Other techniques</td>
<td>STEM [Schurgers et al., 2002] (two channels)</td>
</tr>
<tr>
<td></td>
<td>BMAC [Polastre et al., 2004] (improve CCA)</td>
</tr>
</tbody>
</table>

---

\(^4\)For simplicity, we consider that the length of the data message is negligible compared to the length of the preamble.
Figure 2.7: In preamble sampling, the optimal check-interval largely depends on the network load.

2.2 State of the Art in Routing Techniques

As the deployment area can be large, nodes are often too far away from the destination sink node to report their messages directly. Routing is the communication software layer which is responsible for finding a sequence of intermediate nodes to relay the message to the sink node (layer 3 in the Open Systems Interconnection – OSI – model [Tanenbaum, 2002]).

We chose to classify routing protocols in three families. The first one uses flooding as a basic building block, an efficient technique for e.g. a small group of wirelessly connected mobile robots. The second group of routing protocols assumes nodes know their geographical position and uses this information to find a multi-hop path to the sink node. Because this assumption is hard to meet, the third protocol family combines geographical-inspired routing protocols with positioning techniques which inform the nodes of their topological rather than geographic coordinates [Akkaya and Younis, 2005, Al-Karaki and Kamal, 2004].

2.2.1 Flooding-Based Routing

Delay Tolerant Networks (DTNs) are a very specific class of networks in which the application supports large delivery delays of the sensed data. Examples include the water quality monitoring WSN deployed in a lake for multiple months, presented in [McDonald et al., 2007]. As the ring buoys (equipped with sensing and communicating devices) are too far apart to communicate, a ship is used to navigate from buoy to buoy, retrieving data as it floats by. Delays between sensing instant and data retrieval instant are typically in the order of days.

When nodes are close enough to form a mesh, DTNs approaches are still usual when the high mobility of the nodes prevents the use of the more traditional routing approaches to be described in this section. In case nodes move very fast, messages can be exchanged using hot-potato rout-
ing, in which pair of nodes exchange packets opportunistically when they are within reach. The message eventually reaches its destination [Khelil et al., 2005]. In [Leguay et al., 2006], Leguay et al. analyze the delivery ratio when using different DTN approaches over the mobility traces of laptop users logged at the Dartmouth College campus, NH, USA [Henderson et al., 2004].

The concept of near-random path is exploited in rumor routing [Braginsky and Estrin, 2002]. A number of messages called agents continuously wander inside the network, advertising a node’s sensing activity. This approach is used only to advertise the nodes’ activity; another routing protocol needs to be used to transmit the actual data across the network.

Link state and distance vector protocols are the two classes of routing protocols used in today’s Internet [Tanenbaum, 2002]. Efforts have thus been made to translate those protocols to the wireless multi-hop world.

Optimized Link State Routing protocol (OLSR [Jacquet et al., 2001]) is a link state protocol for ad hoc network; it uses network-wide flooding to inform the nodes of changes in the topology. OLSR optimizes flooding by having only a subset of the nodes relay the broadcast packet. Each node maintains a set of nodes called its Multipoint Relays (MPR), which are the node’s one-hop neighbors whose connectivity covers the two-hop neighborhood. The use of MPR can greatly reduce the control traffic in dense networks, as the number of MPRs should not increase significantly as a network becomes denser.

In Ad hoc On Demand Vector routing (AODV) [Perkins and Royer, 1999] and DSR [Johnson et al., 2001] (Dynamic Source Routing), the sensor network sits idle most of the time; only when the sink node issues requests, messages are sent to reply to these requests. Data aggregation techniques are then used by the WSN to combine the partial responses issued by individual nodes while they flow back to the sink; this creates a final response [Madden et al., 2005]. During the flooding phase – when the request is broadcast to the network –, nodes label the neighbor node they received the request from as their parent; during the response phase – when response messages are traveling back to the sink – nodes use their parent node as next hop. Although not different in essence, Dynamic Mobile On-Demand routing (DYMO) is an evolution of AODV with differences mainly in implementation issues. The concept of reverse path forwarding is also used in Directed Diffusion [Intanagonwiwat et al., 2000]. In reverse path forwarding, messages flow back to the sink by climbing up the tree of parents. In the protocols presented so far, nodes have no explicit knowledge of where they are in the tree of parents.

Most of the protocols described in the previous paragraphs were proposed by the IETF MANET (Mobile Ad hoc NETwork) working group, and have undergone the process of standardization. While the more recent DYMO is still at the stage of Internet-Draft, OLSR, AODV and DSR have reached the state of Request-For-Comments (RFC). These protocols are most suitable for ad hoc networks, where large quantities of data needs to be transmitted on a small number of not-so-energy-constrained mobile nodes. The IETF work group ROLL (Routing Over Low-power Lossy links) has been chartered in April 2007 to eventually come up with routing techniques applicable to WSNs.

Gradient based routing (GBR) [Faruque et al., 2005, Han et al., 2004, Powell et al., 2005] takes the concept of gradients one step ahead by building permanent gradients. During a setup phase, the sink node issues a message containing a counter set to 1. When receiving this message, a node sets an internal variable called height to the counter in the message, increments this counter by one, and relays the message to its neighbors. This sets up a network-wide permanent gradient. Once the gradient is set up, routing can start. Each node keeps a neighbor table containing the height of its neighbors, and sends a message to its neighbor with smallest height. A message therefore
follows the gradient towards the sink much like an impatient hiker would follow the steepest slope towards the bottom of a crater.

GBR is a promising solution for WSNs. Using timers, the gradient can be set up by having each node send only one packet [Ye et al., 2001a], hence proving the solution to be energy-efficient. As long as the connectivity graph does not change (i.e. link do not (dis)appear), the gradient ensures messages reach the sink in the minimum number of hops. Its height tells a node how many hops separate it from the sink.

[Faruque and Helmy, 2003] uses gradients in a slightly different way. Instead of centering on the sink node (enabling messages to flow towards the sink), Faruque et al. use the natural diffusing nature of a some sensed phenomenon such as temperature to build an implicit gradient. This enables the flow of data to be directed towards the area where sensing is highest. In the case of fire detection in a building, where a WSN is coupled to the network of sprayers, this technique would enable the latter to spray extinguishing fluid proportionally to how vigorous the fire is. This, in turn, would avoid computer rooms to be destroyed by the fluid when there is no fire in that area.

As links are unreliable, it may be wise to send multiple copies of the same message to increase the message delivery ratio. In this case, a band of nodes between source and sink nodes all relay the message; the larger the band, the higher the delivery ratio (and the energy consumed). This is the idea behind GRAdient Broadcast (GRAB [Ye et al., 2001b, Ye et al., 2005]). A gradient is set up using previously described techniques. During run-time, each message contains a credit field which is decremented at each hop, and a message is discarded when the cost reaches 0. The message’s credit is decremented at each hop by the difference between the heights of the nodes on that link. The source node controls the width of the band of relaying nodes by allocating more of less credit to the message it issues.

The height of a node can be modulated by other factors than hop count to the sink. In [Faruque et al., 2005, Han et al., 2004] a node with low battery increases its height so that messages flow around it, relayed by nodes with more energy. Liu et al. [Liu and Abu-Ghazaleh, 2006] use a function of the node’s neighborhood to modulate its height. This results in lesser nodes having the same height and a smoother gradient.

### 2.2.2 Using Geographical Information for Routing

As stressed by [Karl and Willig, 2005] and [Dohler et al., 2007a], while Wireless Sensor Networks and ad hoc networks are both wireless multi-hop networks, they are different in mainly three aspects: (1) energy-efficiency is a primary goal for WSNs, (2) in most envisioned applications, the amount of data transported by a WSN is low and (3) all information flows towards a limited number of destinations in WSNs. We describe how WSN protocols can efficiently exploit location information to enable network-wide communication.

Applications for WSNs are foreseen in a large range of domains [Culler et al., 2004]. In the example case of a city-wide automated water meter reading WSN, nodes are attached to each home’s water meter and report the daily consumption to the local water supplier. Knowledge of the physical location of the water meter is not useful as long as the latter can be identified. On the other hand, when considering a WSN used for tracking the location of lions in a National Park, having the location of the sending node in a reported message is essential.

If the application requires the nodes to know their location, there is no overhead to reuse this location information for communication purposes. This is the philosophy behind geographic routing, which uses the knowledge of a node’s position together with the positions of its neighbors and the sink node to elect the next hop.
CHAPTER 2. STATE OF THE ART

Different ways of defining distance to destination.

Figure 2.8: Greedy geographic routing.

(b) Geographic routing fails when a message is sent from A to F.

Figure 2.9: An example of a void area. The plain circle depicts the communication range of node S. The dotted circle is shown for readability only, it is centered at D and has radius $|DS|$. It shows that no neighbor node of S is closer than S to D.

Greedy geographic routing protocols is the simplest form of geographic routing [Stojmenovic and Olariu, 2005, Finn, 1987]. When a node receives a message, it relays the message to its neighbor geographically closest to the sink. Several definitions of proximity to destination exist. We use Fig. 2.8 (a) as a basis for our description, where node S wants to send a message to node D. Most-forward within radius considers the position of a node’s projection on a line between source and destination. In Fig. 2.8 (a), node S would choose A as closest to D. Another definition considers the Euclidian distance to destination (in this case, S would choose B). Finally, a last variant, also known as myopic forwarding, chooses the node with smallest deviation from the line interconnecting source and destination (node C in Fig. 2.8 (a)).

Irrespective of the definition of proximity, greedy routing can fail. In Fig. 2.8 (b), if a message is sent from node A to X, it reaches X with a number of hops close to optimal. Consider now the message is sent from C to X. C sends it to F, its neighbor closest to X. F, however, has no neighbor closer to X than itself; the same message ends up at a local minimum, or void area. A void area (or simply void) is depicted in Fig. 2.9. It appears when a node has no neighbor closer than itself to the destination. A greedy geographic routing algorithm fails when it reaches a void.

The occurrence of such failures depends on the topology. In Fig. 2.10, we present simulation results obtained by randomly scattering nodes in a 1000x1000 area. Each node has a circular communication area of radius 200. We tune the number of nodes to obtain a desired average number of neighbors and measure the delivery ratio. Results are averaged over $10^5$ runs. For our
Figure 2.10: Delivery ratio for different routing protocols. Note that results for the GFG and 3rule protocols coincide at 1, which is the best possible case.

simulations, the source and sink nodes are chosen randomly and change at each run. A ratio equal to 1 means that all sent messages are received. Fig. 2.10 shows results for GFG and the 3rule routing protocols which are described later.

Delivery ratio is close to 1 for very high densities because the probability of having void areas decreases as the number of nodes increases. For typical WSN densities (5-10 neighbors), over 20% of sent messages are not received because of this flaw in the routing protocol.

**Geographic routing with guaranteed delivery.** More advanced geographic routing protocols guarantee delivery under the assumption of reliable links and nodes. The key idea of these protocols is to switch between two modes. The default mode uses the greedy approach described above. In case this mode fails, a second mode is used to circumnavigate the void area. Once on the other side of this void area, the greedy mode can be resumed.

Bose et al. propose Greedy-Face-Greedy (GFG) routing [Bose et al., 1999], which uses exactly this principle. We use Fig. 2.11 to exemplify our explanation. A message is sent from node $S$ to $D$. Upon arriving at a void area (node $A$ has no neighbors closer than itself to $D$), GFG switches from *greedy mode* to *face mode*. Face mode is used to circumnavigate the void. When the current node is closer to destination than the node initially starting the face mode (here node $B$), the protocol returns to greedy mode – the void is considered circumnavigated. In face mode, a node only considers the edges between itself and its neighbors which are on the planar Gabriel Graphs [Gabriel and Sokal, 1969]. Among these neighbors, it chooses the next hop using the left hand rule. The left hand rule consists in “rolling” to the left along the edges. Note that GFG has been reinvented by Karp and Kung and called GPSR [Karp and Kung, 2000].

Graph planarization consists in removing edges which cross from the connectivity graph of the network. Fig. 2.12 shows why planarization is mandatory for GFG and GRPS to guarantee delivery. Fig. 2.12 (a) shows a non-planar graph, where edge $EF$ crosses two other edges. Let us consider that a message is sent from $S$ to $D$ in face mode. From $S$ the message reaches $F$. The right hand rule goes as follows. $F$ draws a virtual line from itself to $S$ which it turns
CHAPTER 2. STATE OF THE ART

Figure 2.11: The route found by GFG/GPSR between nodes $S$ and $D$. Greedy mode is represented by double green arrows; face mode by single red arrows. Note that greedy mode is resumed at node $B$ because it is closer to $D$ than $A$, the node which triggered the switch to face mode.

counter-clockwise$^5$. The first neighbor this line hits is the next hop. To ease readability, we represent the execution of this algorithm by arrows. One can see that the message follows the path $S \rightarrow F \rightarrow G \rightarrow H \rightarrow E \rightarrow F$, and infinitely loops between last four hops.

Fig. 2.12 (b) depicts the Gabriel graph planarization algorithm which will be described in Section 3.2.3, yielding the planar graph of Fig. 2.12 (c). Applying face mode on the planar version yields path $S \rightarrow F \rightarrow G \rightarrow H \rightarrow E \rightarrow H \rightarrow D$. [Frey and Stojmenovic, 2006] shows that GFG and GPSR guaranteed delivery as long as the underlying graph is planar. We verify this result by simulation in Fig. 2.10.

Both the Greedy and the Face routing modes have been optimized to reduce energy consumption. The main idea is for a node to select its neighbor which minimizes cost over progress. In the purely geographical spirit, progress is defined as the reduction in Euclidean distance to destination when hopping to the next node; cost is defined as the energy spent for that hop, following any suitable energy model. To the best of our knowledge, $\text{EiE}$ (End-to-End routing process) [Elhafsi et al., 2008] is the first proposal to optimize both modes at the same time.

Some applications require the nodes to know their locations. Geographic communication protocols take advantage of this knowledge to perform some tasks which would otherwise be more expensive, such as routing. Nevertheless, having a node know its position is expensive. A first solution is to equip each node with a positioning device (e.g. GPS), but GPS-like systems are reportedly "cost and energy prohibitive for many applications, not sufficiently robust to jamming for military applications, and limited to outdoor applications" [Patwari et al., 2005]. Another solution is to program each node’s position manually during deployment. This, however, removes the possibility of randomly deploying a large number of nodes.

$^5$Again, it does not matter whether this line is turned clockwise or counter-clockwise as long as all the nodes follow the same rule.
CHAPTER 2. STATE OF THE ART

Figure 2.12: An example of a topology which shows why a graph needs to be planar for the face mode of GFG/GPSR to function. Circular arrows in (a) and (c) represent the execution of the right hand rule. A detailed description is given in Section 2.2.2.

2.2.3 Real, Approximate and Relative Coordinates

If even possible, having each node know its location comes at a price. The cost of location-awareness can be monetary (e.g. the cost of a GPS chip), energy-related (e.g. to power a GPS chip), related to man-power (e.g. manually programming a node’s position during deployment) or any combination thereof. While not completely solving the problem, reducing the portion of location-aware nodes in a network is a step forward. We call the subset of nodes which are location aware anchor nodes.

Regardless of the technique used, each anchor node is assumed to know its position (e.g. a set of \([x, y]\) coordinates in a two-dimensional deployment). Non-anchor nodes need to infer their own coordinates from the anchors using local measurements and localization protocols. When using anchor nodes, there is a clear distinction between localization (i.e. determining the physical positions in space/plane of the nodes) and routing. The nodes in the network typically determine their coordinates first; the geographic routing protocol then uses this information to send a message from any node to the sink.

There are two cases. In the first one, anchor nodes are location aware, meaning that they know their real coordinates (e.g. by means of GPS). As a result, non-anchor nodes determine approximate coordinates, as close as possible to their real ones. In the second case, anchor nodes do not know their real coordinates. Nodes thus have relative coordinates – a concept defined later – not related to their real coordinates.

Location-aware anchors. With anchor nodes knowing their real position, the goal of a node is to determine coordinates which are as close as possible to its real coordinates. We call these coordinates "approximate coordinates". Trilateration may be used: if each node knows its distance to a set of anchor nodes, it determines its position as the intersection of the circles centered at each anchor node and with radius the distance to this anchor node.

Whereas it is essentially the same idea as the one used by the GPS system, the main difficulty is to determine distances. As WSNs are multi-hop, a first approximation to the distance to an anchor node is the sum of distances of the individual links constituting the multi-hop shortest path. There are a number of techniques to measure these one-hop distances, including received signal strength (RSS) and time of arrival (TOA) measurement [Patwari et al., 2005]. Niculescu and Nath show that angle-of-arrival (AOA) is another valid technique for positioning in a wireless multi-hop network [Niculescu and Nath, 2003] but would require an antenna array, which is not practical in WSNs. [Destino et al., 2008] report experimental results on Time-of-Flight (ToF) ranging in which dedicated hardware is used to measure the time it takes for an ultra-wide band (UWB) signal to
Figure 2.13: By measuring its distance to \( A, B \text{ and } C \) (which know their location), location un-aware node \( X \) can infer its own location. This technique, known as (tri)lateration is applied similarly to GPS (a) and multi-hop wireless networks (b).

travel between two nodes. The authors, however, use a centralized localization algorithms.

In a GPS-like system (Fig. 2.13 (a)), localization precision depends on the number of anchors (i.e. satellites), their relative positions and the precision of distance measurements. Things are more complicated when applying trilateration to WSNs. First, distance measurement errors add up on a multi-hop link. Moreover, localization precision depends also on the alignment of nodes on this multi-hop link. As shown in Fig. 2.13 (b), \(|AX| \neq |AD| + |DX|\) because nodes \( A, D \text{ and } X \) are not aligned. This localization technique is used by the GPS-Free-Free [Benbadis et al., 2005] protocol. Localization accuracies of about 40m are reported on networks where each node has on average 10 neighbors (results are worse with sparser networks).

[Benbadis et al., 2007] extends these results with simulations showing that the success ratio of greedy routing when using approximated coordinates is lower than when using real coordinates.

The most critical drawback of using real or approximate coordinates for routing is that geographic proximity is not synonymous with electromagnetic proximity. In other words: geographically close nodes can not always communicate, and nodes which can communicate are not always geographically close. This observation by itself annihilates all geographic routing protocol solutions, and has been largely overseen. Most of the proposed protocols are evaluated by simulation. For most of them, the simulated propagation model is over-simplified, which precisely sustains the confusion.

Routing protocols perform well under these assumptions; yet, when confronted with a real propagation model, they fail dramatically. This is shown in [Kim et al., 2005] for the GFG and GPSR routing protocols. The same observation applies to all routing protocols based only on real or approximate coordinates. This will be covered in detail in Chapter 5.

In some applications, a node needs to know its position in order to report to the sink node where the sensed event is located. Nevertheless, the idea of using this geographical position alone for routing purposes does not hold in the general case because of the over-simplified assumptions on the propagation model it conveys. Real coordinates (determined by GPS-like hardware, manually programmed or determined relatively to anchor nodes) can not be used directly for routing purposes. A new localization system is needed in this case, which is related to the topology of the network.
CHAPTER 2. STATE OF THE ART

Figure 2.14: An example topology where each node is assigned relative coordinates. Each small white circle represents a node; edges interconnect nodes capable of communicating. A small white square represents an anchor node. A relative coordinate is a vector of number of hops to anchor nodes $A$, $B$, and $C$, respectively.

**Location-unaware anchors.** We call these coordinates "relative coordinates". These can be determined by using a set of location-unaware anchor nodes.

Relative coordinates of node $V$ are defined as a vector $\{V_1, V_2, \ldots, V_N\}$ where $V_i$ is the hop distance from the current node to anchor node $i$, and $N$ the number of anchor nodes. A simple way of assigning these relative coordinates is to ask each anchor node to periodically broadcast a message containing a counter incremented at each hop as it propagates through the network. Note that nodes can learn how many anchor nodes there are by listening to these broadcasted messages. Relative coordinates are not related to real coordinates. An example topology where each node is assigned relative coordinates is presented in Fig. 2.14.

Geographic routing protocols require a notion of distance to function. As we discuss later, note that the resulting relative distance is not directly related to physical distance. [Cao and Abdelzaher, 2004] proposes to infer distance from hop count to the anchor nodes using Euclidian distance. In their proposal, relative distance $||D||$ between nodes $V = V_1, V_2, \ldots, V_N$ and $W = W_1, W_2, \ldots, W_N$ is calculated as

$$||D|| = \sqrt{\sum_{i=1}^{N} (V_i - W_i)^2}$$

Several aspects of the relative coordinates need to be clarified. First, several distinct nodes may end up having the same coordinates. We refer to nodes with the same coordinates as "zones". Furthermore, because coordinates are not orthogonal (i.e. having more than three anchor nodes in a plane or four in a 3D space introduces redundancy), $||D||$ is not directly related to physical distance.

Despite these peculiarities, using relative coordinates is a promising approach to routing in WSNs. Simulation results in [Cao and Abdelzaher, 2004] show that with the relative coordinate space, less voids are encountered. This means that the success ratio of greedy geographic routing when using relative coordinates is higher than when using real coordinates, and hence more energy in the network is conserved. These results are confirmed experimentally by [Fonseca et al., 2005]. This work serves as a proof-of-concept experiment for relative coordinate routing in WSNs.
The difficulty when using anchor nodes is to select those anchor nodes. The Virtual Coordinate\(^6\) Assignment Protocol (VCap [Caruso et al., 2006]) elects anchor nodes dynamically during an initialization phase. A distributed protocol is designed to elect a predefined number of anchor nodes, evenly distributed around the edge of the network. This obviates the need for manual selection.

VCost [Elhafsi et al., 2007] is an extension of VCap. The authors keep the same landmark election scheme and relative coordinate assignment process, yet replace greedy routing by cost over progress routing. In this approach, a node elects as next hop its neighbor which minimizes the ratio between cost (several energy models are presented) over progress (decrease in relative distance to sink). Although the energy consumption of finding a multi-hop path is reduced, delivery ratio is low.

As said above, "zones" refer to a group of nodes which have the same relative coordinates. As the routing protocol bases its decision on these coordinates, ties may appear inside a zone, and the protocol may make the wrong decision. This can cause the multi-hop transmission to fail. [Liu and Abu-Ghazaleh, 2006] addresses this problem by turning each virtual coordinate into a floating point value, and slightly changing these coordinates as a function of the nodes neighborhood. The occurrence of ties and inconsistencies in the distances used for routing is hereby drastically reduced. To our knowledge, this is the first paper where a routing process using relative coordinates outperforms a routing process using real coordinates, in terms of hop count.

Real coordinates represent the nodes' geographical positions; relative coordinates represent the topological position the nodes, i.e. their position in the connectivity graph of the network. Routing over real coordinates suffers from void areas which makes greedy geographic routing fail. Some geographic routing protocols can deal with void areas, but they discover paths which are potentially very long. When using relative coordinates, void areas do not exist. As a result, routing paths can be shorter than when using real coordinates, provided the problem of "zones" is addressed.

So far, relative coordinates were obtained by counting the number of hops separating a node and the set of anchor nodes. The Greedy Embedding Spring Coordinate protocol (GSpring) [Leong et al., 2007] takes this concept one step further by introducing the spring model. Each link connecting two nodes is considered as a spring. These abstract springs have a rest length which is a function of the node’s neighborhood. If two nodes are closer to each other than this rest length (using the distance calculated as a function of the nodes’ relative coordinates), the repulsion force of the spring causes their relative coordinates to part away. Inversely, if the length of the abstract spring is larger than its rest length, an attraction force brings the nodes relatively closer.

During initialization of GSpring, an algorithm identifies a predefined number of anchor nodes at the edge of the network, and initializes their relative coordinates. The relative coordinates of these nodes do not change, and they appear as anchors to the spring system. An iterative process causes the abstract springs to be elongated and shortened until the spring system converges. Simulation results show that using this coordinate system yields better performance (in terms of number of hops) than using real coordinates.

Using relative coordinates for routing in WSNs is a very promising approach. Because the coordinate system is related to the topology of the network (and not to the physical location of the nodes), using routing protocols on top of relative coordinates yields better performances than using real coordinates. Moreover, relative coordinates avoid the cost of acquiring approximate coordinates.

Relative coordinates do require either a human operator to manually select the location of the anchor nodes, or a time-consuming and costly election protocol to perform the same task. Having

\(^6\)We argue that coordinates obtained using VCap are 'relative'; we use the term 'virtual coordinate' when node anchors are needed.
to elect a number of anchor nodes is relatively static. None of the cited work answers the questions related to network dynamics. During the lifetime of the network, nodes - including anchor nodes - disappear, and new nodes appear. Moreover, wireless links are dynamic. The usual answer to these problems is to periodically rebuild the relative coordinate system. This is not satisfactory as coordinates may continuously become outdated, and periodic rebuilding may be unnecessary when there is no traffic. To answer these problems, we propose to use virtual coordinates in Chapter 5.

2.3 Clustering and Virtual Backbones

2.3.1 Clustering Techniques

Clustering and virtual backbones are concepts inherited from research on wireless ad hoc networks [Tonguz and Ferrari, 2006]. Clustering techniques group nodes into clusters; a clusterhead ensures connectivity between the nodes in the cluster and other clusters [Younis et al., 2006, Amis et al., 2000]. Once the cluster is built, routing can be done hierarchically; intra-cluster and inter-cluster routing protocols are often very different [Theoleyre and Valois, 2007]. Ideas developed are inspired by wired networks where routers are grouped into Autonomous Systems and IP addresses are assigned hierarchically [Tanenbaum, 2002].

The Low-Energy Adaptive Clustering Hierarchy protocol (LEACH) [Heinzelman et al., 2002] is an early and very simple clustering protocol. Depending on a pre-defined probability, nodes elect themselves as clusterhead; other nodes join the closest clusterhead. The number of clusters grows linearly with the number of nodes, a behavior which may not be desirable. As clusterheads are placed randomly, some non-clusterhead nodes may end up with no clusterhead within communication range. This results in a disconnected graph, although the network is physically connected.

Implicit Earliest Deadline First (I-EDF) [Caccamo and Zhang, 2003] uses clustering to guarantee timeliness constraints. The authors assume that nodes know their physical location. A node implicitly knows to which cluster it belongs by mapping its position onto a pre-defined hexagonal grid. Clusterheads are assumed to be placed at the center of each hexagonal cell. Besides its limited practicality, this solution suffers from the unrealistic assumption that geographical distance is directly related to electromagnetic distance (i.e. Unit Disk Graph assumption).

[Krishnan and Starobinski, 2006] focuses on the process of nodes joining a clusterhead. The authors assume that clusterheads have been identified, and that only a limited number of nodes can be attached to each clusterhead. Expanding Ring Search, which can be considered the traditional approach, suffers from high overhead because it involves a sequence of expanding multi-hop broadcasts initiated by the clusterhead. At each broadcast, the newly contacted nodes declare themselves to the clusterhead; the process is stopped when the maximum number of cluster members is reached. [Krishnan and Starobinski, 2006] introduces the "growth budget" technique. This field is put into the multi-hop broadcast message initiated by the clusterhead, and decremented at each hop by the approximate number of newly contacted nodes at that hop. This technique enables cluster formation with as little as one tenth the signaling messages of Expanding Ring Search.

[Mitton and Fleury, 2005] defines the density of a node as the ratio between the number of links within the node’s 2-hop neighborhood and the number of 2-hop neighbors. Nodes which have the local highest density are elected clusterhead. The authors show that the obtained cluster-based structure can be used for example for efficient broadcasting [Mitton et al., 2006]. This assumes 2-hop neighborhood knowledge, which implies sending several Hello packets per node.

For correctness in this description we keep the term density as used in [Mitton and Fleury, 2005]. In the rest of this thesis, however, we keep the definition given in Section 3.2.3 where density is defined as the number of neighbors of a node.
The idea of attaching weights to nodes, and having the node with largest local weight be a clusterhead is very attractive. The Hybrid, Energy-Efficient, Distributed clustering protocol (HEED) [Younis and Fahmy, 2004] for example defines weight as a function of the node’s residual energy, while ACE (emergent Algorithm for highly uniform Cluster formation) [Chan and Perrig, 2004] uses the number of neighbors of a node.

2HopsCH [Aguilar and Afifi, 2008] assumes each node has knowledge of its 2-hop neighborhood. Similar to previously presented solutions, 2HopsCH assigns a metric to the nodes which is a compound of residual energy, node degree and density in a 2-hop environment.

In the AnyBody protocol [Watteyne et al., 2007a], nodes are grouped into clusters before clusters are interconnected through gateway nodes (which are not per se clusterhead nodes). Clustering is done using the density calculation proposed by Mitton et al. [Mitton and Fleury, 2005]; gradient routing is used to discover routes between clusterheads to the sink node. Nodes exchange Hello packets to maintain neighbor tables. These tables are used to calculate node densities (as defined in [Mitton and Fleury, 2005]); nodes with highest local density become clusterheads. The resulting structure is presented in Fig. 2.15 (a). After clusters are formed, a gradient is set up from the sink node to allow network-wide communication. A WSN can be represented logically as a set of interconnected clusterheads (Fig. 2.15 (b)).

Once a cluster is built, routing is done in a hierarchical way. A message goes from the source node to its cluster head, and is then relayed until it reaches the sink node. Some protocols assume that the clusterhead of the source node can directly reach the sink (e.g. LEACH), others assume cluster heads can communicate directly with neighbor clusterheads (e.g. I-EDF), a last group assumes gateway nodes are used to serve as a relay between clusterheads (e.g. AnyBody).

Although older proposals such as CBRP [Jiang et al., 1999] consider clustering and routing and different issues, more recent proposals merge both algorithms (i.e. I-EDF, AnyBody, Mitton’s proposal). [Théoleyre and Valois, 2005] takes the extreme approach of first identifying routing paths, before grouping nodes into clusters. This proposal elects dominating nodes which are connected by a virtual backbone (i.e. the network-wide routing paths); dominated nodes join the closest dominator to form clusters.

2.3.2 Quantifying Organization by Means of Entropy

With the aim to reduce disorder and improve efficiency, nodes in wireless multi-hop networks run a self-organization scheme to cooperatively organize the network. Although metrics such as complexity or self-stability are commonly used for evaluation, none of them quantifies the
efficiency to build and maintain an organization (order).

[Lu et al., 2008] applies the notion of entropy to wireless multi-hop networks to obtain a quantification of the internal organizational state generated by different self-organization schemes. The authors define the entropy of a link as \(-p(u, v) \log (p(u, v))\), where \(p(u, v)\) is the probability that a link between nodes \(u\) and \(v\) exists using a particular organization protocol. The entropy of a network is defined as the sum of the entropies of its links. This extended definition of entropy serves as a basis to compare several link-pruning strategies.

2.4 Bio-Inspired Self-Organization

Self-organization can be defined as "the emergence of system-wide functionality from simple local interactions between individual entities" [Prehofer and Bettstetter, 2005]. Self-organization principles can be applied to any collection of individual entities, be it a group of economic agents, individual bacteria, a school of fishes, or a wireless multi-hop network [Dressler, 2007].

The goal of self-organization in WSNs is to create a fully-autonomic network, which can be used without human intervention after deployment. From a networking point of view, it includes enabling network-wide communication from local simple interactions between nodes. This is, in fact, the definition of self-organization given above. [Mills, 2007] extends this definition by describing the design strategies of self-organizing systems. In the following paragraphs, we give examples of emergent behavior in economics and biological systems.

Emergent behavior principles apply to economics. Every economic agent uses only local information to decide how to behave. Buyers know only their own preferences and their own budget constraints, sellers know only their own costs. Their buying and selling on markets generate market prices, containing and transmitting all information about preferences, resources and production techniques. This way, market prices guide economic agents in making the best use of the resources available. Adam Smith called the market price "the invisible hand" which leads people to behave in the interest of society even when they seek only their self-interest [McMillan, 2002].

Emergent behavior also applies to much simpler systems such as a colony of Escherichia coli, a type of bacteria. Each bacterium is provided with flagella enabling it to move. In the presence of succinate (a chemical component), each bacterium excretes chemical substances which serve as attractants for other bacteria. Whereas these unicellular beings follow simple rules, these local interactions between individual entities yield chemotactic pattern formation: the bacteria organize into swarm rings and aggregates [Brenner et al., 1998].

In "migrating groups of fish, ungulates, insects and birds, [...] crowding limits the range over which individuals can detect one another" [Couzin et al., 2005]. Despite the local knowledge of each bird, a flock of birds moves in a coherent way (see Fig. 2.16). Moreover, as detailed in [Prehofer and Bettstetter, 2005], bird flocks exhibit all the advantageous properties of a self-organized system, namely scalability, adaptability (the flock changes when attacked by a bigger bird) and robustness (the flock is still coherent even when a bird gets killed).

The concepts of emergent behavior can be applied to large scale WSNs in a fashion similar to biological systems achieve [Prehofer and Bettstetter, 2005,Mills, 2007]. Because of the potentially very high number of nodes creating a wireless multi-hop network, the manufacturing cost of each individual node needs to be kept low. As a consequence, each node is capable of fulfilling only a limited set of tasks, and can only communicate with a limited number of close neighbor nodes. Emergent behavior enables extraordinary accomplishments by the network as a whole.

The ultimate goal of a self-organizing network is to be fully autonomic: to be deployed and used without any human intervention. The challenge of self-organizing a wireless multi-hop network is exemplified in Fig. 2.17. Each small white circle represents a node and edges interconnect nodes capable of communicating. Self-organization in such a network consists of enabling node
Figure 2.16: (a) Illustration of the main principles of a self-organizing system borrowed from [Prehofer and Bettstetter, 2005]. (b) picture of a starlings flock in Denmark (by Bjarne Winkler).

Figure 2.17: Depicting the problem of self-organizing a wireless multi-hop network.

To send a message to node X, by only having nodes communicate locally with their neighbor nodes (the ones within communication range). We call routing the process of finding a sequence of nodes to relay the message from C to X. This process needs to happen in an energy-efficient and robust manner. Energy-efficiency guarantees a long network lifetime; robustness implies that communication is still possible even under lossy links, or when nodes move and/or (dis)appear.

2.5 Open Issues

While the MAC layer has been addressed mostly from an energy-efficiency point of view, other important topics are still open issues. Industrial and other critical applications may be more interested in the timeliness constraints (i.e. how much time a message takes to reach the sink node) rather than energy constraints. As an example, a class 0 industrial network as described in [Pister et al., 2008] requires fixed and bounded delays to be able to react in time to possibly catastrophic events. It is safe to assume that the sensors in that network are powered by the mains.

Timeliness constraints are part of what is called a network’s quality-of-service (QoS). Another important QoS metric is reliability, i.e. what part of the messages sent in part network eventually
reach the destination. TSMP achieves 99.999% reliability. While this number is encouraging, TSMP is a fully scheduled and centralized MAC protocol, which is hence energy-consuming and bound to smaller networks. To our knowledge, no studies address the reliability vs. energy-consumption trade-off. An interesting question to answer is how reliable a network running an ultra-low power MAC protocol such as preamble sampling can achieve. Of course, QoS constraints such as timeliness and reliability are intertwined, and, most probably, there will be a trade-off between both.

Routing solutions are numerous. As the evaluation of their efficiency at large scale is complex by experimentation, routing schemes have been evaluated by simulation most of the time. For simulation results to be representative, many runs need to be done, which takes time and computation power. To reduce both, simple models have been used, especially for propagation. For routing schemes proposals which are based on simplistic models, simulation results maintain the illusion of their efficiency.

With hardware and software solutions becoming widespread and cheap, we believe important solution will be learnt when proposals are confronted to the real-world (early examples include the failure of GFG presented in [Kim et al., 2005]). Although this may be considered more engineering, we believe it would be very valuable to develop a set of tools to debug large-scale deployments.

Security is a much-overlooked topic which spans over all communication layer. Whereas security can be added on top of a working communication architecture in traditional networks (e.g. through Public Key Infrastructures – PKI), WSNs are much more constrained and we believe security should be integrated at protocol design stage.

### 2.6 Summary

WSNs draws on many applied research topics which spans from antenna and protocol design to market studies and social acceptability. It has received an increasing attention since 2000.

The MAC layer is responsible for controlling the state of the radio chip of a node (and its energy consumption) while arbitrating access to the medium. It can arguably be considered the cornerstone of the protocol stack of a wireless nodes. Proposed solutions can be classified in three categories, the selection among which mainly depending on how loaded the network is. Preamble-sampling solutions make more sense in low-load networks as it does not require any form of (periodic) synchronization between modes. With load increasing, nodes in the network need to be somehow synchronized to handle the many packets in a coordinated way. The loosest way of synchronizing is to have common active periods during which contention between nodes is resolved by variants of CSMA. When the traffic gets really important, contention-based medium access methods reach their limit, and scheduling takes over. Framed MAC protocols chop time into slots, and assign a slot to only one sender, receiver or sender-receiver pair, drastically reducing contention. Although such tight coordination comes at the cost of an important signaling traffic, it is the only to handle very high loads.

Self-organization techniques have long been synonymous to clustering, in which specialized algorithms group nodes and elect a leader node – called cluster-head – for each group. Besides their relatively hard implementability, this approach does not make much sense if data pipes between the nodes are equal. A paradigm shift is needed to make self-organization techniques more practical. Perhaps a good source of inspiration are biological swarms (e.g. fish, birds, ant) in which global movement of a swarm is an emergent behavior from local and simple interactions between individual entities.

Some applications require nodes to be location-aware. Geographic routing protocols reuse this information to find network-wide multi-hop paths. This technique is promising for WSN as it is
extremely scalable (each node has a very local view of the network) and stateless (no structure needs to be built and maintained). Although the simplest greedy approach fails in sparse network because of the presence of void areas, more advanced routing techniques such as GFG or GPSR guarantee delivery.

However, assuming all nodes know their position is in most cases energy and cost prohibitive. Moreover, it is more efficient for a node to know its topological rather than its geographical position. Relative coordinates are just that. After electing a subset of anchor nodes, nodes use lateration techniques to discover their relative coordinates. Although they are independent from a node’s real coordinates, routing over relative coordinates is possible.

If anchor nodes are placed correctly (ideally equally spaced on the edge of the network), relative coordinates are more accurate and less routing ties appear. Moreover, algorithm can be used to align the relative coordinates of the nodes and create a smoother gradient to the sink. If both these techniques are used, routing over relative coordinates yields higher delivery ratio and lower hop count than routing over real coordinates. The path discovered are, however, about a few percent longer than shortest paths.
3.1 Urban Wireless Sensor Networks

The Internet Engineering Task Force (IETF) is a major standardization body for computer networks. Its recently formed the "Routing Over Low power and Lossy networks" work group (ROLL) classifies WSN applications into Home and building automation, Industrial and Urban. The work presented in this document focuses on Urban WSNs (U-WSN).

In an U-WSN [Dohler et al., 2008c], nodes communicate in a wireless multi-hop manner and send their measurements to a subset of nodes called sink nodes; a sink node acting as a gateway to the outside world. Whilst millions of sensing nodes may very well be deployed in an urban area, they are likely to be associated to more than one U-WSN, where these networks may or may not communicate between one another. We consider a given U-WSN is composed of hundreds to thousands of nodes sending their measurements to a small number of sink nodes (typically 1-10). Deployment of nodes is likely to happen in batches, e.g. boxes of hundreds to thousands of nodes arrive and are deployed. The location of the nodes is random within given topological constraints, e.g. placement along a road, river, or at individual residences.

The nodes are highly resource constrained, i.e. cheap hardware, low memory and no infinite energy source. Although different node powering mechanisms are available, most node are battery-powered, hence network lifetime is strongly related to the nodes’ energy consumption. Battery shelf-life is usually in the order of 10-15 years, rendering any network lifetime maximization beyond this lifespan useless for battery powered nodes.

As the physical and electromagnetic distances between nodes can generally be large, i.e. from meters to several hundreds of meters; not every field node is likely to reach the access point in a single hop. Because of the wireless nature of communication and the size of the networks, the communication links between nodes are unreliable.

The majority of sensing nodes are configured to report their readings on a regular basis. The frequency of data sensing and reporting may be different but is generally expected to be fairly low, i.e. in the range of once per hour, per day, etc. Latency of an end-to-end delivery and acknowledgments of a successful data delivery may not be vital as sensing outages can be observed at the access point(s) - when, for instance, there is no reading arriving from a given sensor or cluster of sensors within a day. Rarely, the sensing nodes measure an event which classifies as alarm where such a classification is typically done locally within each node by means of a pre-programmed or prior diffused threshold. Unlike traditional ad hoc networks, the traffic pattern in U-WSNs is highly directional, the data messages flowing from the sensing nodes towards the sink node(s).

A communication architecture for U-WSNs needs to address the following challenges:

- **Energy efficiency.** Ideally, network lifetime should be limited by battery shelf-life, and should be in the order of 10-15 years.

- **Autonomous communication.** A non-specialist should be able to deploy the network, and the network should run without human intervention after deployment. The communication architecture should hence be robust against unreliable links and nodes (dis)appearing.
• **Scalability.** Communication should be possible in large (1,000+ nodes) or dense networks. Density refers to the number of neighbors of a node.

### 3.2 WSN System Model

As a practical means of evaluating the efficiency of a given solution, we use analysis and simulation. This, however, requires models to represent the behavior of a real – experimental – WSN.

#### 3.2.1 Centralized, Distributed, Localized

Communication algorithms in a multi-hop network can be centralized, distributed, localized. A centralized algorithm uses a specific node which gathers enough information to build a complete map of the network. Typically, it then arbitrates the communication by telling the nodes in the network when to communicate. An example of a centralized communication protocol is TSMP ([Networks, 2007], to be detailed) in which a single controller schedules the communication of all the nodes during network ramp-up, and then sends back this schedule to the network. Although centralized approaches can provide levels of quality of service other techniques cannot, it somewhat breaks the ad hoc nature of a multi-hop network.

Distributed is the opposite of centralized. In a distributed communication protocol, there is no network-wide leading entity. All nodes hence perform similarly; network-wide communication can be considered an emergent behavior. WSNs are envisioned to use distributed protocols because of their robustness against node failure and because they typically scale better. From an energy point of view, distributed protocols are more efficient as signaling messages do not need to cross the network to reach a central controller.

A localized algorithm is one where a node has a view of the network limited to a few hops away. Geographic routing (to be detailed) is localized as network-wide communication happens although nodes only know the position of their neighbors\(^1\). Link state routing, in which each node knows the entire topology, is not localized.

Note that distributed algorithms may be localized (e.g. geographic routing) or not (e.g. link state routing). A centralized algorithm is, however, never localized as the controller node has a network-wide view.

#### 3.2.2 Propagation Models

A useful signal can get distorted by noise, interference and the wireless channel. In addition to the free space propagation (the "deterministic", distance-dependent source of distortion), a signal can undergo reflection and refraction from surfaces, diffraction (from roof edges in an urban setting) or scattering (e.g. from trees).

Because of these effects, multiple copies of the same signal may reach the receiver, arriving via different paths. This multipath propagation can cause intra-symbol interference (overlap of symbol replicas within symbol, which can potentially lead to mutual cancellation) and inter-symbol interference (overlap of symbol replicas belonging to different symbols).

Because the number of nodes and their locations, the number of multi-path components and the number of reflections are random, the resulting signal strength and power received at a given node is random. Nevertheless, the propagation characteristics can be decomposed in pathloss, shadowing and fading. Fig. 3.1 shows how the receiver signal strength changes as the receiver node moves away from the transmitter, in an urban setting.

---

\(^1\)Strictly speaking, a node also needs to know the destination’s position.
CHAPTER 3. ASSUMPTIONS AND MODELS

Figure 3.1: Evolution of the received power as the receiver moves away from the transmitted, in an urban setting. The resulting signal can be decomposed in pathloss, shadowing and fading.

**Pathloss** is function of distance (as well as frequency, environment, antenna heights). It is obtained in Fig. 3.1 by averaging RSS over 1000 $\lambda$ ($\lambda$ is the wavelength of the signal).

Pathloss can be modeled easily. In the simplest case of free space, the Friis propagation model in (3.1) can be applied [Saunders, 2007]. Eq. (3.1) is used to calculate the RSS as a function of distance $d$ and frequency $f$, assuming a signal transmitted at 0dBm and isotropic antennas with unity gain. RSS decreases by 20dB per decade of distance.

$$\text{RSS}_{\text{dBm}} = 32.8 + 20 \log_{10}(f_{MHZ}) + 20 \log_{10}(d_{km})$$ (3.1)

The two-way ground reflection model considers all nodes are deployed at an altitude above an infinite plane. It captures the additive/destructive interaction between the direct line-of-sight wave path and a second path bouncing off the plane. Under the same assumption as for (3.1), and assuming the $d \gg h_T$ and $d \gg h_R$ ($h_T$ and $h_R$ being the elevation of the transmitter and receiver, respectively) (3.2) can be used to calculate the RSS. This model differs from the free space model in that (1) it is frequency independent because we assume $d \gg h$, and (2) it presents a far more rapid decrease in received power with range, 12dB for each doubling in distance in this case.

$$\text{RSS}_{\text{dBm}} = 40 \log_{10}(d) - 20 \log_{10}(h_T) - 20 \log_{10}(h_R)$$ (3.2)

Although path loss limits the desired signal power, it is beneficial for wireless communication as it also limits the area of interference of a node.

**Shadowing** is medium-scale fading, which can be obtained in Fig. 3.1 by averaging RSS over 40 $\lambda$. Shadowing is function of the environment (as well as frequency, distance, antenna heights) and random effect due to randomly appearing and disappearing waves. It traditionally obeys lognormal statistics, i.e. the logarithmic value of the random variable follows is Gaussian distributed\(^2\).

\(^2\)This is confirmed by the experimental results (see Fig 6.12).
Advantages of shadowing in wireless communication is that it limits interference and facilitates the capture effect. The capture effect, also called co-channel interference tolerance, is the ability of certain radios to correctly receive a strong signal from one transmitter, despite significant interference from other transmitters. This property violates the assumption that packet collision results in packet corruption, and can be exploited by many MAC and networking protocols to prevent packet collisions, increase network throughput and decrease delay [Whitehouse et al., 2005]. Disadvantages of shadowing is that it cause outages and is difficult to predict. Shadowing is the physical phenomenon which causes communication areas not to be perfect circles, rendering the Unit Disk Graph assumption described in Section 3.2.3 unrealistic.

Fading is obtained by subtracting the pathloss and shadowing from the initial signal (without averaging out). It is function of the environment and frequency, and random effect due to randomly wave additions/cancellations. Fading can can obey numerous statistics in dependency of the environment, ranging from Rayleigh (for non-Line-Of-Sight links - nLOS) to Ricean (Line-Of-Sight links – LOS).

Although fading potentially increases the capacity in MIMO channels, it causes errors and requires strong channel code.

Discussion. Although comprehensive measurement campaigns are still missing, the majority of WSN applications can assume slow fading, frequency-flat fading and spatially-selective channels [Dohler, 2008].

In simulation, a realistic propagation model should include pathloss, shadowing and fading. Although pathloss is easy to model, adding shadowing and fading is computationally intensive. As a result, the two-way ground reflection model is the default model implemented in most popular network simulators such as Network Simulators 2 and 3 [ns3, 2008] or the Georgia Tech Sensor Network Simulator (GTSNetS [Ould-Ahmed-Vall et al., 2005]) [Watteyne, 2006].

Modern radio chips can demodulate signals received at power levels as low as -95dBm (an example value obtained by experimentation on a TI CC1100 radio chip [CC1100, 2008]). When received with levels close to sensitivity, many packets can not be successfully demodulated. This causes the packet-error-rate (PER) – the ratio between unsuccessfully received and sent packets – to be close to unity. When packets are received at power levels well above sensitivity (15dBm above), PER is close to zero. In between lays a ”gray region”, where PER increases quasi linearly with received signal strength. These remarks are summarized in Fig. 3.2.

3.2.3 Graph Representation

We model a WSN as a collection of communicating nodes; each node can only communicate directly with a subset of the other nodes. We call this subset the node’s neighborhood. We describe different graphs in the following paragraphs; we detail a more realistic graph obtained through experimental observations in Chapter 6.

Unit Disk Graph. In the simple case, all nodes can be modeled as having a circular communication area of the same radius. Only nodes closer than this radius to each other can communicate. The resulting connectivity graph is called a Unit Disk Graph (UDG), an example of which is presented in Fig. 3.4 (a). Although arguably unrealistic, a UDG is easy to use in analysis and simulation and can easily be turned into a planar graph.

A planar graph is a graph where no edges cross; planarization is the action of removing some edges from a graph for it to become planar. Although all graphs can be planarized [Gabriel and Sokal, 1969], UDGs are particularly interesting because planarization algorithms are simple and distributed. A popular method is called Gabriel Graph transformation. In this method, an edge
is removed if there is a node inside the disk of diameter that edge. As an example, consider the 3-node network in Fig. 3.3. Edges $u - w$ and $v - w$ remain because there is no nodes inside the disks of diameter $u - w$ and $v - w$, respectively. Edge $u - v$ is removed because node $w$ is inside the disk of diameter $u - v$.

The strength of this method is that nodes agree on removing an edge without communicating. That is, assuming nodes know the positions of the other nodes, if node $u$ decides to remove edge $u - v$, so does node $v$. When applying this localized algorithm at all vertices in the graph, the graph becomes planar. An example of a planar UDG is given in Fig. 3.4 (b). An important result is that a Gabriel Graph preserves connectivity, that is, if the initial UDG is connected, so does its planar version [Gabriel and Sokal, 1969]. UDG and Gabriel Graphs are extensively used by geographic
routing protocols, which will be covered in Chapter 5.

**Unweighted Graph.** More realistic propagation models (Friis/two-way shadowing, Rayleigh/Rice) can be used to calculate RSS values as a function of distance. A first approximation is to consider that all packets can be successfully demodulated when received above sensitivity level. The resulting graph is similar to a UDG in that links are deterministic. Fig. 3.4 (c) represents such an unweighted graph. Note that node 18 cannot communicate with relatively close node 26 (on the right of the graph), although it can communicate with node 71 which is farther away. This is due to the random nature of this propagation model, and is the main difference with a UDG.

In reality, links come and go. A good approximation is to consider links as probabilistic. Based on gathered experimental data, we come up with a realistic probabilistic graph model of a WSN. This is described in Chapter 6.

**Arbitrary (random) graph.** When a graph does not meet the unit disk graph assumption or any other, it it said to be arbitrary. UDGs are therefore a subset of arbitrary graph. We use this definition in Chapter 5.
CHAPTER 3. ASSUMPTIONS AND MODELS

3.3 Analytical, Simulation and Experimental Apparatus

Chapters 4 and 5 constitute the technical contribution of this thesis by presenting a number of protocols and self-organization techniques. This section describes the tools used to evaluate their efficiency. We strongly believe that analysis, simulation and experimentation are complementary.

3.3.1 Analytical Model

We borrow the analytical settings from [Gupta and Kumar, 2000] where \( N \) nodes are homogeneously scattered in a disk of unit area (i.e. of radius \( \frac{1}{\sqrt{\pi}} \)). The collecting node, or sink, is positioned at the center of this disk. Each node has on average \( d \) neighbors, i.e. \( d \) nodes with which it can communicate in both directions.

During one second, a total of \( \alpha \) messages are generated in the network (i.e. this is the sum of all messages generated by all nodes). Sending one data message down one link requires \( D_{\text{data}} \) seconds. The source node of a message is randomly and uniformly chosen over the period of one second. Generation instants are randomly and uniformly distributed in time.

We assume a perfect routing protocol capable of setting up shortest route paths. We define shortest route as the sequence of hops from a given node to the sink with the smallest number of hops among all possible paths. In our analysis, we attach to each node its height \( h \), which is the minimum number of hops it takes for a message to be sent from that particular node to the sink. We assume this height is discovered by the routing protocol, and messages follow a sequence of nodes with decreasing heights to reach the sink. We call \( x \) the geographical distance – in meters – between a node and the sink.

**Relationship between height and distance to sink.** Let us first evaluate \( R \) the radio range of the nodes by (3.3). Knowing the number of nodes \( N \) deployed in a unit area, and neglecting border effects, the radio range \( R \), which leads to an average number of neighbors of \( d \), can be calculated using (3.3).

\[
R = \sqrt{\frac{d}{\pi (N - 1)}} \tag{3.3}
\]

The average distance between a node and its neighbor is
\[
\int_0^R x \frac{2\pi x}{\pi R} \, dx = \frac{2R^3}{3}. \]
As a result, the distance between a node of height \( h \) and a node of height \( h + 1 \) is, on average, \( \frac{2R^3}{3} \). As height increases when moving away from the sink on a straight line, the relationship between a node of height \( h \) and its distance to the sink \( x \) is given in (3.4).

\[
x = \frac{2R}{3} h \tag{3.4}
\]

We confirm these results by simulation. We randomly place 10,000 nodes in a unit area disk, and compute the average distance for each height. Presented results are averaged over 1,000 runs and presented in Fig. 3.5 together with a 95% confidence interval.

**Number of nodes of height \( h \).** Fig. 3.5 also shows that the number of nodes at a certain height increases linearly with that height. \( N(h) \), the number of nodes at height \( h \) is given by (3.5).

\[
\begin{cases}
N(h = 1) = d \\
N(h) = \pi NR^2 h, \forall \ h > 1
\end{cases} \tag{3.5}
\]
Figure 3.5: Distance to sink $x$ and number of nodes at the same distance $N(x)$ increase linearly with height. The straight lines are the theoretical values given by (3.4) and (3.5). These plots are drawn assuming density $d = 10$ and $D_{data} = 4ms$.

Using (3.4), we have $N(x) = \frac{3}{2} \pi R x \forall x \geq \frac{2R^3}{3}$.

**Impact of the convergecast communication scheme.** $\alpha$ messages are homogeneously generated per second over the whole network. We call $\rho(x)$ the number of messages crossing the co-centric circle of radius $x$ each second, see (3.6). $\rho(h)$ is the rate of messages that need to be sent by all the nodes at distance $h$ from the sink (obtained using the discrete version in (3.6)). The rate of messages to be sent by a single node at distance $h$ from the sink is simply $\frac{\rho(h)}{N(h)}$.

\[
\rho(x) = \int_{x}^{1} \alpha \cdot 2\pi \cdot t \, dt = \alpha(1 - \pi x^2)
\]

\[
\rho(h) = \begin{cases} 
\alpha, & \text{for } h = 1 \\
\alpha(1 - \pi \left(\frac{2R}{h}\right)^2), & \text{for } h > 1
\end{cases} \tag{3.6}
\]

We have $\rho(x) = 0$ for $x = \frac{1}{\sqrt{\pi}}$ (outer border of the deployment area) and $\rho(x) = \alpha$ for $x = 0$ (at the sink node). Eq. (3.6) shows that more messages need to be relayed when closer to the sink. This is because WSN are convergecast in nature. It is important to understand that any MAC protocol needs to deal with this constraint. $\rho(x)$ is depicted in Fig. 3.6.

**Number of messages sent in a 1-hop neighborhood.** We call $\bar{m}$ the number of messages sent per second in a 1-hop neighborhood. We call a 1-hop neighborhood a given node and its immediate $d$ neighbors. A 1-hop neighborhood is composed on average by $d+1$ nodes and occupies a circular area of radius $R$ around the node of interest. We assume all nodes in a 1-hop neighborhood need to relay the same amount of messages.

All $N(h)$ nodes of a given height need to transmit $\rho(h)$ messages per second. Each node thus transmits at rate $\frac{\rho(h)}{N(h)}$. As there are $d+1$ nodes in a neighborhood, $\frac{\rho(h)}{N(h)} (d+1)$ messages are transmitted per second in a 1-hop neighborhood. We translate this into the continuous domain (using $x$ instead of $h$) for easier manipulation, yielding (3.7).

\[
\bar{m}(x) = \frac{\rho(x)}{N(x)} (d+1) \tag{3.7}
\]

\[^{3}\text{Strictly speaking, the number of nodes at a given distance is always null, } N(x) \text{ is hence the density of nodes at distance } x.\]
Figure 3.6: $\rho(x)$, the number of messages crossing the co-centric circle of radius $x$ during one second. This graph was generated for $\alpha = 80$.

$\rho$, rate of relayed messages
$x$, distance from sink

Figure 3.7: M/M/1 buffer model

Theoretical bounds on $\alpha$. As shown by (3.6), contention is maximum close to the sink, i.e. between the sink and its immediate neighbors. The sink can only receive $\frac{1}{D_{data}}$ messages per second. Under the assumption of a constant generation of messages, $\alpha$ is bound by (3.8).

$$\alpha \leq \frac{1}{D_{data}} \quad (3.8)$$

Assuming $D_{data} = 8ms$, $\frac{1}{D_{data}} = 125$ messages can be absorbed by the sink, per second. With $N = 1000$, and assuming no data aggregation in the network, this means that each sensor node can generate one message every 8 seconds.

Buffer model. Messages travel over multiple hops. As the network can get busy at a given time or location, a message may have to wait in a node’s buffer before being sent to the next hop. The higher the value of $\alpha$, the fuller the node’s buffer gets. Buffers can be efficiently modeled using a buffer model derived from Markov Chains [Norris, 1998].

The M/M/1 buffer model used here is represented in Fig. 3.7. $\lambda$ and $\mu$ are message arrival and transmission rates. They are the mean number of received/transmitted messages per second, respectively. This buffer model is extremely helpful to derive both the average number of buffered messages and the delay experienced by messages sitting in a buffer. Note that we assume that all messages have the same priority, i.e. we assume a First-In-First-Out buffer.

Assuming that messages enter and leave the buffer following a Poisson distribution, an im-
important result on the M/M/1 buffer depicted in Fig. 3.7 is that the expected number of buffered messages in a M/M/1 buffer is \( \frac{1}{1-\frac{\lambda}{\mu}} \). The nodes close to the sink has the largest number of buffered messages, hence these buffers need to store \( \frac{\lambda}{\mu}(h=1) \) messages.

Moreover, Little’s law says that, for the M/M/1 buffer model, a message is buffered for \( \frac{1}{\mu} - \frac{1}{\lambda} = \frac{1}{\mu} \cdot \left( \frac{\lambda}{\mu} - 1 \right) \) seconds. This can be explained as follows. When a message gets into the buffer, it has to wait for the already buffered messages to be sent. It then takes \( \frac{1}{\mu} \) seconds to be sent itself.

**Clock drift.** Let \( \theta \) be the frequency tolerance of the time-base crystal. If the sensor node crystal has a real frequency of \((1 + \theta)f\) instead of \(f\), its clock will have an advance of \(\theta\) seconds after one second. Protocols such as SMAC only support de-synchronization up to \(D_{\max} = 500\mu s\) under practical assumptions (as reported in [Van Dam and Langendoen, 2003]). With a clock drift of \(\theta = 10ppm\), this means nodes need to resynchronize at least every \(2\cdot10^{-6} \cdot 25\) sec.

Note that the 2 in the denominator accounts for the fact that clocks may de-synchronize in opposite directions (one going at frequency \((1 + \theta)f\), the other at \((1 - \theta)f\)) [Sivrikaya and Yener, 2004]. \(T_{\text{sync}}\) is calculated using (3.9).

\[
T_{\text{sync}} = \frac{D_{\max}}{2\theta}.
\] (3.9)

**Energy model.** As current short-range radio chips consume about the same power when transmitting, receiving or idle listening, we choose to use radio-on time as a primary metric for energy consumption. Radio-on time is the number of seconds a radio is on. This model implicitly assumes the radio does not consume energy when it is off, which is a fair assumption as the current draw drops by about 3 orders of magnitude when the radio is switched off, as observed with contemporary hardware.

We put radio-on time into perspective by using radio duty cycle. A node’s radio duty-cycle \(\eta\) is the portion of time a radio is on \(\eta \in [0 \ldots 1]\). Duty cycle can be converted into node lifetime as follows. Let’s assume the node’s battery holds 17kJ (a typical values for two AAA batteries) under the hypothetical condition of a perfect voltage converter. A radio consumes about 60mW when it is on (to be shown in Table 2.1), lifetime can hence be estimated by (3.10).

\[
\text{lifetime (days)} = \frac{17 \cdot 10^3}{3600 \cdot 24 \cdot 0.060 \cdot \eta}
\] (3.10)

We depict the expected node lifetime as a function of \(\eta\) in Fig. 3.8.

### 3.3.2 Simulation Environment

The proposed communication protocols in this document have been evaluated by simulation. We chose to use one of two platforms (GTSNetS and a home made simulator), based on the modeling details required. Results are presented with a 95% confidence interval (average over many runs, typically \(10^5\)). The result actually is a triplet: the average value and the min/max value between which 95% of the results fall. In practice, we observe that, the larger the number of runs, the smaller the confidence interval, thus the more confident one can be about the average value.
CHAPTER 3. ASSUMPTIONS AND MODELS

Figure 3.8: Expected node lifetime as a function of $\eta$, assuming a battery containing 17kJ and node which consumes of 60mW when its radio is on.

GTSNetS. The Georgia Tech Sensor Network Simulator (GTSNetS) is a discrete-event simulator developed by Prof. George Riley [Ould-Ahmed-Vall et al., 2005] at the Georgia Institute of Technology, GA, USA. It was developed for WSNs, and thus contains suitable propagation models, battery and sensor models, as well as models for the environment. A graphical interface enables the user to interact with the simulation in real-time (this is unlike NS-2 or OPNET Modeler which offer only post-simulation replay). GTSNetS has inherited from the long experience of Prof. Riley on scalability, and distributed simulations of 50,000 nodes was reported [Zhang and Riley, 2005].

GTSNetS is entirely written in C++ (i.e. there is no second script language as in NS-2) considerably speeding up the simulation. It is open-source and comes with models for popular MAC (IEEE802.11 [Society, 2007]) and routing (DSR [Johnson et al., 2001], AODV [Perkins and Royer, 1999]) protocols. The active user community has developed more advanced models such as the support for IEEE802.15.4 [Cheng et al., 2006] or advanced mobility models [Konishi et al., 2005].

We have chosen to use GTSNetS over other simulators, such as NS-2 or OPNET Modeler, because it was built for WSNs. It contains most models used in WSNs and can simulate large networks [Watteyne, 2006]. We have used GTSNetS for simulations which required fine-grained results on, for example, energy consumption. When this level of detail is not required (typically for simulating emergent behavior in routing protocols), we have preferred to use a lighter homemade simulator.

Homemade simulator. This lightweight C++ simulator abstracts MAC layer details (i.e. it models neither collisions nor delays). It generates unit disk graphs (see Section 3.2.2) as well as the realistic propagation model presented in Chapter 6 and virtual graphs presented in Chapter 5. It was used to evaluate the efficiency of the geographic and virtual coordinates based routing protocols under different constraints such as variable number of nodes.
CHAPTER 3. ASSUMPTIONS AND MODELS

Graph generation. As communication architectures should be tested on topologies of different sizes and densities, the graph generation module is important. In particular, it is essential to generate random graphs in such a way that when density differs, other characteristics such as graph diameter/depth stay constant.

We use the graph generation method described in [Onat and Stojmenovic, 2007]. The algorithm generates random unit disk graph of \( n \) nodes with average density exactly equal to \( d \). The key is to determine the communication range as follows. We first position \( n \) nodes randomly in a square region (\( x \) and \( y \) coordinates are chosen uniformly). We then sort the \( \frac{(n-1)n}{2} \) inter-node distances by increasing order and set the communication range at the \( \frac{nd}{2} \) th smallest distance. This ensures that, on average, each node has exactly \( d \) neighbors. By default, we position the nodes a square (see examples in Fig. 3.9).

3.3.3 Experimental Apparatus

The following platforms were used:

- The EM2420 platform (Fig. 3.10 (left)) was used for the WiFly experiment. These boards are driven by an Atmel Atmega128L microcontroller and a CC2420 radio chip [CC2420, 2007] communicating on the 2.4 GHz ISM band. They are powered by two AAA batteries. This board is sold by Texas Instruments.

- The WSN430 platform (see Fig. 3.10 (right)) was used for the Sense& Sensitivity experiment. These boards are driven by an MSP430 microcontroller (10kB RAM/48kB ROM) and a CC1100 radio chip [CC1100, 2008] communicating at the 868 MHz ISM band. Two light sensors (one for visible light, one for infra-red) and a temperature sensor are present on the board. The board also features an external 1MB flash memory, a unique 48-bit serial number and three LEDs (Light Emitting Diodes). It is powered by a 3.8V 830 mA h Polyflex rechargeable battery cell. This board has been developed at the CITI laboratory.

The software implementation was done using the component based programming environment Think [Fassino et al., 2002]. Think (THInk Is Not a Kernel) is a C-implementation of the Fractal component model, licenced under GPL and LGPL.\(^4\) Similar to the TinyOS [Gay et al., 2007] or Contiki [Dunkels et al., 2007] operating systems, Think glues together components taken from a library, following an architecture language.

\(^4\)Think is available online at http://think.objectweb.org/.

Figure 3.9: Graphs obtained by the graph generation algorithm described in Section 3.3.2 which yield graphs with exact average densities.
CHAPTER 3. ASSUMPTIONS AND MODELS

Figure 3.10: (left) The EM2420 platform used for the WiFly experiment; (right) the WSN430 platform used for the Sense&Sensitivity experiment.

3.4 Summary

Within the large spectrum of WSN applications, we focus mainly of an urban-wide deployment involving large quantities of nodes and low throughput. Major challenges in those type of networks are scalability and energy-efficiency. Because in such large networks, topological changes can not avoided and may be frequent, communication solutions need to be self-organizing in nature.

From a communication point of view, a WSN can be described accurately as a communication graph in which nodes are the vertices and edges interconnect nodes which can communicate. Within all possibilities of arbitrary graphs, some families meet specific criteria such as the unit disk graphs which assume all nodes have a perfectly circular communication range. Graphs can also be built taking into account the way signals propagate from the sender nodes antenna.

We have presented the three tools which we have used extensively over this thesis work: we strongly believe that analysis, simulation and experiment are complementary.
CHAPTER 4

1-hopMAC, Pushing Neighborhood Discovery to MAC

In Section 2.1, we have presented an overview of the proposed MAC approaches. We have given an overview of framed MAC protocols, contention-based MAC protocols with common active periods and preamble-sampling MAC protocols. Results from Section 4.1 suggest that preamble-sampling MAC protocols are most appropriate for large-scale low-throughput and autonomous networks, such as Urban WSNs. This chapter presents the different versions of 1-hopMAC, their design and efficiency. This MAC protocol is used in the experiments presented in Chapter 6.

4.1 Analytical Comparison of MAC Approaches

Three families of MAC protocols have been described. In this section, we would like to evaluate which family to use depending on the application constraint (number of nodes, density, network load, expected average delay). We use the analytical framework introduced in Section 3.3.1 and apply it to a perfect MAC protocol, SMAC, basic preamble sampling, MFP and X-MAC. We believe these protocols best capture the essence of each MAC protocol family.

Without loss of generality, we assume for this description a $N = 1000$ node network with density $d = 10$ and 100 byte messages of length $D_{data} = 8ms$.

Baseline: a perfect MAC protocol. A perfect MAC protocol sends packets in such an order that there are no collisions. Nevertheless, we do not assume that at the sources a perfect MAC protocol schedules the generation time of the messages, hence messages may be buffered in relaying nodes.

We apply the buffer model and evaluate the arrival/transmission rates $\lambda$ and $\mu$. $\lambda$ is a function of $h$, and increases when $h$ decreases. $\lambda = \frac{\alpha}{\pi N R^2} \approx \frac{\alpha}{d}$ for $h = 1$ as neighbor nodes from the sink need to relay all the traffic generated by the network. For nodes at height 2 or more, when the current node wants to send a packet, it shares the channel with its $d$ neighbors (only $d - 1$ nodes compete at $h = 1$ because the sink nodes does send messages). If all of its neighbors have messages to send at the same time – assuming a fair MAC protocol – the current node can send $\frac{1}{D_{data}(d+1)}$ messages per second. At the edge, the network is less busy, and not all the neighbor nodes may want to send. The current node thus only competes with a subset of nodes wanting to transmit at that instant, which increases the transmission rate available to it. Nodes of a 1-hop neighborhood need to send $\bar{m}$ messages per second, yielding a channel occupancy of $\bar{m} D_{data}$. This means that at a given time, only $\bar{m} D_{data} d$ neighbor nodes have, on average, messages to send. The transmission rate available the current node thus increases to $\frac{1}{D_{data} (\bar{m} D_{data} d + 1)}$. These observation are summarized in (4.1), where $N(h)$ is the number of nodes at height $h$.

$$\begin{align*}
\lambda(h) &= \frac{\rho(h)}{N(h)} \\
\mu &= \frac{1}{D_{data} (\bar{m} D_{data} d + 1)}
\end{align*}$$ (4.1)

Typical sensors have in the order of 4kB of available RAM. Out of these 4kB, typical TinyOS applications require 2-3kB of RAM, so 1kB can be allocated to message buffering. With 100-byte long messages, a typical buffer can thus hold at most 10 messages. We depict the buffer size as a function of $\alpha$ in Fig. 4.1 and see that, even through the sink is capable of absorbing 125 messages per second, buffer size limits $\alpha$ to about a worst case of 120. To allow a margin between
the average and the worst case, we further limit $\alpha$ to 80. In a 1000-node network, this means each node can send a message every 12.5 seconds.

From the buffer model, we use Little’s law to derive end-to-end latency. In Fig. 4.2, the delay at a particular hop is the solid line plot. To derive the end-to-end latency, we sum the contribution of each hop, yielding the upper stair function. A message sent at the border of the network takes close to $350\, ms$ to reach the sink. We approximate the average delay by the delay taken by a message sent at $x = \frac{2}{3\sqrt{\pi}}$ (average distance to the sink) in Fig 4.3 (b).

In one second, the total radio-on budget is $N$ seconds (there are $N$ nodes), while the amount of used radio-time for sending the data is $2D_{data} \sum_h \rho(h)$ (the 2 stands for the fact that both the transmitter and the receiver need to be on when sending a message). The duty cycle is calculated in (4.2) and depicted in Fig. 4.3 (a).

$$\eta = \frac{2D_{data} \sum_h \rho(h)}{N} \quad (4.2)$$

**Framed MAC protocols.** The complexity of a framed MAC protocol lies within the creation of the scheduling table. The strength of this approach is that time slots can be scheduled to cope with per flow constraints. The major concern with this technique is that (1) node integration is a costly procedure, and should be used as seldom as possible, (2) multi-hop paths are over-provisioned to cope with the maximum traffic; this causes the overall throughput to go down when traffic is light.

Each time slot has a certain duration ($10\, ms$ for TSMP). Latency can be brought down to $10\, ms$ times the number of hops, as slots can be scheduled back-to-back. Using the default parameters, there are at most 15 hops, thus a maximum latency of $150\, ms$.

Beside the energy spent for transmitting/receiving data messages, a scheduled MAC protocol spends energy for (1) idle listening on receive slots which are not used, (2) sending keepalive reports, (3) signaling messages dealing with new node integration/removal. Duty cycle is thus largely a function of how dynamic the flows are, and is hard to extract for the general case.
Figure 4.2: **Perfect MAC.** A message experiences delay at each hop for $\alpha = 80$. From the continuous value of the delay experience at a specific $x$ (i.e. at a given hop), we construct the cumulative delay.

Figure 4.3: **Perfect MAC.** Both duty cycle and average delay increase when more messages are sent over the network.
Contention-based MAC protocols with common active periods. The buffer model developed for the perfect MAC case applies directly to SMAC, the only difference being that only one message can be sent every \( T_{\text{active}} \).

\[
\mu = \frac{1}{T_{\text{active}} \cdot (\bar{m}D_{\text{data}}d + 1)}
\]

While \( T_{\text{active}} \) must be large enough to allow for a data message to be sent (\( T_{\text{active}} \geq D_{\text{sync}} + D_{\text{rtscts}} + D_{\text{data}} \)), it should be small enough so that nodes can send enough messages to drain the network. This constraint is highest for sink neighbor nodes (\( h = 1 \)), hence \( \mu(h = 1) > \lambda(h = 1) \), and \( T_{\text{active}} < \frac{d}{\alpha(1+(d-1)D_{\text{data}})} \). We plot the range of possible values for \( T_{\text{active}} \) in Fig. 4.4.

The number of buffered messages as a function of \( \alpha \) with \( T_{\text{active}} = D_{\text{sync}} + D_{\text{rtscts}} + D_{\text{data}} \) is plotted in Fig. 4.5. Even in this best case (results are worse with larger values of \( T_{\text{active}} \)), \( \alpha \) is severely limited by the buffer space to around 30 messages per second. In a 1000-node network, this means a node should leave about 30 seconds between two successive transmissions. This low value is due to the fact that a message is preceded by an incompressible time of \( D_{\text{sync}} + D_{\text{rtscts}} \).

Every \( T_{\text{active}} \), all nodes wake up and have their radios on for at least \( D_{\text{sync}} + D_{\text{rtscts}} \). This accounts for a global on-time of \( N \cdot \frac{D_{\text{sync}} + D_{\text{rtscts}}}{T_{\text{active}}} \) per second. On top of this, \( \sum_h \rho(h) \) messages of length \( D_{\text{data}} \) are sent. Because both the sender and the receiver need to be on, this accounts for an extra \( 2D_{\text{data}} \cdot \sum_h \rho(h) \) radio-on time per second. The average duty cycle of our network is given in (4.3).

\[
\eta = \frac{N \cdot \frac{D_{\text{sync}} + D_{\text{rtscts}}}{T_{\text{active}}} + 2D_{\text{data}} \cdot \sum_h \rho h}{N}
\]

We extract the end to end latency from the buffer model as a function of \( \alpha \) and \( T_{\text{active}} \) (Fig. 4.6 (b)).

Preamble sampling protocols. In preamble-sampling, due to contention, the transmission burden per node is increased because relaying nodes need to resend collided messages. We call \( r(x) \)
Figure 4.5: SMAC. The number of buffered messages as function of $\alpha$ with $T_{active} = D_{sync} + D_{rtscts} + D_{data}$. Results are worse with larger values of $T_{active}$.

Figure 4.6: SMAC. There is a trade off between duty cycle and latency: the lower the duty cycle, the higher latency, and vice-versa. Although relatively large values of $\alpha$ can be coped with, both duty cycle and end-to-end latency can take large values. The areas where there is no plot are not possible (the network is saturated because $T_{active}$ is too large for that value of $\alpha$).
CHAPTER 4. 1-HOPMAC, PUSHING NEIGHBORHOOD DISCOVERY TO MAC

Figure 4.7: **Preamble MAC.** Number of retries as a function of distance to sink for $\alpha = 1$.

The average number of retries before a message is successfully sent, a function of $x$. We call $P(x)$ the collision probability of a given packet at the MAC layer. Both variables are given in (4.4), which assume a message of length $L$ including its preamble [Bertsekas and Gallager, 1987]. Eq. (4.4) is recursive in that $P(x)$ depends on $r(x)$ which itself depends on $P(x)$. This translates the fact that retransmitted messages cause more collisions, hence more (re)transmissions. As a result, if the network load is too high, the number of messages increases to infinity and the network collapses (a phenomenon known as the broadcast storm problem [Tseng et al., 2002]).

\[
\begin{align*}
P(x) & = 1 - e^{-2Lm(x)r(x)} \\
r(x) & = \frac{1}{1-P(x)}
\end{align*}
\] (4.4)

As recursive functions are hard to use, in practice we initialize $r(x)$ to 1 and iterate the process of $P(x)$ and $r(x)$ 5 times. By this iterative process, we implicitly assume that a node drops a packet after 5 unsuccessful transmission attempts. Fig. 4.7 depicts the number of retries as a function of distance to sink for $\alpha = 1$.

The number of messages which need to be relayed by a node at distance $x$ from the sink is given in (3.6). It decreases when the relaying node is further away from the sink. Because of collisions, each of these relayed messages may need to be retransmitted a number of times. The average number of retries for a message sent at distance $x$ from the sink is given in (4.4).

The total number of sent packets at a given $x$ is $\rho(x) \cdot r(x)$, i.e. $\rho(x)$ the number of generated message times $r(x)$ the number of retries for each message (Fig. 4.8). We call $M_{ps}$ the total number of sent messages over the network, which is the sum of messages sent for each hop. $M_{ps}$ is presented in (4.5) and Fig. 4.8. The step function in the upper part shows the contributions of nodes at different heights to the total number of sent messages.

\[
M_{ps} = \left\lfloor \frac{\sqrt{3}}{2R} \right\rfloor \sum_{i=1}^{\left\lfloor \frac{2\sqrt{3}}{3R} \right\rfloor} \rho \left( \frac{i}{3} \right) \cdot r \left( \frac{2R}{3} \right)
\] (4.5)
When a node located at distance $x$ from the sink issues a message, the number of transmissions needed by that single message to reach the sink (including all hops and retransmissions) is given in (4.6). An average node is at distance $\frac{2}{3\sqrt{\pi}}$ from the sink; we approximate the average delay by the delay experienced by a message sent from $x = \frac{2}{3\sqrt{\pi}}$.

\[
\sum_{i=1}^{\left\lfloor \frac{2R}{3} \right\rfloor} x \left( \frac{2R}{3} \right) \quad (4.6)
\]

We call $E_{\text{msg}}$ the amount of radio on-time for a 1-hop neighborhood (i.e. $d+1$ nodes) when a message is sent. We call $P_{\text{idle}} = \frac{D_{\text{on}}}{CI}$ the radio on-time per second when a node sits idle. During one second, $\mathcal{M}_{ps}$ messages are sent, costing a radio on-time of $\mathcal{M}_{ps} E_{\text{msg}}$. These transmissions last for $\mathcal{M}_{ps} (d+1) L$ (there are $d+1$ nodes in the 1-hop neighborhood of a sending node). The rest of the time (i.e. $N - \mathcal{M}_{ps} (d+1) L$ seconds), the network sits idle. This idle listening consumes $(N - \mathcal{M}_{ps} (d+1) L) P_{\text{idle}}$ radio on-time. We summarize these observations in (4.7).

\[
\eta = \frac{\mathcal{M}_{ps} E_{\text{msg}} + (N - \mathcal{M}_{ps} (d+1) L) P_{\text{idle}}}{N} \quad (4.7)
\]

We use the different $E_{\text{msg}}$ and $L$ value for basic preamble sampling, MFP and X-MAC listed in Table 4.1 to plot $\eta$ and $\Delta$ as a function of $\alpha$ and $CI$ in Fig. 4.9. It shows that (1) network load should be kept low (2) X-MAC outperforms MFP and basic preamble sampling. For a given network load $\alpha$, Fig. 4.9 (a) can be used to infer which $CI$ yields the lowest duty cycle and latency.

Discussion. Preamble-sampling is simple and robust, energy-efficient, synchronization-free and adding/removing nodes is transparent. Nevertheless, the use of long preamble limits the network load to a couple of messages per second. Contention based MAC protocol with common active periods exhibit a trade-off between latency and duty cycle. Buffers do not overflow until network load reaches tens of messages per second. Frame based scheduled MAC protocols is the only
### 4.1 Basic Formulas

<table>
<thead>
<tr>
<th>Protocol</th>
<th>$E_{\text{msg}}$</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>$CI + D_{\text{cca}} + \frac{CI + D_{\text{CCA}}}{2}d + (d + 1)D_{\text{data}}$</td>
<td>$CI + D_{\text{cca}} + D_{\text{data}}$</td>
</tr>
<tr>
<td>MFP</td>
<td>$(CI + D_{\text{cca}}) \left( \frac{D_{mf}}{T_{mf}} \right) + d(D_{mf} + T_{mf}) + 2D_{\text{data}}$</td>
<td>$CI + D_{\text{cca}} + D_{\text{data}}$</td>
</tr>
<tr>
<td>X-MAC</td>
<td>$(d - 1)(2D_{mf} + 2D_{\text{turn}} + D_{\text{ack}}) + \frac{(CI + D_{\text{CCA}})}{2} \left( \frac{D_{mf} + D_{\text{turn}}}{D_{mf} + D_{\text{turn}}} \right) + 2D_{\text{ack}} + 2D_{\text{data}}$</td>
<td>$\frac{CI + D_{\text{cca}}}{2} + D_{\text{data}}$</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison between different preamble-sampling based MAC protocols

(a) Duty cycle $\eta$ as a function of $\alpha$ and $CI$

(b) Average delay $\Delta$ as a function of $\alpha$ and $CI$. The surfaces for basic and MFP are undistinguishable.

Figure 4.9: **Preamble MAC.** X-MAC is better than MFP or basic preamble sampling in all cases. The area where there is no plot is not attainable (radio on-time is saturated).

family offering quality-of-service. Latency and throughput can be tuned on a per-flow basis. This flexibility comes at the price of tight synchronization and periodic network-wide communication, limiting network lifetime and scalability. Fig. 4.10 presents these remarks graphically.

### 4.2 Needs and Design-Drivers

A MAC protocol manages many things such as resource allocation (i.e. which part of the communication medium a user may use), division between up and downlink (i.e. whether the same frequency/time/code are used, or different ones) and coordination of access (i.e. how each user can use the wireless medium). Wireless links are unreliable, a node’s neighborhood hence varies continuously. As will be described in the results of the Sense& Sensitivity experiment (Section 6.3), the packet error rate of a link is directly related to its signal strength. Even when signal strength is stable, the associated PER may be high. As a result, a node may see another node as it neighbor only a fraction of the time. As this neighbor might be the only one available to route a message to the sink, it is essential to perform neighborhood discovery regularly.

Traditionally, neighborhood discovery is done using Hello messages. Each node periodically transmits a message containing its identifier and possibly other information such as routing information or battery status. Each node maintains a neighbor table, populating it as Hello messages arrive. Hello messages are no different from data messages for the MAC layer, and hence neighborhood discovery is, in this case, a routing layer issue.

This proactive approach has some serious drawbacks, mostly related to the periodic nature of Hello messages. How the period is chosen is non-trivial, and depends largely on the dynamics of...
Figure 4.10: Range of $\alpha$ over which a preamble-sampling, synchronized and scheduled MAC protocol should be used. The number below the axis show how much time a node should wait between successive packets. Although this specific result is obtained for $N = 1000$, $d = 10$ and $D_{\text{data}} = 8\text{ms}$, the qualitative result stays the same for other parameter sets.

the system. In a heavily loaded network with periodic traffic, Hello messages can be omitted when the information they convey is piggybacked on data messages; neighborhood discovery may thus work fine this way. Things get more complex in case of low and/or bursty traffic.

Let us take the example of a leakage detection WSN deployed in a large refinery. In this application, the network sits idle for months, but every now and then a group of sensors, simultaneously detecting a leak, generate bursts of messages. In this situation, proactively maintaining neighborhood tables makes little sense. Periodically exchanging Hello messages in an idle network is energy prohibitive, and increasing the Hello message period causes the neighbor tables to be out-of-date when they are needed.

Let us take another example. In this example, tens of thousands of battery-powered wireless sensors are attached to the water meters of homes in a small city. They report the homes’ water consumption once a day to a central server to increase billing accuracy and detect leaks. Because of the number of sensors, replacing batteries is clearly impractical. Although the flow of data is periodic, the period is so large that neighbors may have (dis)appeared between successive messages.

To solve these problems, the obvious solution is to discover neighbors reactively, i.e. right before the data message is sent. Once this list is built, the routing protocol of a node can browse the freshly constructed neighbor list to elect the next hop node. Because data messages are sent right after discovery, the neighbor list has a higher chance of being accurate\(^1\).

A first option is to consider neighbor discovery as a routing layer issue. Performing neighborhood discovery in a reactive way thus involves a three-way handshake at the routing layer: (1) a neighbor discovery routing message is sent, (2) neighbors contend to answer and (3) the data packet is sent to the most appropriate neighbor. Each of these steps generates independent MAC layer packets. The delay between neighbor discovery and data transmission together with the high contention during the second phase make this solution clearly inefficient.

We propose to perform reactive neighborhood discovery at the MAC layer. We use preamble-sampling as the underlying medium access scheme to guarantee energy-efficiency. We call the resulting protocol 1-hopMAC because it discovers a node’s 1-hop neighborhood on-demand. Depending on the application needs, one of two versions can be used. 1-hopMAC\(_{v1}\) discovers only the neighbor node which is closest to destination while 1-hopMAC\(_{v2}\) discovers all neighbors.

\(^1\)This statement is confirmed by the experimental data presented in Chapter 6.
CHAPTER 4. 1-HOPMAC, PUSHING NEIGHBORHOOD DISCOVERY TO MAC

4.3 Electing a Neighbor Based on a Metric

We start by giving the assumptions and models used before presenting the 1-hopMAC\textsubscript{v1a} protocol and the 1-hopMAC\textsubscript{v1b}, 1-hopMAC\textsubscript{v1c} and 1-hopMAC\textsubscript{v1d} variants. We consider the nodes’ radios cannot detect collisions, i.e. they are not able to distinguish collisions from reception errors due to external interference. We assume that each node knows the approximate network density \(d\), i.e. the average number of 1-hop neighbors\(^2\).

We assume each node contains a metric \(\beta\). This metric reflects the distance from a node to the sink (e.g. hop count, geographical distance or any number which contains a notion of distance to the sink). A node running 1-hopMAC\textsubscript{v1} sends the data message to the node’s neighbor which contains the smallest metric as next hop. By repeating this process over each hop, 1-hopMAC\textsubscript{v1} performs greedy routing over the metric space. As we will see in Chapter 5, the routing layer is responsible for choosing \(\beta\) suitably. We assume a \(\beta \in [\beta_{\text{min}} \ldots \beta_{\text{max}}]\).

We use the topology depicted in Fig. 4.11 when describing our protocol. Node \(S\) wants to send out a message to its neighbor with lowest metric (here node \(A\)). We consider energy consumption to be directly proportional to radio on-time.

4.3.1 Description of the Different Variants

Not using handshaking. As a starting point, let’s consider \(S\) sends out a request message. Upon hearing this message, nodes \(A, B\) and \(C\) each trigger a timer proportional to their metric and answer when their timer elapses. To have only one relaying node, all timers are aborted upon hearing the first reply. Nodes \(A, B\) and \(C\) may not hear one another. This could lead to three distinct relays. Successful timer abortion is only possible if all nodes are within each other’s transmission range. Fig. 4.12 shows that this assumption is hard to reach. 1-hopMAC\textsubscript{v1} therefore involves three-way handshaking.

1-hopMAC\textsubscript{v1a}. The three following steps are used in 1-hopMAC\textsubscript{v1a}:

1. \(S\) sends out a request message \((REQ)\);

2. \(A, B\) and \(C\) reply with as acknowledgment message \(ACK\) after \((\beta_A - \beta_{\text{min}}) \cdot \Delta t\), \((\beta_B - \beta_{\text{min}}) \cdot \Delta t\) and \((\beta_C - \beta_{\text{min}}) \cdot \Delta t\), respectively \((\Delta t\) is a small duration known by all nodes a priori). Each node performs a Clear-Channel-Assessment (CCA) before sending their ACK,

\(^2\)This is a loose assumption which is used for optimizations only.
CHAPTER 4. 1-HOPMAC, PUSHING NEIGHBORHOOD DISCOVERY TO MAC

Figure 4.12: Without handshaking, successful relaying node election is only possible if $A \cap B$ and $C$ are within each other's transmission range, i.e. in $A \cap B \cap C$. Without loss of generality, communication areas are presented here as disks.

![Diagram showing overlapping circles representing transmission ranges A, B, and C.](image-url)

Figure 4.13: 1-hopMAC\textsubscript{v1a}. The radio receiving/transmitting is represented by a rectangle under/above the line. No rectangle indicates that the radio is switched off.

\[
S \rightarrow \text{REQ} \rightarrow (\beta_{\text{max}} - \beta_{\text{min}} \cdot \Delta t + D_{\text{ACK}}) \rightarrow \text{DATA}
\]

\[
\begin{align*}
A & \rightarrow 1 \cdot \Delta t \rightarrow \text{ACK} \\
B & \rightarrow 3 \cdot \Delta t \rightarrow \text{ACK} \\
C & \rightarrow 2 \cdot \Delta t \rightarrow \text{ACK}
\end{align*}
\]

Figure 4.13: 1-hopMAC\textsubscript{v1a}. The radio receiving/transmitting is represented by a rectangle under/above the line. No rectangle indicates that the radio is switched off.

...to make sure the medium is free, thus avoiding collisions between $ACK$ for those nodes which can hear one another;

(3) \( S \) sends the $DATA$ message to its neighbors which answered first (here $A$). This $DATA$ message starts with the destination node’s identifier; the other nodes can thus go to sleep.

The resulting communication diagram is presented in Fig. 4.13. $S$ starts by sending out its request $REQ$, which is sliced into micro-frames. Each micro-frame contains a counter which indicates when the $REQ$ ends. All nodes are thus synchronized at the end of the $REQ$ message (the first vertical bar in Fig. 4.13). From this first synchronization point on, $S$ listens for possible $ACK$ messages. As $\beta < \beta_{\text{max}}$, the contention window during which nodes can send $ACK$ messages is limited. After the contention window, $S$ sends out a small header containing the next hop node it has elected. All nodes listen to this message, and go to sleep, except the elected node – here $A$ – which receives the $DATA$ message.

1-hopMAC\textsubscript{v1b}: reduce $S$’s listening period. After hearing the first $ACK$ message, $S$ knows which node to elect as relaying node, so it can turn its radio off. This is the idea behind 1-hopMAC\textsubscript{v1b}, presented in Fig. 4.14.
1-hopMAC\textsubscript{v1b}: reduce S’s listening period. After A’s ACK, B and C’s ACKs are useless for S. To avoid that B and C send these messages, S sends out a “don’t answer anymore” message from the moment it has received its first ACK until the end of the contention window. This message can be detected during the CCA period. This variant, called 1-hopMAC\textsubscript{v1c}, is presented in Fig. 4.15. Note that 1-hopMAC\textsubscript{v1c} is beneficial only when the extra power consumed by S to send its "don’t answer anymore" message is smaller than the energy B and C would have used to send their ACK message. We study this threshold in the next section.

1-hopMAC\textsubscript{v1d}: direct answer. As depicted in Fig. 4.16, it is beneficial to send the DATA message during the "don’t answer anymore" message period. As with 1-hopMAC\textsubscript{v1c}, nodes B and C do not send useless ACKs. After having sent an ACK, A needs to listen to the medium. Indeed, S announces it has elected A right after receiving its ACK. A needs to stay in receive state to receive the message.

4.3.2 Dynamic Switching Between Variants

We have described four variants, we now study which one is more energy-efficient. We use radio on-time as an energy-efficiency metric. We assume that the average duration of receiving one micro frame is equal to $D_{cca}$.

Eqs. (4.8)–(4.11) show the radio on-time $TR$ of the different variants. The items between square brackets correspond to the energy consumed by the sender, by the receiver, and by other nodes within sender’s 1-hop neighborhood, respectively. $D_x$ refers to the duration of message $x$; $\beta_{first}$ to the $\beta$ value of the node which answers first.
By comparing the above equations, we see that:

- $TR_s^{v_1a} > TR_s^{v_1b}$, 1-hopMAC$_{v_1b}$ should thus always be used instead of 1-hopMAC$_{v_1a}$;

- by looking at $TR_s^{v_1c} - TR_s^{v_1b}$, we see that 1-hopMAC$_{v_1c}$ should be used instead of 1-hopMAC$_{v_1b}$ only if $\beta_{\text{first}} > \frac{\beta_{\text{max}} - \beta_{\text{min}} \cdot \Delta t + D_{\text{ACK}}}{\Delta t}$. We call the right hand part of this inequality $\beta_{\text{thresh}}$;

- $TR_s^{v_1c} \geq TR_s^{v_1d}$, 1-hopMAC$_{v_1d}$ should thus always be used instead of 1-hopMAC$_{v_1c}$.
Either 1-hopMAC\textsubscript{v1b} or 1-hopMAC\textsubscript{v1d} should be used, depending on the value of \( \beta \text{_{first}} \). Swapping between 1-hopMAC\textsubscript{v1b} and 1-hopMAC\textsubscript{v1d} can be done dynamically. All nodes calculate \( \beta \text{_{thresh}} \); any node with \( \beta \) higher than \( \beta \text{_{thresh}} \) uses 1-hopMAC\textsubscript{v1d}, others 1-hopMAC\textsubscript{v1b}. For a neighbor node, the difference is that it has to listen to the channel after having transmitted its ACK. \( S \) starts using 1-hopMAC\textsubscript{v1b}. If it receives the first ACK after \( \beta \text{_{thresh}} \cdot \Delta t \), it switches to 1-hopMAC\textsubscript{v1d}.

Using this approach, we obtain a multi-mode MAC protocol, where the decision on which mode to use is distributed and no signaling messages are necessary. By swapping modes, we achieve energy efficiency. We call this protocol 1-hopMAC\textsubscript{v1}.

### 4.4 Reducing Collision Probability in Backoff-Based Election Mechanisms

Encoding information in time difference is used by WSN protocols other than 1-hopMAC\textsubscript{v1}. In [Bachir and Barthel, 2005], Bachir et al. propose to delay relaying a message by a duration inversely proportional to the remaining energy in the node. When used with Directed Diffusion [Intanagonwiwat et al., 2000], this method elects the max-min path in terms of energy among several candidates. The self-organizing protocol proposed in [Ye et al., 2001b] performs a backoff-based gradient establishment, where nodes use backoff timers proportional to a given link cost. This minimizes the number of broadcasted messages during the gradient setup phase.

We focus on the subset of these protocols (like 1-hopMAC\textsubscript{v1}) were the node which answers first is elected. It is possible in 1-hopMAC\textsubscript{v1} that first and second answers are separated by such a small duration that the ACK messages collide. We consider that messages collide (thus get corrupted) as soon as they overlap in time. For the remainder of this section, by collision probability we mean the collision probability involving the first answer message. This study is different from the well-known ALOHA multi-access protocol in that it only considers collision including the first response message. In this section, we calculate the collision probability and propose a backoff function to alleviate this problem.

#### 4.4.1 Motivation

We consider a node which sends a request to which its neighbors answer. We want to identify the first neighbor which speaks. Collision involving messages other than the first one is not considered a problem. Note that the instant at which each node speaks within the contention window is dictated by the node’s metric. We call a mapping function a function which translates the metric into a backoff duration.

Each node \( i \) has a metric \( \beta_i \). \( \mathcal{F} \) is the mapping function such that \( x_i = \mathcal{F}(\beta_i) \), where \( x_i \) is the backoff timer before answering to a request. Our goal is to tune \( \mathcal{F} \) so as to reduce the collision probability between response messages.

A first solution would be to use a CSMA approach in which each node performs a Clear Channel Assessment (CCA) before sending its message. If during the CCA, a message is detected, the transmission of the message is postponed. Whereas this approach is used in wireless networks, it induces the hidden node problem [Chaudet et al., 2005]. We hence do not assume this approach is used in this analysis, although it can easily be added later. We propose to tune the mapping function to minimize the collision probability (nodes do not perform a CCA before sending).

We summarize the variables, including the ones not presented yet, in Table 4.2. We assume the metric is bounded, and we have a window of length \( D \) in which the starting instants of \( N \) messages of length \( d \) are randomly positioned (Fig. 4.17). We call \( x_1, x_2, \ldots, x_N \) the instants when messages of nodes 1, 2, \ldots, \( N \) start. So \( x_i \in [0 \ldots D], \forall i \in [1 \ldots N] \). We call \( x_{\text{first}} = \)
Figure 4.17: In this case we have \( N = 4 \) nodes, and \( x_3 = x_{\text{first}} \). Even though messages 2 and 4 collide, we are not concerned about it because it does not involve the first message, here message 3.

\[
\begin{array}{c}
\begin{array}{cccc}
\text{x}_3 & \text{x}_1 & \text{x}_2 & \text{x}_4 \\
\text{d} & & & \\
\end{array}
\end{array}
\]

\( \text{min}(x_1, x_2, \ldots, x_N) \), the instant the first message is sent. The probability we want to calculate is formally written out in (4.12). \( P \) is the collision probability between the first message and at least one of the \( N - 1 \) other messages. Calculations presented in Section 4.4.2 are preliminary and necessary for evaluating the impact of a tunable mapping function studied in Section 4.4.3.

\[
P( x_i < x_{\text{first}} + d, \forall i \in [1 \ldots N] | x_i \neq x_{\text{first}}, \\
x_{\text{first}} = \text{min}(x_1, x_2, \ldots, x_N))
\]

(4.12)

**4.4.2 Calculating \( P \)**

**Assuming uniformly distributed values of \( x_i \).** We start by expressing \( P \) as a function of \( x_{\text{first}} \), \( D \) and \( N \) (Eq. (4.13) and Fig. 4.18, plotted for \( N = 10 \)).

\[
P(x_{\text{first}}, D, N) = \begin{cases} 
1 - \left( \frac{D-x_{\text{first}}-d}{D-x_{\text{first}}} \right)^{N-1} & \forall x_{\text{first}} \in [0 \ldots D - d] \\
1 & \text{otherwise.}
\end{cases}
\]

(4.13)

**Proof.** Let’s call \( p_1^{x_{\text{first}}=0} \) the probability that an individual message collides, and \( q_1^{x_{\text{first}}=0} \) that it does not, assuming \( x_{\text{first}} = 0 \). We have:
Figure 4.18: Collision probability decreases as \( x_{\text{first}} \) decreases and \( D \) increases. Having a high value of \( D \) also implies large transmission delays. These simulation results were obtained by averaging out 105 runs.

\[
\begin{cases}
p_1' = 0 = \frac{d}{D} \\ q_1' = 0 = \frac{D - d}{D}
\end{cases}
\]

With \( N - 1 \) messages, we have:

\[
p_{N-1} = 1 - (q_1' = 0 \times \cdots \times q_1' = 0)^{N-1} = 1 - (\frac{D - d}{D})^{N-1}.
\]

We generalize to the case where \( x_{\text{first}} \) is not necessarily equal to 0. For a single message, we obtain:

\[
\begin{cases}
p_1 = \frac{d}{D - x_{\text{first}}} \\ q_1 = \frac{D - x_{\text{first}} - d}{D - x_{\text{first}}}
\end{cases}
\]

When generalizing to \( N - 1 \) messages, we obtain:

\[
p_{N-1} = 1 - \left( \frac{D - x_{\text{first}} - d}{D - x_{\text{first}}} \right)^{N-1}
\]

With \( p_{N-1} = 1 \forall x_{\text{first}} \in [D - d \ldots D] \), we obtain Eq. (4.13).

We assume that the values of \( x_i \) are uniformly distributed over \([0 \ldots D]\), we calculate \( \mathbb{D}_{x_{\text{first}}} \), the probability density function (PDF) of \( x_{\text{first}} \). We integrate \( \mathbb{D}_{x_{\text{first}}} \) into (4.13), and obtain \( P(D, N) \) a formulation of the collision probability which is not a function of \( x_{\text{first}} \). By applying ordered statistics, we obtain Eq. (4.14).

\[
\mathbb{D}_{x_{\text{first}}} = \frac{N}{D} \left( \frac{D - x_{\text{first}}}{D} \right)^{N-1}.
\]
Figure 4.19: This is a formulation of the collision probability, independent of $x_{\text{first}}$. We assume here that values of $x_{i}$ are uniformly distributed over $[0 \ldots D]$. These simulation results were obtained by averaging out $10^5$ runs.

When integrating $D_{x_{\text{first}}}$ into $\mathbb{P}(x_{\text{first}}, D, N)$, we obtain $\mathbb{P}(D, N)$ (Eq. (4.15) and Fig. 4.19).

\[
\mathbb{P}(D, N) = \int_{0}^{D-d} \left(1 - \left(\frac{D-x_{\text{first}}-d}{D-D_{x_{\text{first}}}}\right)^{N-1}\right) \times \\
\frac{N}{D} \left(\frac{D-x_{\text{first}}}{D-D_{x_{\text{first}}}}\right)^{N-1} \, dx_{\text{first}} \\
+ \int_{D-d}^{D} \frac{N}{D} \left(\frac{D-x_{\text{first}}}{D-D_{x_{\text{first}}}}\right)^{N-1} \, dx_{\text{first}} \\
= 1 - \left(\frac{D-D_{x_{\text{first}}}}{D}\right)^{N}.
\]  

(4.15)

**Assuming non-uniformly distributed values of $x_{i}$.** We have assumed so far that values of $x_{i}$ were uniformly distributed over $[0 \ldots D]$. At a given node, $x_{i}$ is obtained by applying the mapping function $\mathcal{F}$ to the node’s metric $\beta_{i}$ (i.e. $x_{i} = \mathcal{F}(\beta_{i})$). For the remainder of this paper, we consider the values of $\beta_{i}$ are uniformly distributed. By tuning the mapping function, we are able to tune the PDF of the values of $x_{i}$. In Section 4.4.3, we aim at finding a mapping function which minimizes $\mathbb{P}(D)$.

We call "probability chain" the sequence of calculations required to obtain $\mathbb{P}(D)$ given a mapping function $\mathcal{F}$. The chain is depicted in Table 4.3.

- **step 1.** We consider that $\beta_{i}$, the metric contained in a node, is uniformly distributed.

- **step 2.** The mapping function is applied to the metric in order to obtain a backoff duration (i.e. $x_{i} = \mathcal{F}(\beta_{i})$). In our approach, we have $\mathbb{D}_{x_{i}}$ the PDF of $x_{i}$, and we want to find the mapping function $\mathcal{F}$ such that, from a uniformly distributed $\beta_{i}$, we obtain $x_{i}$ with PDF $\mathbb{D}_{x_{i}}$. $\mathcal{F}$ can be calculated using (4.16).

\[
x_{i} = \mathcal{F}(\beta_{i}) \iff \int_{0}^{x_{i}} \mathbb{D}_{x_{i}}(x) \, dx = \beta_{i}
\]  

(4.16)
\begin{table}[h]
\centering
\begin{tabular}{|c|c|l|}
\hline
step & variable & PDF / function \\
\hline
1 & $\beta_i$ & uniform \\
2 & $x_i = \mathcal{F}(\beta_i)$ & $\mathcal{D}_{x_i} = \mathcal{D}_{x_{\text{first}}} - i$ \\
3 & $x_{\text{first}} = \min(x_i)$ & $\mathcal{D}_{x_{\text{first}}} = N \mathcal{D}_{x_i}(x_{\text{first}}) \left( \int_{x_{\text{first}}}^{D} \mathcal{D}_{x_i}(x) dx \right)^{N-1}$ \\
4 & $\mathbb{P}(D, N) = \int_{0}^{D-d} \left[ 1 - \left( \frac{D-x-d}{D-x} \right)^{N-1} \right] \mathcal{D}_{x_{\text{first}}}(x) dx + \int_{D-d}^{D} \mathcal{D}_{x_{\text{first}}}(x) dx$ & \\
\hline
\end{tabular}
\caption{The probability chain.}
\end{table}

\textit{Proof.} Using the Jacobian transformation of random variables, we have:

\[(x_i = \mathcal{F}(\beta_i)) \Rightarrow (\beta_i = \mathcal{F}^{-1}(x_i))\]

\[\mathcal{F}^{-1}(x) = \beta_i = \beta_{\text{max}} \int_{0}^{x} \mathcal{D}_{x_i}(\xi) d\xi\]

Without loss of generality, we assume $\beta_{\text{max}} = 1$, from which (4.16) follows. \(\square\)

- \textbf{step 3.} The backoff duration $x_i$ of node $i$ is calculated using $\mathcal{F}$ the mapping function found in step 2 (i.e. $x_i = \mathcal{F}(\beta_i)$). We guarantee hereby that if $\beta_i$ is uniformly distributed, $x_i$ is distributed according to the PDF specified in step 2.

- \textbf{step 4.} Knowing $\mathcal{D}_{x_i}$, we can calculate $\mathcal{D}_{x_{\text{first}}}$ using ordered statistics:

\[\mathcal{D}_{x_{\text{first}}} = N \mathcal{D}_{x_i}(x_{\text{first}}) \left( \int_{x_{\text{first}}}^{D} \mathcal{D}_{x_i}(x) dx \right)^{N-1} \quad (4.17)\]

- \textbf{step 5.} When having $\mathcal{D}_{x_{\text{first}}}$, we can calculate the resulting probability by using:

\[\mathbb{P}(D, N) = \int_{0}^{D-d} \left[ 1 - \left( \frac{D-x-d}{D-x} \right)^{N-1} \right] \mathcal{D}_{x_{\text{first}}}(x) dx + \int_{D-d}^{D} \mathcal{D}_{x_{\text{first}}}(x) dx \quad (4.18)\]

We use this probability chain in the next section to reduce the collision probability by tuning $\mathcal{F}$.

### 4.4.3 Improving $\mathbb{P}$ with non-Uniformly Distributed Values of $x_i$

\textbf{Discussing $\mathcal{D}_{x_{\text{first}}}$}. We want to explore different mapping functions and evaluate how they affect the collision probability. We want the PDF of $x$ to be increasing because of the following. At best one message is sent at the beginning of the contention window $D$ (i.e. $\mathcal{D}_{x_i}$ should be small for small values of $x_i$), and the others sent as late as possible to avoid collision (i.e. $\mathcal{D}_{x_i}$ should be high for high values of $x_i$).

Yet, having a highly increasing function may not be beneficial. Consider the case where $\mathcal{D}_{x_i}$ equals 0 $\forall x_i \in [0 \ldots D-d]$ and $1/d \forall x_i \in [D-d \ldots D]$. It is a valid PDF ($\int_{0}^{D} \mathcal{D}_{x_i}(x) dx = 1$) and is increasing, but $\mathbb{P} = 1$. We therefore study the family of increasing mapping functions such that $\mathcal{D}_{x_i} = c \cdot x^\alpha$, with varying values of $\alpha$.

We start by studying $\mathcal{D}_{x_i} = 1/D$, $\mathcal{D}_{x_i} = 2x/D^2$, $\mathcal{D}_{x_i} = 3x^2/D^3$ and $\mathcal{D}_{x_i} = 4x^3/D^4$. Using the probability chain of Table 4.3, we extract $\mathcal{D}_{x_{\text{first}}}$. We confirm our calculations by simulation (see Fig. 4.20).
CHAPTER 4. 1-HOPMAC, PUSHING NEIGHBORHOOD DISCOVERY TO MAC

Figure 4.20: As expected, $D_{x_{first}}$ is highly impacted by $D_{x_i}$. This graph presents the theoretical curves of $D_{x_{first}}$ for four different $D_{x_i}$. These simulation results (dots) were averaged over 10^5 runs.

Table 4.4: Applying the probability chain on example $D_{x_i}$ functions.
Figure 4.21: These simulation results show that $P$ is impacted by $D_{x_i}$. Furthermore, for $D_{x_i} = c \cdot x^\alpha$, the best $\alpha$ is neither the highest nor the lowest. Best $\alpha$ is close to 1. These simulation results were obtained by averaging out $10^5$ runs.

On the other hand, we use the probability chain to find the mapping functions corresponding to these $D_{x_i}$. Results are presented in Table 4.4.

In Fig. 4.21, obtained by simulation, we see the impact of the different $D_{x_i}$ on $P$. As expected, some $D_{x_i}$ functions yield better results than a uniform distribution. Moreover, when a $D_{x_i}$ performs better than another one at a given $D$, it does so for all $D$ (i.e. curves do not cross). This is due to the fact that changing $D$ only expands the contention window, but does not affect the overall behavior of a given $D_{x_i}$.

So far, the best results were obtained with $2x^2$. In the next paragraph, we narrow down our research of the $D_{x_i}$ which yields lowest collision probability.

Reducing $D_{x_i}$ by tuning $\alpha$. We have $D_{x_i}(x) = \frac{(\alpha+1)x^\alpha}{D^\alpha}$ $\forall \alpha$. We plot $P(D = 200)$ for a varying value of $\alpha$ in Fig. 4.22. For $\alpha = 1.3$, $P(D = 200) = 0.255$ is the lowest value we obtain. We do not claim optimality, but have shown this value of $\alpha$ yields lowest $D_{x_i}$.

Applying the corresponding mapping function to 1-hopMAC\textsubscript{v1}. The lowest value of $P$ is obtained for $\alpha = 1.3$. By using the probability chain, we calculate that the corresponding mapping function is $F(\beta_i) = D \cdot \beta_i^{\frac{1}{\alpha}} = x_i$.

This result is very powerful. By changing just the mapping function – which is entirely local to the node, i.e. there is no overhead – we reduce the collision probability by over 40%. This result can be applied directly to 1-hopMAC\textsubscript{v1}. Upon receiving a $REQ$ message, a node with metric $\beta$ waits for a duration equal to $(f_{\text{max}} - f_{\text{min}}) \cdot \delta t \cdot \beta_i^{\frac{1}{\alpha}}$.

The analysis presented in this section has been previously published in [Watteyne et al., 2007c]. This work has since then been extended for the slotted case in [Bettstetter et al., 2008].
CHAPTER 4. 1-HOPMAC, PUSHING NEIGHBORHOOD DISCOVERY TO MAC

Figure 4.22: We narrow down our search for the best value of $\alpha$ in $D_{x_s} = c \cdot x^\alpha$. We therefore plot $P(D = 200)$, and we find it to be minimal for $\alpha = 1.3$. These simulation results were averaged over $10^7$ runs.

4.5 Retrieving the Neighbor List

1-hopMAC is energy-efficient because (1) it is based on preamble-sampling which can achieve duty cycles below 1% and (2) by moving neighborhood discovery to the MAC layer, it avoids periodically exchanging Hello messages at the routing layer. 1-hopMAC$_{v1}$ sends the data message to the neighbor which has the smallest metric, typically the closest to the sink node. 1-hopMAC$_{v1}$ does not create a complete neighbor table. This creates a problem when higher layer protocols or applications require this information. An example is the virtual coordinate routing scheme detailed in Chapter 5, in which a node’s virtual coordinate is calculated using the virtual coordinates of all its neighbors.

In this section, we describe 1-hopMAC$_{v2}$ which retrieves the list of all the neighbors, not just the identifier of the neighbor with lowest metric as 1-hopMAC$_{v1}$ does. 1-hopMAC$_{v2}$ can be seen as a generalization of 1-hopMAC$_{v1}$ usable in a wider spectrum of applications.

4.5.1 Overview of 1-hopMAC$_{v2}$

The following steps are used in 1-hopMAC$_{v2}$:

1. $S$ sends out a request message (REQ);

2. $A$, $B$ and $C$ each reply with an acknowledgment message $ACK$ after a backoff time taken randomly into $CW$. Each node performs a Clear-Channel-Assessment (CCA) before sending its $ACK$. In case the medium is not free, the $ACK$ message is resent at the next contention window;

3. $S$ sends a new $CW$ message to indicate a new contention window is open;

4. nodes which had re-scheduled their $ACK$ message because the channel was not free in the
CHAPTER 4. 1-HOPMAC, PUSHING NEIGHBORHOOD DISCOVERY TO MAC

Figure 4.23: 1-hopMAC\textsubscript{v2} retrieves the list of all the neighbors, not just the identifier of the neighbor with lowest metric as 1-hopMAC\textsubscript{v1}. It does so by opening successive contention windows until all nodes have been able to send their ACK message.

previous contention window resend their ACK message after a random time. Again, if the channel is not free, the ACK message is re-scheduled for the next contention window;

(5a) if s does not receive any new ACK messages in the new contention window, it sends the DATA message;

(5b) if at least one new ACK message was received by S during the last contention windows, S jumps back to step (3).

The resulting communication diagram is presented in Fig. 4.23. S starts by sending out its request REQ, which is sliced into micro-frames. Each micro-frame contains a counter which indicates when the REQ ends. All nodes are thus synchronized at the end of the REQ message (the first vertical bar in Fig. 4.23). From this first synchronization point on, S listens for possible ACK messages for CW seconds. Nodes A, B and C each choose a random backoff time within CW and thereafter perform a CCA. Nodes A and C sense the medium is free, and send their ACK messages. Nodes B senses a busy channel, and postpones its ACK for the next contention window. As S has received ACK messages during this first contention window, it sends a newCW message indicating a new contention window is opened. Only node B sends during this new contention window. As S has received ACK messages during this new contention window, it sends a newCW message indicating a new contention window is opened. No nodes answer during this third contention window, and S sends the DATA message. All nodes listen to the beginning of the DATA message, but go to sleep if their identifier is not listed in the header’s destination field (here B and C). A finACK message sent by the receiver node after successfully receiving the DATA message tells S the transmission has been successful.

After the last (empty) contention window, S knows the list of its neighbors. It can then inform the routing layer, which returns the identifier of the next hop. This information is then put in the header of the DATA message right before it is sent.

4.5.2 Tuning the Contention Window Size

1-hopMAC\textsubscript{v2} opens up new contention windows as long as at least one node sends an ACK message in the previous window. Let us assume each contention window has the same size CW and that a node has on average d neighbors which answer with an ACK message of length \(D_{ACK}\).
Moreover, $D_{newCW} = D_{ACK}$. For simplicity, we also assume that nodes are fully meshed, i.e. each node can hear the $ACK$ messages of all other nodes.

In this section, we describe how to tune $CW$, knowing $d$ and $D_{ACK}$. The objective is to have the smallest contention time $CT$ between the start of the first contention window and the end of the last one. The smaller $CT$, the less energy is consumed. If $CW$ is too small, $ACK$ messages repeatedly collide, and a large number of contention windows are created, yielding a large $CT$. Similarly, $CW$ may also be too large. As a result the collision probability drops and $S$ listens to the medium for too long.

We call $\alpha_i$ the number of contending nodes in the $i$th contention window. We thus have $\alpha_1 = d$. During a contention window, a node may see the channel as busy, and hence re-schedules its $ACK$ message transmission for the next contention window. We have $\alpha_{i+1} \leq \alpha_i$. When $\alpha$ reaches 1 (or less), no additional re-scheduling is possible as the sending node always see the medium is free.

At $CW_i$, a node sees the medium free with probability $(\frac{CW - 2D_{ACK}}{CW})^{\alpha_i-1}$. On average, $\alpha_i \cdot \left(1 - \left(\frac{CW - 2D_{ACK}}{CW}\right)^{\alpha_i-1}\right)$ will be re-scheduled. The expected number of contending nodes in a given contention window is given by (4.19).

$$\begin{cases} \alpha_1 = d \\ \alpha_{i+1} = \alpha_i \cdot \left(1 - \left(\frac{CW - 2D_{ACK}}{CW}\right)^{\alpha_i-1}\right) \end{cases}$$

We call $j$ the smallest value such that $\alpha_j \leq 1$ ($j$ is the number of contention windows in which at least an $ACK$ message is sent). We have $CT = (j + 1) \cdot CW + j \cdot D_{ACK}$. We plot $CT$ as a function of $CW$ in Fig. 4.24 for $d = 5$ and $D_{ACK} = 2ms$. It predicts the total contention time $CT$ when a node has 5 neighbors, each wanting to reply with a $2ms$ $ACK$ message. If the contention window $CW$ is too small, $ACK$ messages continuously collide and many contention windows are opened (the $j$ value is depicted in Fig. 4.24), hence a large $CT$. Inversely, collisions are very unlikely with very large $CW$ values, and $CT$ grows with $CW$. The saw shape in between these two extremes is explained as follows. At $CW = 11ms$, $j = 4$ which yields $CT = 63$; at $CW = 13ms$, $j$ is still equal to 4, yielding $CT = 73ms$. With $CW = 14ms$, only 3 windows are needed, and $CT$ is $62ms$. $CT$ goes through a minimum for $CW = 14ms$. Table 4.5 completes these results by giving the optimal value of $CW$ for various values of $d$ (assuming again $D_{ACK} = 2ms$).

Table 4.5 reads as follows. Consider a case in which each node has 5 neighbors on average. In order to have a smallest total contention time $CT$, one has to tune the contention window length to $14ms$. This yields a total contention time of $CT = 62ms$, on average.

<table>
<thead>
<tr>
<th>$d$</th>
<th>best Contention Window length $CW$</th>
<th>expected total Contention Time $CT$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>9 ms</td>
<td>31ms</td>
</tr>
<tr>
<td>4</td>
<td>10ms</td>
<td>46ms</td>
</tr>
<tr>
<td>5</td>
<td>14ms</td>
<td>62ms</td>
</tr>
<tr>
<td>6</td>
<td>17ms</td>
<td>74ms</td>
</tr>
<tr>
<td>7</td>
<td>16ms</td>
<td>88ms</td>
</tr>
<tr>
<td>8</td>
<td>19ms</td>
<td>103ms</td>
</tr>
<tr>
<td>9</td>
<td>22ms</td>
<td>118ms</td>
</tr>
<tr>
<td>10</td>
<td>25ms</td>
<td>133ms</td>
</tr>
</tbody>
</table>

Table 4.5: Tuning 1-hopMAC$_{v2}$. Best $CW$ as a function $d$, assuming $D_{ACK} = 2ms$. 
CHAPTER 4. 1-HOPMAC, PUSHING NEIGHBORHOOD DISCOVERY TO MAC

Figure 4.24: $CT$, the total contention time, is minimized by tuning the contention window size $CW$. Drawn for $d = 5$ and $D_{ACK} = 2ms$.

4.6 Summary

The MAC protocol is arguably the cornerstone of a WSN communication architecture as it arbitrates the access to a common medium while controlling the node’s energy consumption. Several MAC protocol families exist, among which preamble sampling has proven to be extremely efficient for large-scale networks with low throughput, such as urban WSNs.

Routing protocols use a node’s list of neighbors to elect the next hop node. Such a list is typically built and maintained by periodically exchange Hello messages. These routing layer packets contain the sender’s identifier as well as information useful for routing. Such a pro-active approach seems only poorly energy-efficient in low data rate WSNs. A long period between successive Hello messages causes the routing table to go stale; and small periods are energy prohibitive.

We propose the 1-hopMAC protocol which, based on preamble sampling techniques for energy efficiency, discovers the node’s neighborhood entirely on-demand. When a message needs to be sent, this MAC protocol first sends a request to which all neighbors answer. The node’s routing protocol can – based on the neighbors’ responses – elect the next hop node. Two versions of 1-hopMAC exist: 1-hopMAC$_{\text{v1}}$ and 1-hopMAC$_{\text{v2}}$.

1-hopMAC$_{\text{v1}}$ assumes that the node closest to the destination (i.e. the next hop node when using a greedy routing approach) answers first. This means that subsequent answers are not useful. Based on the time the first node answers, 1-hopMAC$_{\text{v1}}$ dynamically switches between a mode where the sending node shuts its radio off after having received the first message (preserving the sending node’s energy), and a mode where the sending node occupies the channel to prevent its neighbor from answering (preserving the neighbor nodes’ energy). Switching is done in such a way that the total energy consumption is kept to a minimum.

1-hopMAC$_{\text{v2}}$ retrieves the list of all the neighbors, not just the identifier of the neighbor with lowest metric as 1-hopMAC$_{\text{v1}}$ does. 1-hopMAC$_{\text{v2}}$ can be seen as a generalization of 1-hopMAC$_{\text{v1}}$, usable in a wider spectrum of applications. It opens up successive contention windows until all the nodes have been able to respond to the initial request. After all neighbors have announced
themselves, 1-hopMAC\textsubscript{v2} passes this freshly created neighbor list to the routing layer which elects the next hop node. This information is written inside the \textit{DATA} message header right before this message is sent.
CHAPTER 5

Geographic Routing over Virtual Coordinates

This chapter focuses on routing in large scale WSNs with low throughput. Section 5.1 presents an analytical study on clustering which suggests that clustering techniques make little sense in a fully-homogeneous setting because they do not increase throughput while being complex to implement and energy-hungry in a dynamic network.

For these reasons, we choose to focus on geographic routing techniques, where routing protocols reuse location information from the application layer to find a path between source and sink nodes. This is particularly suited for WSNs as no structure – i.e. a hierarchical or more simple routing structure – needs to be created and maintained.

[Frey and Stojmenovic, 2006] shows that GFG/GPSR guarantee delivery. We show in Section 5.2 that this is not true in the general case, i.e. when nodes do not know their location with infinite accuracy and when communication areas are not perfect circles. Section 5.3 presents a proposal of a more practical geographic routing technique which does not rely on hard-to-meet assumptions while performing similarly to GFG/GPSR.

We introduce virtual coordinates in Section 5.4. Like relative coordinates, they are not related to the node’s real coordinates, and are only meant to be used for routing in WSNs. They offer solutions which perform significantly better than using real coordinates. Unlike relative coordinates, no anchor nodes are required for setting up virtual coordinates, and the solution elegantly copes with network dynamics.

Virtual coordinates come in two flavors: 2D virtual positions and 1D virtual heights, presented in Sections 5.5 and 5.6, respectively.

5.1 An Analytical Study on Clustering

By lack of a direct and generic metric to quantify the quality of a self-organized network, we propose to look at the indirect metric of normalized throughput [Dohler et al., 2007b, Dohler et al., 2008b]. We assume a 2-tier hierarchy with $N$ total nodes, $C$ clusters and $M = N/C$ nodes per cluster. This 2-tier hierarchy requires two communication phases. There are $T_1 = M$ time slots in the first phase to transmit $c \cdot N$ bits from cluster members to the clusterhead, where $c$ is a constant which, without loss of generality, is assumed to be one. In the second phase, these $N$ bits are transmitted from the clusterheads to the sink.

Let us assume the pipes between cluster-heads and sink are $\xi$ times stronger than the data pipes between nodes towards the clusterheads; alternatively, this can also be achieved if the clusterheads perform data aggregation, leading to $\xi$-times less data volume to be forwarded.

Throughput of one-hop clustered network. We first assume that all nodes can communicate with their clusterheads over a single hop (1-hop clusters), and also that all clusterheads can reach the sink in a single hop. We compare the throughput when using this clustered structure close to LEACH [Heinzelman et al., 2002] with direct communication.

In the first phase, there are $T_1 = M$ time slots to transmit $N$ bits to the clusterheads and, in the second phase, there are $T_2 = M \cdot C / \xi$ time slots to transmit these $N$ bits to the sink. The throughput is hence
Figure 5.1: Normalized network throughput for different data pipes assuming a one-hop clustered network.

\[
\Theta = \frac{N}{M + M \cdot C/\xi} = \frac{\xi \cdot C}{\xi + C},
\]

which is depicted in Fig. 5.1. It can clearly be seen that clustering improves performance if the inter-cluster data pipes are stronger than the intra-cluster pipes. Also, note that with the number of clusters \( C \to \infty \), the normalized throughput \( \Theta \to \xi \).

For the flat topology (where nodes are not differentiated), for the sake of a fair comparison with the clustered approach, we consider \( C \) nodes having a bandwidth \( \xi \) times stronger than other nodes; we call them super-nodes. These super-nodes would play the role of clusterheads in a clustered structure. In this case, only one phase is needed which is composed of \( T_1 = (N - C) + C/\xi \) time slots. The normalized throughput of the flat topology is hence

\[
\Theta' = \frac{N}{(N - C) + C/\xi} = \frac{\xi}{\xi + C/N \cdot (1 - \xi)}
\]

This allows us to establish the conditions under which clustering is worthy by obtaining the difference \( \Delta \Theta \) between (5.1) and (5.2). It can easily be shown that clustering improves performance if the number of nodes \( N \) and super-nodes \( C \) relate as follows:

\[
N > \frac{C^2 \cdot (1 - \xi)}{C \cdot (1 - \xi) + \xi}.
\]
CHAPTER 5. GEOGRAPHIC ROUTING OVER VIRTUAL COORDINATES

Figure 5.2: Number of nodes versus number of super-nodes, where configurations above the lines yield performance gains.

This is plotted in Fig. 5.2, which shows that for a large number of super-nodes \( C \) the condition of clustering being beneficial is simply \( C/N < 1 \) and for a small number of super-nodes the number of nodes \( N \) has to be large. Note that clustering is not beneficial if links are undifferentiated (i.e. \( \xi = 1 \)).

**Throughput of multi-hop clustered network.** So far, we have assumed that node–clusterhead and clusterhead–sink communications all consisted of a single hop. Under this assumption, and with a clustered network, we do have multi-hop transmission. Nevertheless, the number of hops is limited to 2. Leaving the reader with this analysis is not entirely satisfactory as real-world rollouts span large areas, and the number of hops needed to go from a given node to the sink may be larger than 2. The aim of these paragraphs is to study the case where clusterhead–sink links are multi-hop.

We base this analysis on the results of [Gupta and Kumar, 2000], which states that in a network of \( N \) nodes and in the absence of structure, the bandwidth available at each node scales to \( \Theta \left( \frac{W}{\sqrt{N \log N}} \right) \), with \( W \) being its physical bandwidth. For the sake of simplicity, and without loss of generality, we consider \( W = 1 \text{bit/s} \). Sending one bit over a link of \( \bar{h} \) hops with nodes of bandwidth \( \frac{1}{\sqrt{N \log N}} \) takes \( 1 \cdot \bar{h} \cdot \sqrt{N \log N} \).

To analyze multi-hop communication, we need to know what the average number of hops is. We therefore use the topology model presented in Section 3.3.1. The average distance from any point in the unit disk to the sink in the center is \( \bar{L} = \int_{0}^{1} r \cdot 2\pi \cdot r dr = \frac{2}{3\sqrt{\pi}} \). Nodes are uniformly scattered in this domain, and to conserve energy, they tend to keep their transmission power (thus communication range) as small as possible. Each node covers \( \frac{1}{\sqrt{\pi N}} m^2 \), which is equivalent to a disk of diameter \( \frac{1}{\sqrt{\pi N}} m \). Two neighbor nodes are at a distance of \( \bar{r} = \frac{2}{\sqrt{\pi N}} m \). The average number of hops \( \bar{h} = \frac{\bar{r}}{\bar{r}} \) between a node and the sink is given in Eq. (5.4).
CHAPTER 5. GEOGRAPHIC ROUTING OVER VIRTUAL COORDINATES

\[ \tilde{h} = \frac{\sqrt{N}}{3}. \] (5.4)

Let’s detail the reasoning for the case where the network is clustered. During the first phase, in each cluster, \( \frac{N}{C} \) nodes send 1 bit to their clusterhead. This takes \( T_1 = \frac{N}{C} \) s. In the second phase, each clusterhead sends the collected \( \frac{N}{C} \) bits to the sink over a path of \( \frac{\sqrt{C}}{3} \) hops. We consider that only clusterheads are used as relays, each of which has an available bandwidth of \( \xi \sqrt{\log C} \). As a consequence, this second phase takes \( T_2 = \frac{N}{3\xi} \sqrt{\log C} \) seconds. Following the same notation, we have:

\[ \Theta = \frac{3 \cdot C \cdot \xi}{3 \cdot \xi + C \cdot \sqrt{\log C}}. \] (5.5)

We follow the same reasoning for the flat topology, and obtain \( T_2 = \frac{N}{3} \sqrt{\log N} \). As \( T_1 = 0 \), we have:

\[ \Theta' = \frac{3}{\sqrt{\log N}}. \] (5.6)

This allows us to establish the conditions under which clustering is worthy by obtaining the difference \( \Delta \Theta \) between (5.5) and (5.6). It can easily be shown that clustering improves performance if the number of nodes \( N \) and super-nodes \( C \) relate as follows:

\[ N > \exp \left[ \left( \frac{3}{C} + \frac{\sqrt{\log C}}{\xi} \right)^2 \right]. \] (5.7)

Fig. 5.3 depicts a WSN using either a clustered or unclustered approach. Although (5.7) suggests that clustering can be beneficial even with undifferentiated links (\( \xi = 1 \)), it is important to understand that this result stands only when clusterheads are capable of directly connecting with other clusterheads. This may involve more powerful transmitting devices, which somewhat breaks the undifferentiated link assumption.

Discussion. The results presented in this section are indicative only, where different assumptions on energy consumption, choice of hierarchy, choice of data flows (e.g., directed towards WSN sink), etc., yield different absolute results. Nonetheless, the results expose tendencies which can be of use for the large-scale system designer. The results indicate that, if clustering is used, the data pipes between the cluster heads need either to be stronger or data aggregation needs to be performed at the cluster-heads.

These results also suggest that clustering techniques make little sense in a fully-homogeneous setting. Moreover, clustering techniques are fairly complex and energy-hungry, the cost of which may become prohibitive in low-throughput networks. For these reasons, we propose to study self-organization techniques which do not involve clustering, but which are rather bio-inspired, and focus on geographic routing techniques.

5.2 Practicality of Geographic Routing Techniques

Bose et al. propose Greedy-Face-Greedy (GFG) routing [Bose et al., 1999], a variant of geographic routing in which, upon arriving at a void area, the protocol switches from greedy mode to face mode. Face mode is used to circumnavigate the void. When the current node is closer to destination than the node initially starting the face mode, the protocol returns to greedy mode – the void is
Figure 5.3: Clustered vs. unclustered multi-hop paths. The nodes are uniformly scattered in the circular domain. Black dots represent a clusterhead, white dots cluster members; black square the sink. The plain/dotted lines represent a multihop path with/without clustering, respectively. Less hops are needed in the clustered case because clusterheads are further apart than nodes.
CHAPTER 5. GEOGRAPHIC ROUTING OVER VIRTUAL COORDINATES

considered circumnavigated. In face mode, a node only considers the edges between itself and its neighbors which are on the planar Gabriel Graphs [Gabriel and Sokal, 1969]. Among these neighbors, it chooses the next hop using the left hand rule. The left hand rule consists in "rolling" to the left along the edges. Note that GFG has been reinvented by Karp and Kung and called GPSR [Karp and Kung, 2000].

GFG/GPSR only consider the communication edges which are part of the planar graph, a concept detailed in Section 3.2.3. We assume here that Gabriel transformation is used to planarize a graph, although results hold for any distributed planarization technique such as Relative Neighborhood Graph. Each node executes Gabriel transformation locally and decides, for each neighbor node, whether the link to that neighbor should be kept or not. Links which are not kept are removed logically as they are not taken into account for routing. After executing Gabriel transformation at each node, the graph is planar, i.e. no edges cross. Examples of a Unit Disk Graph and the associated planar graph is presented in Fig. 3.4 on page 42.

5.2.1 Impact of Loose Positioning

For a Gabriel Graph to preserve connectivity, a unit disk graph is assumed. When nodes approximate their position to a position different from their real position, this assumption does not hold. Although the connectivity graph is connected, the resulting Gabriel Graph may be disconnected. This results in GFG not guaranteeing delivery.

Let's take the example of Fig. 5.4. (a) represents 4 nodes positioned at their real locations. Edges interconnect nodes that can communicate. The topology (b) results from all nodes applying Gabriel transformation: it is planar (i.e. no edges cross) and connected (i.e. all connected nodes in the real graph remain to in the planar version). Let’s now take the graph depicted in (c). This is the same graph, with the same nodes connected by the same links, only nodes approximate their location to a position slightly different from their real one. Nodes are hence positioned at their approximate location. Graph (d) is obtained when each node in Graph (c) applies Gabriel transformation.

In Fig. 5.4 (c), node B sees A inside the circle of diameter B − C (which it is not in reality, but appears so because of location inaccuracy). Node B hence removes C from its logical neighbors. When comparing Fig. 5.4 (b) with (d), we clearly see the impact of loose positioning on GFG and other related protocols: the planar graph gets disconnected, a message may not reach destination although there exists a path, and delivery is not guaranteed. Note that this result stems for nodes knowing an approximate of their coordinates; it still assumes the communication area of each node is a perfect circle.

To evaluate how delivery ratio drops as localization becomes less and less accurate, we run simulations on the home-made packet level simulator. 200 nodes are positioned randomly in a square area of side 1000 according to a Poisson process. Each node can perfectly communicate with nodes closer than 200 units to them, i.e. a UDG assumption. The nodes acquire a position which is located randomly inside a square box of side 'accuracy' around their real position. The larger the size of the box (the x-axis in Fig. 5.5), the more inaccurate their positioning, and the lower the delivery ratio of GFG/GPSR.

5.2.2 Impact of non-UDGs

A Unit Disk Graph is built over the unrealistic assumption of communication areas being perfect circles. We simulate the impact of non-UDG by adding walls to our simulated topology. Fig. 5.6 (a) depicts such a simulated topology, where 10 100-unit-long walls are randomly put inside a 1000x1000 area. Both the center of the wall and its orientation (horizontal or vertical) are randomly and uniformly chosen. The propagation model is tuned such that two nodes with
Figure 5.4: GFG can not guarantee delivery with imperfect positioning. Removing the edges not part of the Gabriel Graph results in a disconnected graph (d), because Gabriel Graph transformation is run using the positions approximated by nodes (c) while these are different from their real positions (a). When assuming perfect positioning, Gabriel Graph preserves connectivity (b).

Figure 5.5: Delivery ratio drops when positioning gets inaccurate for GFG/GPSR. Delivery ratio stays optimum for the 3rules routing protocol presented in Section 5.3. These simulation results were averaged over $10^5$ runs.
a wall in their line-of-sight cannot communicate, even when they are closer than their 200-unit communication range. These nodes have no perfectly circular communication range. With more walls added, the graph drifts away from a UDG.

Fig. 5.6 (b) details the problem faced by the distributed Gabriel Graph transformation when faced with non-UDGs. The four nodes $X$, $Y$, $V$, $W$ are represented by small black circles; nodes which are able to communicate are linked by either plain or dashed lines. The two dashed circles are represented for construction only and have no physical meaning. The two thick segments represent walls.

We now describe the execution of the distributed Gabriel Graph transformation. Node $V$ sees only node $W$ and sees no other node in the circle of diameter $V \rightarrow W$. It hence decides to keep link $V \rightarrow W$, i.e. it does not hide this link from its routing protocol. Node $W$, which has two neighbors $V$ and $X$, however, takes a different decision. As $X$ is inside the disk of diameter $V \rightarrow W$, node $W$ removes link $V \rightarrow W$, i.e. hiding it from its routing protocol. As a consequence, $V$ may decide to route to packet to $W$ while $W$ will never send a packet to $V$. The planarization phase has changed $V \rightarrow W$ into a directional link. The same applies to link $Y \rightarrow X$. When a graph does not meet the UDG assumption, some links may become unidirectional, causing GFG/GPSR to fail. Fig. 5.7 shows how delivery ratio drops as walls are added, i.e. as the UDG graph is broken.

### 5.2.3 Discussion

In [Kim et al., 2005], the authors report experimental results on GFG/GPSR. As depicted in Fig. 5.8, although the nodes informed of their exact location, the non-circular shapes of the node’s communication areas caused the Gabriel Graph planarization techniques to fail. Fig. 5.8 (a) shows the connectivity graph as an overlay of the indoor map the node were deployed in. It shows that all the nodes form a connected set. Fig. 5.8 (b) shows the logical connectivity graph after Gabriel transformation. It shows that crossing links remain, some links have become unidirectional, and links were removed which should not have, resulting in an unconnected, un-planar directed graph on which GFG/GPSR fail.
CHAPTER 5. GEOGRAPHIC ROUTING OVER VIRTUAL COORDINATES

Figure 5.7: Delivery ratio drops as the UDG assumption is broken by adding walls. Delivery ratio stays optimum for the 3rules routing protocol presented in Section 5.3. These simulation results were averaged over $10^5$ runs.

[Kim et al., 2005] proposes the Cross-Link Detection Protocol (CLDP) which removes crossing links and is able to planarize an arbitrary random graph. While CLDP is a step in the right direction, it uses a rather heavy method using probe messages, which travel across the network and detect crossing links. In practice, this method may end up consuming large quantities of energy and time, especially on deployments in which the connectivity graph is dynamic (e.g. high PER links, mobility).

Greedy geographic routing is simple to implement, yet it fails when a void is met, a situation which appears often in low-density deployments (<7 neighbors per node). Face routing protocols, such as GFG or GPSR, extend greedy approach by switching to a mode capable of circumnavigating the void when such a void is met. Nevertheless, the cornerstone of these proposals is the failure-prone planarization technique. Distributed – thus overhead-free – solutions rely heavily on the Unit Disk Graph assumption and fail dramatically when this assumption is broken. Variants such as CLDP function on arbitrarily random graph but loose the local nature, which is not desirable from an energy-efficiency point of view. New solutions are needed.

5.3 The 3-rule Routing Protocol

In this section, we present the 3rule routing protocol [Watteyne et al., 2007b]. The main idea is to use the sequence of already traversed nodes to help determining the next hop node. Based on the list of traversed nodes, a node can know whether it has already forwarded this packet, and if so, to which neighbor. By applying three simple rules, a graph can be searched methodically. In case of a tie, the neighbor geographically closest to the destination is chosen. We show that our routing technique guarantees delivery on arbitrary random graphs, while keeping hop count similar than the one obtained by GFG/GPSR.
5.3.1 Distributed Exhaustive Search in an Arbitrary Graph

We propose to use the sequence of already traversed nodes to help finding the next hop. This sequence is obtained by asking each traversed node to append its identifier to the packet’s header\(^1\). The alternative idea of recording the relayed packets inside the nodes does not seem suitable as we do not want nodes to keep long-term information.

Based on this list of traversed nodes, sensors can take actions to guarantee that the message does not use the same path twice in the same direction. This way, infinite loops are avoided, and no unrealistic assumptions are made on the accuracy of the position-awareness or the nature of the graph. When combined with geographical information (the resulting protocol called 3rule routing is presented in the next section) the number of hops is comparable to the one obtained when using GFG/GPSR.

The following three rules guarantee that a message visits all the nodes of a network within a bounded number of hops. They can be seen as a distributed version a depth first search. A node is requested to:

1. never send a packet to a neighbor to which it has already sent a packet;
2. send a packet \( \text{back} \) to one of its neighbors (\( i.e. \) this neighbor has sent a packet to the current node before) only if it has no other neighbors it has never communicated with;
3. in case there are several neighbors the current node could send the packet \( \text{back} \) to, it should pick the one which has sent the packet \( \text{last} \).

**Worst case number of hops.** The three rules guarantee that the whole graph is traversed within a finite number of hops (as there are no infinite loops). We extract the worst case number of hops.

**Statement 1.** In a network with \( N \) nodes, the worst case number of hops is \( N \cdot (N - 1) \).

**Proof.** In a network with \( N \) nodes, we have at most \( (N - 1) + (N - 2) + \cdots + 3 + 2 + 1 = \frac{N \cdot (N - 1)}{2} \) edges between nodes. Each of these edges can only be traversed twice according to our three rules, once in each direction. So the number of packets transmitted in the worst case is \( \frac{N \cdot (N - 1)}{2} \times 2 \). \( \Box \)

\(^1\)Even though the size of the header increases, this does not significantly affect the energy needed to transmit the message. This assumption is verified experimentally, as presented in Chapter 6.
CHAPTER 5. GEOGRAPHIC ROUTING OVER VIRTUAL COORDINATES

(a) GFG guarantees delivery for UDGs with perfect position-awareness. This is a planar graph.

(b) 3rule routing guarantees delivery for an arbitrary random graph. The graph does not need to be planarized.

Figure 5.9: Both GFG and 3rule routing guarantee delivery for UDGs with perfect position-awareness, and both switch between two modes (green double arrows represent the greedy model; red thin arrows represent the second mode). 3rule routing is the only of the two protocols which functions on an arbitrary random graph.

A less restrictive worst case number of hops can be given when the connectivity graph (i.e. the number of edges it is composed of) is known.

Statement 2. In a network with $M$ edges – connecting an arbitrary large number of nodes – the worst case number of hops is $2 \cdot M$.

Proof. Similarly to the previous proof, each link can only be traversed twice, once in each direction.

While the three rules guarantee the destination is eventually found (hop count is bounded), the average number of hops is high. We therefore introduce geographic information within the three rules to lower hop count. Simulation results presented in the next section show that the resulting routing protocol – called 3rule routing – performs as well as GFG/GPSR.

5.3.2 Introducing Geographic Information to Lower Hop Count

When greedy geographic routing achieves delivery, delivery is done with a hop count close to minimum. We therefore choose to use a greedy approach as long as it achieves progress. When a void is met, a second mode is needed to circumnavigate the void until the node reached is closer to destination than the node which triggered the face mode. This is the same dual-mode behavior as the one used in GFG or GPSR; 3rule routing has a different face mode which does not rely on the error-prone distributed planarization.

We choose to first describe the algorithm as it is implemented in the nodes, followed by a discussion. Fig. 5.9 depicts the paths discovered by GFG and the 3rule routing. Both routing protocols achieve delivery and simulation results presented below confirm that 3rule routing discovers paths with, on average, the same number of hops than the ones discovered by GFG/GPSR.

We now describe the execution of 3rule routing at a given node (a discussion follows). The algorithm manages three lists: neighbors, received_from, sent_to, received_from.
is an ordered last-in-first-out stack, the others and unordered lists, respectively. The MAC protocol is assumed to maintain an up-to-date list of the node’s neighbors, including the distance from each of these neighbors to the sink node.

- upon receiving a message, lists received_from and sent_to are emptied; neighbors is populated with the identifiers of the neighbors;

- the sequence of already traversed nodes is extracted from the header of the message, and is parsed in chronological order (the source node’s identifier read first). The protocol parses the sequence and looks for its own identifier. For each occurrence of its identifier, it adds the preceding identifier to received_from and the one after its own identifier to sent_to. Once the three lists are filled, the sequence is no longer needed.

- the node removes all identifiers from neighbors which appear in sent_to (rule 1) or received_from (part of rule 2). It discards the sent_to list which is not needed anymore.

- if neighbors is empty (part of rule 2), the node sends the message to the last node contained in received_from. This corresponds to rule (3).

- if neighbors is not empty (part of rule 2), the node orders the entries in neighbors by increasing distance to sink. It sends the message to the first node in the list (i.e. the node closest to the sink). This corresponds to rule (1).

We call this algorithm the 3rule routing protocol. It combines the three rules described above with geographic information to lower hop count. In case there are multiple neighbors which have never sent nor received the current message to/from the current node, the algorithm chooses the one which is closest to the sink. In fact, this is what happens most of the time and corresponds to a greedy approach. If the current node receives a message it has already sent, 3rule routing explores its neighbors, using the three rules. This ensures messages always reach destination.

### 5.3.3 Simulation and Analysis

We have shown that GFG does not guarantee delivery when node position-awareness is not perfect. In this section, we first show that, with perfect position-awareness, paths discovered by 3rule and GFG/GPSR are of equal length, i.e. both protocols perform equally well. We show how 3rule routing keeps a 100% delivery ratio while GFG/GPSR fail as positioning accuracy is degraded, or when the connectivity graph drifts away from a perfect UDG.

We implement GFG/GPSR, greedy geographic routing and 3rule routing in the home-made simulator. Unless otherwise stated, we position nodes in a square area of side 1000 units, each node having a communication range of 200. We vary the number of nodes in order to obtain the desired density, as described in Section 3.2.3. Presented results are averaged over $10^5$ runs, each run involving a new topology; source and sink nodes are chosen randomly at each run, among the already placed nodes.

**Hop count with perfect positioning.** The first set of results assumes perfect positioning and a UDG. Under these assumptions, both GFG and 3rule routing achieve 100% delivery ratio (to be shown in the next paragraphs). Fig. 5.10 shows a comparison of the length of the paths discovered by GFG/GPSR and 3rule routing, in number of hops, over a range of densities. It shows that 3rule performs as well as GFG/GPSR (3rules finds paths on average 0.4% shorter than GFG/GPSR).
CHAPTER 5. GEOGRAPHIC ROUTING OVER VIRTUAL COORDINATES

Figure 5.10: Average number of hops for reaching the sink when using routing protocols GFG and 3rule routing. As a basis for comparison, we provide $SP$ the hop count a centralized shortest path algorithm (such as the well-known Dijkstra algorithm) would have found. These simulation results were obtained by averaging $10^5$ runs and are presented with 95% confidence intervals.

**Impact of non-perfect positioning.** GFG/GPSR fail when nodes do not know their location with perfect accuracy, i.e. they do not deliver all sent messages. As 3rule routing uses the sequence of already traversed nodes as a guide to find the destination – which is independent from positioning accuracy, 100% delivery ratio is always guaranteed. This is depicted in Fig. 5.5.

**Impact of non-UDG.** As shown in Section 5.2, GFG/GPSR fail when they run over a non-UDG. Fig. 5.7 shows how 3rule routing delivers all sent messages, independently from the number of walls inside the deployment area (walls break the UDG assumption).

**Summary.** Greedy geographic routing is simple to implement, yet it fails when a void is met, a situation which appears often in low-density deployments (<7 neighbors per node). Face routing protocols, such as GFG or GPSR, extend the greedy approach by switching to a mode capable of circumnavigating the void when such a void is met. Nevertheless, the cornerstone of these proposals is the failure-prone planarization technique. Distributed – thus overhead-free – solutions rely heavily on the Unit Disk Graph assumption and fail dramatically when this assumption is broken. Variants such as CLDP function on arbitrarily random graphs but are not distributed, which is not desirable from an energy-efficiency standpoint.

We propose to use the sequence of already traversed nodes to help the routing algorithm. Three simple rules are used to filter through the list of neighbors, removing the ones which have already been used, to explore the graph. Even when not coupled to geographic information, this distributed depth-first search guarantees that a message eventually reaches destination. To lower hop count, the three rules are combined to geographic information to guide a new packet in the right direction. The resulting 3rule routing protocol performs as well as GFG/GPSR under perfect location-awareness, yet outperforms the latter when applied to arbitrary random graphs.
5.4 From Real to Virtual Coordinates

Some applications require each node to know (an approximation of) its real coordinates. These obtained approximated coordinates (using GPS, manual programming or localization protocols) cannot be used as such for routing because they are not related to the network topology. To answer this, a node can determine coordinates relatively to a set of location-unaware anchor nodes; we call these relative coordinates.

Discussion on relative coordinates. Nodes obtain their relative coordinates by measuring the topological distance in number of hops to a set of anchor nodes. As no node in the network knows its geographical position, the obtained relative coordinates locate the nodes inside the topology, not in the geographical space. Although relative coordinates are not related to geographical coordinates (which can be considered a drawback as they can not be used to location-stamp messages), they can be used very efficiently for routing.

When using a simple greedy approach, delivery ratio on these relative coordinates outperforms delivery ratio on real coordinates because less voids appear. Moreover, when these coordinates are properly aligned (i.e. when the gradient they form to the sink is smoothened out), routing over relative coordinates outperforms any routing technique over real coordinates. Real coordinates are thus never needed for routing purposes and should be replaced by the more efficient relative coordinates.

The use of relative coordinates has drawbacks. First, it requires anchor nodes to be identified in the network. Whether this is done by a human being when placing the nodes, or by a distributed election phase after roll-out does not change the fact that this induces costly overhead. Second, the paths discovered when using relative coordinates, although shorter than paths found by any routing protocol over real coordinates, are still about 10% longer than the optimal shortest path discovered by a fully centralized algorithm.

Greedy embedding in a graph. Research on applying non-real coordinates to wireless multi-hop nodes has been driven by the quest for a greedy embedding. A graph is defined as a set of vertices interconnected by edges. A greedy embedding of a graph is composed of the same edges interconnecting the same vertices, only the vertices are placed at coordinates such that greedy routing always functions when sending a message between arbitrarily chosen nodes (i.e. there are no void areas).

The notion of greedy embedding was developed by Papadimitriou and Ratajczak [Papadimitriou and Ratajczak, 2004], who studied the special case of the Euclidian space. They provided examples of graphs which do not admit a greedy embedding in the Euclidean plane, yet they conjectured that every 3-connected planar graph admits a greedy embedding in the Euclidean plane.

Kleinberg has extended this work by showing that every connected finite graph has a greedy embedding in the hyperbolic plane [Kleinberg, 2007]. The underlying algorithm, however, assumes that the network is capable of computing a spanning tree rooted at some node. Although a fair assumption (distributed protocols for computing a spanning tree are abundant in the literature and in practice), using a spanning tree requires the network to maintain this structure, which may be hard and costly. Moreover, in theory, the worst-case path stretch (the ratio of the number of hops of discovered paths to the number of hops on the shortest route between the same pair of nodes) increases linearly with network size.

Introducing virtual coordinates. The solution we propose does not require an initialization phase. This means that it is functional as soon as the network is deployed. The nodes use virtual coordinates which are updated throughout the network lifetime. No network-wide periodic
updates are needed, and the system is extremely robust against nodes (dis)appearing and link dynamics. When using 2-dimensional virtual positions, the path stretch is typically a few percent; when using 1-dimensional virtual heights, the path stretch converges to 1, which is the optimal.

As the nodes’ virtual coordinates are constantly updated, there is no distinct localization phase followed by a routing phase, as it is the case when using approximated relative coordinates. This significantly increases the network’s robustness as any topological change is immediately reflected onto the nodes’ virtual coordinates. This also means that localization (i.e., nodes determine their virtual coordinates) and routing (i.e., a geographic routing protocol uses these virtual coordinates to find a path to the destination) are intertwined and happen at the same time.

Virtual coordinates do not rely on costly anchor nodes, but rather on an updating process continuously carried out throughout the lifetime of the network. The generic steps are described below:

1. 3rule routing is applied on top of the virtual coordinates. Remember that 3rule routing guarantees delivery.

2. When a node is switched on, it chooses arbitrary virtual coordinates without communicating with its neighbors. In the academic case that all nodes are switched on at the same time, this leads to virtual coordinates being independent from the topology. Although all sent messages reach destination (thanks to 3rule routing), hop count is clearly suboptimal.

3. Whenever a node sends a message, it updates its virtual coordinates as a function of its neighbors’. This updating process repeats for every sent message throughout the lifetime of the network.

4. The sink node(s) have/has a different behavior. When switched on, it picks virtual position \( \{0, 0\} \) (for 2D virtual coordinates) or \( \{0\} \) (for 1D virtual heights). It never changes its coordinates, always staying at \( \{0, 0\} \) or \( \{0\} \).

Note that the generic term is virtual coordinate, which we subdivide in 2D virtual position and 1D virtual heights for clarity. The design-drivers behind virtual coordinate routing are the following.

**Reactivity.** A node has virtual coordinates from the moment it is switched on. Although these initial coordinates may not reflect the topological position of the node in the network, they can be used for routing immediately. The absence of an initialization phase enables the network to be operational right after deployment, which can be extremely beneficial for applications such as disaster recovery WSNs.

**Convergence.** Virtual coordinates are updated as messages flow through the network. The updating process causes the virtual coordinates of the nodes to converge. Once converged, nodes topologically far from the sink are also virtually far from it\(^2\), and vice-versa. The 3rule routing protocol finds paths which are close to optimal in number of hops.

**Robustness.** Virtual coordinates are continuously updated throughout the lifetime of the network. They immediately reflect topological changes due to nodes (dis)appearing, mobility or link dynamics.

\(^2\)We call virtual distance the Euclidian distance measured using the virtual coordinates.
Energy-efficiency. The updating process goes on only when a message is sent. This reactive approach is especially beneficial when the network load is low or bursty, in which case a more traditional periodic approach would lead to major energy consumption. Moreover, the updating process can easily be coupled with the MAC layer to guarantee energy-efficiency (such a cross-layered architecture is presented in Chapter 6).

Overhead-free relocation of the sink node. The sink node always places itself at virtual height \{0\} or virtual position \{0, 0\}, a position a priori known to all nodes. As a result, the 3rule routing protocol always knows the location of the sink node, which does not have to advertise its position through network-wide broadcasting. This characteristic is extremely beneficial in case the sink is mobile. Section 6.2 presents experimental results where a sink node is mounted on a fast moving radio-controlled airplane.

Using multiple sink nodes. As will be described in the next sections, the virtual coordinates of the nodes in the network structure around the sink node. In case there are multiple sink nodes, the virtual space is structured in such a way that a node sends its message to the topologically closest sink node. Moreover, because the virtual space adapts to topological changes, sink nodes can be added, removed or relocated in the network without requiring any reconfiguration. Section 6.3 presents experimental results where sink nodes are added and removed at run-time without disrupting network operation.

5.5 2D Virtual Positions

A connectivity graph is a drawing where nodes are vertices and edges interconnect nodes which are able to communicate. We call real graph the connectivity graph where nodes are placed at their real coordinates. We call virtual graph the connectivity graph with nodes placed at their virtual coordinates. This is the graph the 3rule routing protocol uses for routing. Real and virtual graphs are composed of the same vertices and edges, only the vertices are placed at different positions.

5.5.1 Initialization and Iterative Convergence Process

When a message travels from source to destination, it jumps from one node to another. The multihop path followed by the message is determined by the routing protocol, which bases its decision on the virtual positions of the nodes. In the following two paragraphs, we describe how nodes initialize their virtual positions, considering they do not know anything about the network at startup.

When the nodes are started up (probably in an unsynchronized way), each one randomly chooses its virtual position as it does not know its real coordinates. A virtual position is a pair of floating point positive numbers each chosen within some range, e.g. \([0 \ldots 1000]\).

Should the sink have the same behavior, it would need to broadcast its virtual position to all the nodes in the network for them to know where to send their message to. The sink would need to rebroadcast each time it updates its virtual position, which would be very costly. Therefore, the sink node has a slightly different behavior. When it is switched on, it always chooses virtual position equal to \{0, 0\}, and never changes them. As a consequence, nodes always know where to send their messages to, independent from the sink’s real position.

Whenever a node transmits its own or a relayed message, it updates its virtual position according to the transformation we describe in the following paragraphs. We call this updating process the centroid transformation. As shown in the lower left graph of Fig. 5.12, virtual positions are random when the nodes are switched on and the initial virtual graph looks rather erratic. The
run-time centroid transformation is used to alter this graph, trying to bring physically connected neighbor nodes closer to each other on the virtual graph.

The run-time centroid transformation uses the three following steps. Each time a node has a message to send, it

1. retrieves its neighbors’ virtual positions;

2. updates its virtual position with the average of its neighbors’ virtual positions, i.e. it sets its virtual position at the center of gravity of its neighbors’;

3. if after step 2 the current node $V$ has a neighbor $W$ virtually closer to it than a threshold distance $MinVirtual$, $V$ tells $W$ to update its location according to Eq. (5.8). This equation calculates node $W$’s new virtual position $\{W_1^*, W_2^*\}$ as a function of its old position $\{W_1, W_2\}$.

   \[
   \begin{align*}
   W_1^* &= (W_1 - V_1) \cdot \frac{MinVirtual}{\sqrt{(V_1-W_1)^2+(V_2-W_2)^2}} + V_1 \\
   W_2^* &= (W_2 - V_2) \cdot \frac{MinVirtual}{\sqrt{(V_1-W_1)^2+(V_2-W_2)^2}} + V_2 
   \end{align*}
   \] (5.8)

Step 1 is not specific to our solution. We assume neighborhood discovery to be part of the MAC layer. In Fig. 5.11 (1), the current node is $V$. After step 1, it knows it has three neighbors $W$, $X$ and $Y$, and knows what their virtual positions are.

Step 2 brings neighbor nodes virtually close to each other. The intuition is that if all nodes do so, a node gets close to its neighbors, which in turn get close to theirs, etc. The resulting convergent behavior is that nodes on a path towards a destination align. Once aligned, a geographical routing protocol follows this path. We call network convergence the transition between the erratic initial and a fully aligned virtual graph. Step 2 is a local calculation which does not need communication. Fig. 5.11 (2) shows how node $V$ updates its virtual position with the center of gravity of its neighbors’.

After step 2, and as part of the routing process, the current node sends out the data message towards its neighbor selected by the routing protocol as next hop. This packet contains in its header the current node’s updated virtual position. As the wireless medium is broadcast by nature, all the current nodes’ neighbors can hear this new virtual position. This, however, supposes a MAC protocol – such as 1-hopMAC – in which nodes are listening to at least the header of the data packet. We believe that this is a very generic feature of current MAC protocols.

Step 3 has no impact on the fact that the network converges; however, it helps implementing our protocol on real hardware where virtual positions are represented by numbers with limited resolution. With finite numbers representing the virtual position, nodes virtually close to each other may end up having the same sequence of bits representing their virtual positions. If those two nodes are neighbors, it creates ties in the routing process. To avoid this, neighboring nodes need to be separated by a virtual distance larger than the resolution of the binary representation of the virtual position. We call this distance $MinVirtual$. $MinVirtual$ can be arbitrarily large, and is chosen as a function of the resolution of the binary representation of the virtual position.

Step 3 ensures that neighbor nodes are separated by a virtual distance at least equal to $MinVirtual$. This step does not require communication. All neighbor nodes receive the new virtual position from the current node. All neighbor nodes then check that they are not virtually closer to the current node than $MinVirtual$. If so, they apply Eq. (5.8) to recalculate their own virtual position. Eq. (5.8) is a homothetic transformation with center the current node and ratio of magnification $\frac{MinVirtual}{\sqrt{(V_1-W_1)^2+(V_2-W_2)^2}}$. This transformation causes the neighbor node which is too close to slide
its virtual position away until it is at virtual distance \( \text{MinVirtual} \) from the current node. Its new virtual position stays on the line formed by the current node and its old virtual position. This is represented in Fig. 5.11 (3): After step 2, node \( X \) ends up too close to \( V \). Node \( X \) thus updates its virtual position to be at distance \( \text{MinVirtual} \) from \( V \). After updating its virtual position, node \( V \) does not need to send them to its neighbors.

**Explaining network convergence.** Before presenting the performance results when using the presented virtual positions, we believe it is important to discuss the intuition behind it. As detailed in the introductory part of this chapter, self-organization in WSN is shifting from complex to lightweight protocols. The functionality of the latter comes from the emergence of a network-wide behavior as a consequence of the (simple) interactions between neighbor nodes.

Analogies can be seen between the presented protocol and the behavior of animal swarms. In a bird flock, a single bird can only see its neighbor birds, and knows where they are. To keep the flock together, a bird moves equally close to each of its neighbors; yet, to avoid collision, it stays at a safety distance. The parameter \( \text{MinVirtual} \) represents this safety distance. Without it, as nodes update their virtual positions, they would get virtually closer to one another and closer to the sink. Virtual positions would take infinitely small values, which are hard to handle by the fixed-point computation unit typically found in the microprocessors/microcontrollers at the heart of wireless sensors.

The flock forms a homogeneous structure, all birds following a leader in the front [Couzin et al., 2005]. Our system adopts the same strategy. As described in the next paragraphs, the virtual positions of the nodes align, the "leader" role being played by the sink node.

To help the reader visualize such emergent behavior, we refer to Fig. 5.12. This figure is obtained by simulating the behavior of 100 nodes, randomly scattered within a square space of side 1000 units, each node having a 200 unit radio range. The upper part shows the real graph, i.e. nodes are positioned at their real coordinates with edges interconnecting nodes able to communicate. The lower part represents the virtual graph, i.e. the same vertices and edges as in the upper drawing, only nodes are positioned at their virtual positions. We show snapshots of the virtual graph after 0, 100 and 500 messages are sent. Each of these messages is sent from a randomly chosen node (different for each message) to the sink node.
Figure 5.12: For the same real graph (upper part), we draw the virtual graph after 0, 100 and 500 sent packets (from left to right) with $\text{MinVirtual} = 40$.

The initial virtual graph (Fig. 5.12, lower left) looks erratic as virtual positions are initially chosen randomly. As the number of sent messages increases, the virtual positions of the nodes tend to align. While messages flow through the network, neighbor nodes are brought virtually closer to one another. In the resulting linear structure, nodes topologically close to the sink node are also virtually close, and vice-versa. Once connected, virtual distance to the sink is closely related to the minimum number of hops to the sink. As we will see in the next paragraphs, using geographic routing protocols on these virtual positions yields near-optimal path length. In these simulations, we used $\text{MinVirtual} = 40$. This guard distance causes the virtual positions to expand, i.e. nodes are virtually farther from the sink after 500 messages sent than after 100.

5.5.2 Demonstrating Network Convergence

The initial virtual graph is erratic (see Fig. 5.12, low-left corner), and the nodes potentially fill the entire square. We show that, under the assumption that $\text{MinVirtual} = 0$, the virtual graph converges to a linear virtual graph. In this state, the virtual positions of all nodes are on a line. An example of such completely converged network can be seen in Fig. 5.13 (b), obtained by simulation. To simplify the analysis, we temporarily consider $\text{MinVirtual} = 0$, thus removing step 3 from the updating process. We consider the network is composed of $N$ nodes.

We define $\theta_{\text{min}}$ and $\theta_{\text{max}}$ as follows (see Fig. 5.13 (a)):

$$
\begin{align*}
\theta_{\text{min}} &= \min \left( \arctan \left( \frac{y_i}{x_i} \right) , 0 < i < N \right) \\
\theta_{\text{max}} &= \max \left( \arctan \left( \frac{y_i}{x_i} \right) , 0 < i < N \right)
\end{align*}
$$

(5.9)

As depicted in Fig. 5.13 (a), the lines passing through the sink node and forming angles $\theta_{\text{min}}$
and $\theta_{max}$ with the lower side of the network form a cone which contains all the nodes in the network.

Let’s consider the updating process. In particular, let’s see how $\theta_{min}$ and $\theta_{max}$ evolve over time. Both values only change if the updating process affects the node defining this angle (in Fig. 5.13 (a), nodes $A$ and $B$ for angles $\theta_{min}$ and $\theta_{max}$, respectively). We call $N_A$ and $N_B$ the set of neighbor nodes of $A$ and $B$, respectively.

We focus on $\theta_{min}$, the same analysis applies for $\theta_{max}$. As $\tan(\cdot)$ is a strictly increasing function over $[0, \frac{\pi}{2}]$, we have $\frac{y_A}{x_A} \leq \frac{y_i}{x_i} \forall i \in N_A$. We call $[x_A^*, y_A^*]$ the new virtual position of $A$, yielding (5.10).

$$\frac{y_A^*}{x_A^*} = \frac{\sum_{i \in N_A} y_i}{\sum_{i \in N_A} x_i} \geq \frac{\sum_{i \in N_A} x_i y_A}{\sum_{i \in N_A} x_i x_A} = \frac{y_A}{x_A}.$$  

(5.10)

$\theta_{min}$ hence increases. A similar analysis shows that $\theta_{max}$ decreases. We have $\lim_{m \to \infty} \theta_{max} - \theta_{min} = 0$, where $m$ represents the number of sent messages. Network convergence is hence achieved where the network converges to a linear virtual graph, as depicted in Fig. 5.13 (b) obtained by simulation.

This analysis still holds with $MinVirtual > 0$. The convergence of the system is equivalent, only neighbor nodes are never virtually closer than $MinVirtual$. The cone always stays slightly open. After the convergence, nodes are still aligned, and the results of the next section applies. Note in Fig. 5.13 (b) the values of the axis; with $MinVirtual = 0$, the network converges to the sink node and each virtual position hence needs to be very accurately represented. This justifies step 3 of the run-time centroid transformation.

Demonstrating Near-Optimality of the Resulting Path Lengths We have shown how the network converges from an erratic to a linear virtual graph as packets flow through the network. We now show that the nodes’ virtual positions are ordered properly along this line. Remember that the sink node never updates its virtual position, but rather stays at virtual position $\{0, 0\}$. We call the height of a node the minimum number of hops separating this node from the sink (although this parameter is used during the explanation, it is not known by the node). We define virtual sequence as the sequence of nodes from virtually close to virtually far from the sink node. We say that a virtual sequence is correct if it matches the heights of the nodes. That is, on a given path to the sink, a node with small height should appear before its neighbor with larger height in the virtual sequence. Note that if the virtual sequence is correct, a geographic routing protocol follows the
optimal route in terms of number of hops. The basic principle of a geographic routing protocol is to elect the neighbor node virtually closest to destination as next hop.

Let’s start by a chain topology, as the one depicted in Fig. 5.14 (a). After convergence of the virtual graph, the virtual positions of all nodes are on the same line. We argue that in this case the virtual sequence is correct. This can easily be proven by showing the opposite statement is not possible. Let’s consider the virtual sequence is $A-C-B-D$ ($B$ and $C$ have been swapped). As node $B$’s neighbors are $A$ and $C$, next time it updates its virtual position, it will appear between $A$ and $C$ in the virtual sequence. Moreover, any permutation can be seen as successive permutations between neighbors. In Fig. 5.14 (a), permuting $B$ and $D$, leads to a virtual sequence $A-D-C-B$ being equivalent to 3 individual permutations: $A-B-C-D \rightarrow A-C-B-D \rightarrow A-C-D-B \rightarrow A-D-C-B$. The proof thus stands for any permutation. Finally, note also that once the virtual sequence is correct, there exists no additional centroid transformation which changes the virtual sequence. The state where the virtual sequence is correct is hence a steady state. With a chain topology, an incorrect virtual sequence is only possible during the convergence of the network, yet in steady state the virtual sequence is always correct.

We extend this analysis to the star topology shown in Fig. 5.14 (b). As there are no links between the branches, there is no effect from the centroid transformations performed on one branch on another branch. Furthermore, since the sink node’s virtual position is fixed, the updated virtual position of a node can only affect the virtual positions of nodes in the same branch, and cannot propagate to other branches. As the branches are independent, they can be considered as independent chain topologies. Hence, on a star topology, the virtual sequence is always correct.

We call generalized tree a star topology where several branches share some of their nodes. Fig. 5.14 (c) is an example of a generalized tree. A property of this type of topologies is that edges interconnect nodes with a difference of height exactly equal to one. On the virtual graph, all nodes of height $h$ are always virtually closer to the sink than neighbor nodes of height $h+1$. Let’s take the example of node $I$ and $K$ in Fig. 5.14 (c). Independently from whether $I$ or $K$ is closer to the sink, the message always follows a shortest path (either $J-I-H-A$ or $J-K-H-A$).

Any topology can be seen as a generalized tree topology with links interconnecting nodes of the same height, as shown in Fig. 5.14 (d). From those links, virtual graph incorrectness may arise. Let’s take the example of link $D-J$. It is possible that $J$ ends up virtually closer to the sink than $C$. In this case, the packet would follow path $D-J-I-H-A$, which is one hop longer than the optimal path. The virtual sequence may be incorrect in this case. We evaluate how this affects the

Figure 5.14: Demonstrating near-optimality of the resulting path lengths. An identifier ($A$, $B$, etc.) and its height are attached to each node. We show both real (upper) and virtual (lower) topologies. The virtual graphs are linear in reality, but for clarity they are shown expanded vertically.
efficiency of our protocol in the next section.

5.5.3 Simulation Results and Discussion

We evaluate the efficiency of the proposed approach from two complementary angles. We start by looking at its energy efficiency. We divide energy efficiency in two parts: at node level (including processing power, hardware, medium access, and memory) and at network level (taking into account the number of hops to reach destination). We end this section by looking at our proposal’s performance with a more realistic propagation model.

Energy-efficiency at node level. Our protocol is near-overhead free. In terms of memory, geographic routing is known to scale well as the nodes need to store only their own and their neighbors’ positions. This amount of data is constant regardless of the number of nodes in the network, provided the average number of neighbors stays the same as the network grows. Because nodes are location-unaware, no specific positioning hardware is needed and our protocol can hence be implemented on standard commercially available motes. This reduces the cost of the individual sensor boards. As for computational power, operations are basic. Finally, the impact of communication is negligible. Our virtual position updating process re-uses the neighborhood information which is required anyway by the routing protocol. It does require the current node to append its identifier into the data packet header, but this should not exceed a couple of bytes. We have implemented our system on real motes and confirmed that the energy spent for transmitting and receiving these extra bytes accounts for only 0.45% of a node’s total energy budget.

Simulation methodology and metrics. To fully understand the impact of each parameter, we used simulation on the home-made packet-level simulator. To measure only its performances, we isolate the routing protocols by using an idealized MAC (neither collisions nor transmission delays are taken into account). Our proposed scheme aims at finding the shortest path and does not interfere with real MAC protocols.

At each simulation, 100 nodes are placed randomly in a $1000 \times 1000$ units square area. This topology is different for each run, and we average out the results over a large number of runs. We calculate and depict a 95% confidence interval, which is the interval within which 95% of the individual results fall. The smaller the size of the 95% confidence interval, the more "confidence" one can have in presented averaged results.

Each node randomly chooses a virtual position. Messages are sent from randomly chosen connected nodes to the sink. At each hop, the current node updates its virtual position. We use the number of sent messages as a metric of time (to quantify speed of convergence). It is easy to convert the number of message back to time if the message generation rate is known.

We evaluate the energy efficiency of routing protocols by assimilating it to hop count. We implicitly assume that each hop consumes the same amount of energy, independently from the routing protocol used. If the number of hops is lower, the energy spent for sending the messages is lower, and the overall lifetime of the network is extended.

Many geographical routing protocols have been proposed. The most widely adopted ones are GPSR and GFG. They guarantee delivery under a perfect unit disk graph assumption (where the communication area of a node is a circular disk). We use GFG/GPSR as a baseline comparison to our protocol.

Comparing multiple protocols is a complex task. In most cases, it is easy to find a set of parameters which are favorable to one or another protocol. In order to present an unbiased comparison, most of our simulation results are presented in cases where GFG/GPSR give good results. In particular, GFG/GPSR present a low hop count when the network density is high as voids are less
CHAPTER 5. GEOGRAPHIC ROUTING OVER VIRTUAL COORDINATES

Figure 5.15: (a) The average path stretch when using the 3rule and the GFG/GPSR protocols. This graph was obtained for \( \text{MinVirtual} = 40 \). (b) This figure is a cumulative version of the figure on the left: it shows the total number of hops consumed by the network since network initialization. The cross point indicates the moment at which our solution is more energy-efficient than the GFG/GPSR protocols.

Convergence and energy efficiency at routing layer. As shown in Fig. 5.15, the communication architecture converges to near-shortest path as the number of sent messages increases. Although the number of nodes traversed by the first messages is large, the hop count quickly drops and converges to near-optimality. We plot the hop count of our system together with the results for GPS/GPSR. After about 170 messages sent, our system outperforms GFG/GPSR in terms of hop count. The system converges to a state where the path stretch is 1.04, meaning that the path length of our system exceeds the shortest path by only 4%.

Convergence takes some time during which hop count is high, leading to a high number of hops during network ramp-up. Fig. 5.15 (b) plots the cumulative normalized hop count. This is proportional to the total energy spent since network initialization. As during network ramp-up our solution consumes more hops than GFG/GPSR, the latter should be chosen if less than 700 messages are sent over the 100-node network (for this set of parameters). Yet, for most WSN applications, a network is deployed for a long period of time, and our solution is more energy efficient than GFG/GPSR. For instance, environmental monitoring solutions are designed to operate over about 15 years and to deliver a sensor reading twice a day, leading to 15 years \( \times \) 365 days/year \( \times \) 2 messages/day/node \( \times \) 100 nodes \( \approx \) 1 million messages; this translates to a gain of 13.3% in terms of path stretch when using our approach compared to GFG/GPSR.

Impact of network density. Geographic routing protocols need to tackle the well-known void problem. This problem occurs when the current node does not have neighbor nodes closer to the destination than itself. In this case, a simple greedy approach, in which the next hop is the neighbor closest to destination, fails. Protocols such as GFG or GPSR then switch to a second mode to circumnavigate the void area. Although this guarantees message delivery, finding a path to circumnavigate the void leads to a high hop count. As voids appear more often in sparse networks, GFG/GPSR does not perform well under low network density conditions.

To diminish the average number of neighbors from 11 to sparse setting of 4, we use here a
Figure 5.16: This figure is equivalent to Fig. 5.15, where only the average number of neighbors has been decreased from 11 to 4.

Figure 5.17: (a) Path stretch with node appearing/disappearing. (b) Cumulative hop count shows that our solution quickly outperforms GFG/GPSR.

Impact of nodes (dis)appearing. Wireless sensors are unreliable in essence. They may disappear from the network for reasons as different as hardware failure, battery exhaustion, theft or destruction by natural elements. Similarly, the administrators of the network may decide to add new nodes after some time. It is therefore essential that communication protocols be robust against nodes (dis)appearing. They should present self-reconfiguration, self-healing and self-management characteristics.

In this paragraph, we simulate a network deployed in a harsh environment, where a disastrous event (for example an earthquake) randomly destroys 30% of all nodes. After that, rescue teams redeploy the same amount of nodes (at random locations in the network). We wanted these two events to happen at times sufficiently separated so that we could see whether the network re-converged after these major topological changes. We chose to have these two events happen after the 200th and the 350th message.
In Fig. 5.17, we plot the path stretch and the cumulative path stretch under this scenario, for both our approach and GFG/GPSR. The results are in line with the previously presented results. After a main topological change, our results re-converges to a near-optimal state. This is a key behavior of our self-organization system as it shows it is entirely self-healing.

There is no convergence for GFG/GPSR as these are stateless protocols. As can be seen on Fig. 5.17 (b), the topological changes have nearly no impact on the network’s energy consumption when using our solution. This means that our solution not only survives major topological changes, but it does so without significantly affecting the expected network lifetime. This behavior is particularly interesting for network administrators, who need to estimate the network’s lifetime to predict when to redeploy new nodes. Major topological changes do not significantly affect the expected lifetime.

A more realistic propagation model. Protocols such as GFG/GPSR use the left-hand rule to circumnavigate void areas. To function, this is used over a planar subset of the connectivity graph, obtained by locally applying the Gabriel Graph transformation. As has been detailed in Section 5.2, this assumption does not hold in real systems, and these protocols fail dramatically.

Our solution uses 3rule routing as a basis. 3rule routing guaranteed delivery independently from localization accuracy (see Section 5.3), including when virtual coordinates are used. Figs. 5.5 (page 81) and 5.7 (page 83) hold for virtual position routing. They show how virtual position routing delivers all sent packets, while GFG/GPSR fail as localization accuracy degrades and the UDG assumption is broken.

5.6 1D Virtual Heights

In a WSN, a node sends it message to a sink node in what is really a one-dimensional problem. It is possible to have 1D virtual height, i.e. a node is attached a single virtual coordinate which reflects its topological distance to the sink node. Similarly to the 2D case, nodes pick their virtual height when they are switched on, without involving communication. Whenever it sends a message, a node updates its virtual height as a function of its neighbors’.

The 1D virtual height solution uses a different updating process than the 2D virtual position solution, as detailed in the next section.

5.6.1 Initialization and Iterative Convergence Process

The mono-dimensional virtual height routing scheme goes as follows. When switched on, a node sets its virtual height at $\text{NULL}$, a non-value. Whenever a node sends a message, it scans through the virtual heights of its neighbors, finds the smallest one, and updates its virtual height by this value incremented by one. We call this updating process $\text{min+1}$. The 3rule routing protocol is used on top of the virtual height structure and uses a node’s virtual height as a direct metric of proximity to the sink.

Instead of being two floating point value in the 2D case, a node’s virtual height is a single integer value in the 1D case, simplifying its binary representation during implementation. Moreover, operations in the 1D case are simpler and can be performed faster by the node’s simple microcontroller. Finally, as we show in the next section, the system converges to the optimal case, in which discovered path are optimal in the general case.
5.6.2 Demonstrating Network Convergence

Let us consider all nodes are switched on at the same time. When this happens, the sink node chooses virtual height \(0\), all other nodes choose NULL. The first message causes the sink’s neighbor relaying the message to take virtual height \(1\). After a few of those messages, all the sink’s neighbors are at virtual height \(1\) while their neighbors have taken virtual height \(2\). As messages travel across the network, virtual heights are set and increase in concentric topological circles away from the sink.

1D virtual heights and gradient based routing. Once the network has converged (i.e. all nodes have acquired a virtual height), what has been set up is a gradient. Like in gradient based routing, each node is attached a number which represents the number of hops separating it from the sink node. By following the largest gradient, i.e. by choosing the neighbors node with lowest virtual height as next hop, the discovered path are shortest paths. After the network has converged, routing is thus done optimally. The virtual height scheme, however, differs greatly from gradient based routing in the following aspects:

- Gradient based routing uses an initialization phase, in which the sink node broadcasts a gradient setup message propagated throughout the network. This is impractical. As links constantly evolve over time, the gradient is rapidly outdated at the initialization phase needs to be redone. Periodically reconstructing the gradient is time and energy consuming. Virtual height routing does not use an initialization phase and is entirely self-healing, constantly reflecting topological changes by adjusting the node’s virtual heights.

- Unlike gradient based routing, virtual height routing requires no signaling messages. Moreover, as we describe in Chapter 6, cross-layer optimization is easy to do, enabling MAC and routing layer to interact closely, forming an energy-efficient communication architecture.

5.6.3 Simulation Results and Discussion

Using the same simulation setting before, Fig. 5.18 shows the path stretch for different densities. Comparison with Figs. 5.15 and 5.16 show that (1) 1D virtual heights converge faster that 2D and (2) 1D virtual heights converge to optimal, a path stretch of 1.

These results are extremely encouraging as they show a stateless\(^3\) and distributed protocol is able to find optimal routes while being self-healing and energy-efficient. We present some further results and discussions in the following paragraphs.

Impact of network density. We perform the following simulations to evaluate the impact of network density. Nodes are randomly positioned in a 1000x1000 area, each having a 200 unit communication range. The number of nodes is chosen as a function of the desired density. 1000 runs are performed. For each run, a new topology is generated and new source/sink nodes are randomly chosen. We calculate for each density the average path stretch over the 1000 first messages. In all cases, the path stretch converges to one. The average path stretch is greater than one as the first 100 messages or so follow long paths. The greater the average path stretch, the slower the convergence.

\(^3\)Strictly speaking, virtual height routing is not stateless, yet the only state kept is the virtual height, a small integer value.
Table 5.1: Density has a very limited impact on speed of convergence on 1D virtual height routing.
The results shown in Table 5.1 suggest that, although higher density networks tend to converge faster, the impact is negligible. For these settings, all runs have fully converged after about 100 messages.

**Impact of network size.** As discussed in Section 5.6.2, nodes acquire their virtual height from the sink node outwards. The size of the network in number of hops should thus influence the convergence speed: the more hops the furthest node is from the sink, the longer the network takes to converge. We simulate topologies with different radii (i.e. the maximum number of hops between a node and the sink), and measure the average path stretch of the first 1000 messages. Simulation results are averaged over 1000 independent runs.

The results in Table 5.2 show that, although smaller networks are quicker to converge, the impact of network size of converge speed is limited.

**Impact of non-UDGs.** Unit disk graphs are assumed for GFG/GPSR as the planarization technique these protocols use fail if this assumption is not met. The 1D virtual height routing technique does not assume UDGs and performs equally well when walls are introduced in the deployment area.

**Using multiple sinks.** The routing protocol sends a message to the current node’s neighbor with lowest virtual height; sink nodes all have virtual height at \( \{0\} \), the virtual heights of the other nodes increase as the are topological far from the sink. Additional sink nodes can be introduced without requiring any configuration, as shown in Fig. 5.19. Nodes send their messages to the sink topologically closest.

Alternatively, by introducing a second height\(^4\), nodes can choose to which sink nodes a message should be sent. This is represented in Fig. 5.20. Each node has two virtual heights, one for each sink node. The application layer can choose the destination sink node by choosing which of the two virtual heights is used.

**Introducing an initialization phase?** As shown in the previous paragraphs, it takes a number of messages for the network to converge. This represents an extra energy expenditure which could easily be reduced by having the sink node build a gradient during network ramp-up. This initialization phase would function as follows. Right after all the nodes have been switched on, and before the first packet is sent, the sink node broadcasts a gradient setup message which is relayed by all the nodes, incrementing an internal counter at each hop. Each node sets its virtual height to the counter in the gradient setup message, which represents the number of hops separating it from the sink node. The network is considered converged after this network-wide broadcast is performed.

\(^4\)Note that these are two 1D integer virtual heights, both updated using the min+1 updating process. This is different from 2D virtual positions.

<table>
<thead>
<tr>
<th>number of nodes</th>
<th>average path stretch</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1.05 (baseline)</td>
</tr>
<tr>
<td>200</td>
<td>1.11 (+5.71%)</td>
</tr>
<tr>
<td>300</td>
<td>1.18 (+12.4%)</td>
</tr>
</tbody>
</table>

Table 5.2: Network size has a limited impact on speed of convergence on 1D virtual height routing.
CHAPTER 5. GEOGRAPHIC ROUTING OVER VIRTUAL COORDINATES

Figure 5.19: With 1D virtual heights, multiple sinks – represented as squares – can be used in parallel without requiring any configuration. Nodes – represented as circles – implicitly send their message to the sink topologically closest. Numbers between curly brackets represent the node’s virtual height, once converged. The fill pattern indicates to which sink a node sends its data. Some nodes may send their data to either of the sink nodes, these are represented with both fill patterns.

Figure 5.20: Nodes can explicitly choose the destination sink node when an additional 1D virtual height is used. Numbers between curly brackets represent the node’s two 1D virtual heights, one for each sink. When sending a message, a node decides on which virtual height this should be sent, hence choosing the destination sink node.
Gradient setup techniques such as the one proposed in [Ye et al., 2001a] require each node to relay only one message. Using such an initialization phase can hence reduce the energy expenditure of the network. Nevertheless, we argue that such an initialization phase makes little or no sense from a practical point of view:

- This initialization phase is not required as the network converges – albeit slower – through the continuous update of the virtual heights. The energy gains should hence be weighted against protocol complexity. The simpler the protocol, the less error-prone its implementation is;

- Assuming all nodes are switched on at the same time is not realistic. In a typical network roll-out, nodes are switched on inside a box and placed at their correct position one after the other. Asking the sink node to setup up the gradient can only be done once all the nodes are placed at their final location. Yet, at that time, the network has converged through the virtual height updating process\(^5\) and no gradient setup is needed.

For these reasons, we did not opt for the optional initialization phase.

**Choosing 1D virtual heights or 2D virtual positions.** Virtual coordinates come in two flavors: 2D virtual positions and 1D virtual heights. While 2D virtual positions cause the network to converge quickly to a near optimal state (paths found are typically a few percent longer than shortest paths), 1D virtual heights converge faster and converge to an optimal state. From a purely efficiency point of view, 1D virtual heights should be used over their 2D counterpart.

We have chosen to present 2D virtual positions because we think this approach can be used efficiently as a basis for hybrid coordinates. Hybrid coordinates, still an open field of research, are coordinates which are somewhere between real coordinates and coordinates used for routing. Because they would be close to the node’s real coordinates, they could be used by the application to location-stamp messages; because they would be updated using a updating process close to the one presented here, less void areas would appear and hybrid coordinates would be more efficient than real coordinates for routing.

We see two possible ways of exploring hybrid coordinates. In the first, we assume nodes are location-aware by some means. An updating process would then slightly alter the coordinates of nodes on the boundary of void areas to increase delivery ratio of a greedy routing approach. The key is not to alter the coordinates too much so they can still be used to location-stamp application messages. The second way is to use location-aware anchor nodes coupled with a continuous updating process as proposed in this chapter. An interesting candidate is a weighted centroid transformation in which each node would have a weight representing a degree of confidence in its coordinates. Location-aware anchor nodes would have a very high weight; weight would decrease as nodes get topologically further from the sink nodes.

### 5.7 Summary

Greedy geographic routing is simple to implement, yet it fails when a void is met, a situation which appears often in low-density deployments (<7 neighbors per node). Face routing protocols, such as GFG or GPSR, extend the greedy approach by switching to a mode capable of circumnavigating the void when such a void is met. Nevertheless, the cornerstone of these proposals is the failure-prone planarization technique. Distributed – thus overhead-free – solutions rely heavily on the Unit Disk Graph assumption and fail dramatically when this assumption is broken. Variants such

\(^5\)This is confirmed during the Sense&Sensitivity experiment presented in Section 6.3, page 116.
as CLDP function on arbitrarily random graphs but are not distributed, which is not desirable from an energy-efficiency standpoint.

We propose to use the sequence of already traversed nodes to help the routing algorithm. Three simple rules are used to filter through the list of neighbors, removing the ones which have already been used, to explore the graph. Even when not coupled to geographic information, this distributed depth-first search guarantees that a message eventually reaches destination. To lower hop count, the three rules are combined to geographic information to guide a new packet in the right direction. The resulting 3rule routing protocol performs as well as GFG/GPSR under perfect location-awareness, yet outperforms the latter as it functions on arbitrary random graphs.

Geographic routing protocols trade delivery ratio against path length. Relative coordinates were introduced, in which nodes infer their location relatively to a set of location-unaware anchor nodes. As relative coordinates reflect topological rather than geographical position, they are more suited for routing. As a result, using geographical routing over relative coordinates yields higher delivery ratio and shorter paths than over real coordinates. If, moreover, relative coordinates are correctly aligned, paths found are as short as 110% the shortest path.

We went one step beyond and proposed virtual coordinates. Using these coordinates does not require a costly initialization phase, but rather relies on the continuous update of the nodes’ virtual coordinates. This continuous update also makes the solution robust, as topological changes are immediately reflected in the nodes’ virtual coordinates. Because updates are performed entirely on-demand and piggybacked inside the messages, the solution is nearly overhead-free. This energy-efficient scheme self-configures the network around the sink node, regardless of its physical location. This enables the sink node to be relocated without requiring any configuration. Additional sink nodes can be added to the network transparently, a sensor simply sends its data to the closest sink node.

Virtual coordinates can either be 2D virtual positions or 1D virtual heights. In both cases, a node chooses its virtual position/height when switched on without communicating with its neighbors. Whenever it sends a message, it retrieves its neighbors’ virtual position/height and updates its own in function. From an initial erratic state, the network is said to converge when the nodes’ virtual position/heights structure around the sink nodes. At the same time, the geographic routing protocol running on top of these coordinates finds shorter and shorter routes. While 2D virtual positions converge to a state where routes are a few percent longer than the shortest path, 1D virtual heights converge to optimality.
We have described 1-hopMAC, the 3rule routing protocol and the use of virtual coordinates to self-organize a network. These proposals were presented and evaluated separately. In this chapter, we present how these protocols can be linked together to form an energy-efficient self-organizing communication stack.

We present experimental results. Section 6.2 covers the WiFly experiment, in which a 16-node network is interrogated by a fast moving mobile sink mounted on a radio-controlled airplane. The goal of this experiment is to show the virtual coordinate approach is successful even under the extreme case of a fast moving mobile sink. Section 6.3 presents the Sense&Sensitivity experiment, which mimics the behavior of an urban WSN. 86 nodes are deployed for 2 weeks and measure temperature and light levels. During the lifetime of the network, nodes and sink nodes are added and removed to show the robustness of the virtual coordinate self-organization scheme.

6.1 Interaction Between MAC and Routing Layers

Although presented separately, 1-hopMAC, 3rule routing and the virtual coordinate scheme were designed to function together. 1-hopMAC is in charge of the solution’s energy-efficiency (it is based on preamble sampling) and it performs neighborhood discovery on-demand. During neighborhood discovery, virtual coordinates are exchanged. The updating process of the virtual coordinates is done after all ACK messages are received by the MAC layer. After the updating process of the virtual coordinates, the routing protocol elects the next hop node. This information is then put into the MAC header, and the message is transmitted.

Depending on the usage scenario, either 2D virtual positions (as in the WiFly experiment) or 1D virtual heights (as in the Sense&Sensitivity experiment) can be used. The resulting protocol stacks are presented in Sections 6.2 and 6.3, respectively. We want to stress the following cross-layer issues when combining 1-hopMAC, 3rule routing and virtual coordinates.

Energy efficiency. The interaction between 1-hopMAC and the 3rule routing protocol is energy-efficient in that:

- Neighborhood discovery is done on-demand, and hence does not need to be done periodically. This significantly lowers the node’s energy-consumption especially in low-throughput scenarios;

- Neighborhood discovery through a dedicated mechanism at the layer rather than through generic messages sent by the routing layer.

Delay. Preamble-sampling is used as a basis for 1-hopMAC. With a preamble lasting for about 100ms, this means each hop takes at least 100ms. For a typical urban application where a message travels over 5 hops or so, this means minimum end-to-end latency is in the range of 500ms.
6.2 The WiFly Experiment

Rolling out a solution is often considered more part of engineering than research. We argue that limited experiments are essential as it confronts the solution with important real-world constraints. In the networking community, most solutions are evaluated by simulation. This is a good approach in that it helps the researcher capture unforeseen emergent behavior of a large-scale network. Yet, the complete behavior of a wireless network can never be entirely captured and reproduced by simulated models. This is particularly true for the wireless propagation model.

Experiments were largely simplified since wireless sensors have become commercially available [Polastre et al., 2005], and tiny operating systems were developed to run on these platforms [Gay et al., 2007]. As expected, important lessons were learnt by confronting protocols to the real world. Whereas [Szewczyk et al., 2004] and [Langendoen et al., 2006] detail the logistics problems faced by real-world experiments, [Kim et al., 2005] presents results about experimenting with geographic routing protocols such as Greedy-Face-Greedy (GFG) or Greedy Perimeter Stateless Routing (GPSR).

Companies are starting to make money out of WSNs. Coronis Systems, a company based in southern France, is selling motes to be attached to home meters (water, electricity, etc.). This company reports a city-wide 25,000 node network deployment in France. Although their solution is functional, the routing paths need to be set up manually during deployment [Dugas, 2005]. ArchRock and Dust Networks have both been founded by faculty from UC Berkeley. ArchRock commercializes a home-automation solution. The sink node embeds a web server for remote data access; the main objective of their work being ease of use. Dust Network builds WSN-based solutions for the industrial world (main customers are production plants). Their vision is quite different, and aims at building highly reliable solutions.

Experiments have been carried out and commercial products exist which involve routing solutions based on periodic exchange of Hello packets, or paths set up manually during deployment. These solutions are functional, but largely sub-optimal in terms of network lifetime and robustness. Recent experiments have shown that more challenging routing proposals are still at a lab stage, and do not function well when confronted to real-world constraints. More generally, there is a need to experiment with new routing paradigms for low-throughput, low-energy WSNs. We try to answer this need by the experiments with the virtual coordinate routing scheme.

The objective the WiFly experiment is to answer the following questions: Is using virtual coordinates a valid approach to routing in WSNs?, and What are the key limiting factors when dealing with such experiments?

To this end, the main contributions the WiFly experiment are:

- We show the validity of the communication architecture presented in Table 6.1 by analysis and simulation.

- We present results of a proof-of-concept experiment in which we confront our architecture with the real-world, under the extreme case of a fast moving mobile sink.

6.2.1 Overview of the Experiment

We want to confront our communication architecture to the extreme case of a fast moving mobile sink. Having a mobile sink is somewhat artificial, and we need to point out that our routing solution is optimal when the connectivity graph is static. By having a fast moving mobile sink, our aim is to show that our solution is robust.

We mount the sink node on a radio-controlled (R/C) plane. We show the setup of the experiment in Fig. 6.1 (a). A WSN composed of 16 nodes is deployed in a field, on one side of the
Figure 6.1: (a) Experimental setup (bird’s view). (b) Maximum distance over which the MS can travel while connected to the BS or a network node.

landing strip, a base station BS is placed on the other side. Note that the BS is completely disconnected from the WSN. The BS issues a request for the WSN to answer (e.g. What are the nodes connected to node 5?). As it is disconnected from the WSN, a mobile sink MS mounted on an R/C plane physically carries this request from the BS to the WSN, retrieves the answer, and physically brings the answer back from the WSN to the BS. The BS can then issue a new request. The MS circles around, continuously carrying a request to the WSN, and an answer to the BS.

For our experiment to operate, we identify six different communication phases:

(1) the BS sends its request to the MS;

(2) the MS sends the request to the first node of the WSN it encounters (this first node changes depending on the MS’s trajectory);

(3) the request is broadcasted over the whole WSN;

(4) the interrogated (source) node sends the answer to the MS in multi-hop mode;

(5) the MS retrieves the message;

(6) the MS delivers the message to the BS.

Each of these phases requires a specific protocol. We only focus on phase 4 which uses the protocol stack detailed in Section 6.2.2 as we are really interested in how the WSN delivers a message from the source node to the MS. The protocols involved in the other phases are of minor importance, and they follow intuitive design. We assume they are robust and function correctly. Details of these protocols are presented in [Watteyne et al., 2008].
CHAPTER 6. CROSS-LAYER INTEGRATION AND EXPERIMENTAL STUDIES

Table 6.1: Communication stack used for the WiFly experiment

<table>
<thead>
<tr>
<th>Application</th>
<th>Mobile sink node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing</td>
<td>3-rule routing</td>
</tr>
<tr>
<td></td>
<td>2D virtual positions</td>
</tr>
<tr>
<td>Medium access control</td>
<td>1-hopMAC_v1a</td>
</tr>
<tr>
<td>Physical layer</td>
<td>18 EM2420 modules</td>
</tr>
</tbody>
</table>

Figure 6.2: The chronograph of 1-hopMAC_v1 represented together with the tasks of the routing layer. The shaded red areas represent the exchange of the virtual positions. The exact tasks of the routing layer are cut into periods T1, T2, T3 and T4, explained in Section 6.2.2.

6.2.2 1-hopMAC_v1 and 2D Virtual Positions

2D virtual positions are used in the WiFly experiment. In this scheme, a node updates its virtual position by averaging out its neighbors’; it thus needs to acquire the list of all its neighbors. 1-hopMAC_v2 was in a too early design stage at the time, so we used 1-hopMAC_v1a to perform neighborhood discovery. We asked nodes to send their ACK messages after a random backoff to maximize the probability of detecting all neighbors. The resulting communication stack is presented in Table 6.1.

To show the interaction between protocol layers, we consider node S wants to send or relay a message. Fig. 6.2 shows the execution of the 1-hopMAC_v1a protocol, together with the tasks of the routing layer.

- **T1.** Node S starts by sending a series of microframes. At the same time, it extracts the sequence of already traversed nodes from the message is has to relay, and populates the sent_to and received_from lists used by the 3rule routing protocol described in Section 5.3.2. It also creates the empty neighbors list which will be filled as ACK messages
are received.

- **T2.** Node $S$ receives $ACK$ messages, each containing the identifier and virtual positions of the neighbors. This information is stored in the $neighbors$ list as they arrive.

- **T3.** Node $S$'s virtual position is updated by the average of its neighbors'. This updated virtual position is written in the header of the $DATA$ message. The 3rule routing protocol is applied, which uses the $sent_to$, $received_from$ and $neighbors$ lists to identify the next hop. The identifier of the resulting next hop is written in the header of the the $DATA$ message. The $DATA$ message is sent.

- **T4.** All the neighbor nodes listen to the header of the $DATA$ message, thus receive $S$' new virtual position and the next hop identifier. All nodes check whether their virtual position is not closer to node $S$ than $MinVirtual$. If this is the case, they update their own virtual position using (5.8). All nodes, except the next hop, switch off their radios after receiving the header.

### 6.2.3 Preliminary Analytical and Simulation Results

**Estimating the maximum speed of the $MS$.** We perform some simple analysis to determine the maximum speed of the $MS$. These calculations were done before starting to simulate or implement the protocols on real nodes, to decide whether the implementation would be at all realistic.

The critical parameter when considering real-time in our setting is the speed of the $MS$. After having received a request, the WSN needs to broadcast this request and return the answer before the $MS$ leaves the network. Based on the protocols involved, we calculate the maximum speed of the $MS$, and see whether the obtained value is realistic, considering a typical R/C plane flies at about 30km/h. As depicted in Fig. 6.1 (b), the $MS$ stays within communication range of a node for about 50m of its trajectory (preliminary measurements have shown that with a transmission power of -25dBm we have a communication range of about 25m).

Based on the values of the different packets length and the protocols involved (see [Watteyne et al., 2008] for details), we calculate that the $MS$ could go as fast as 500km/h and still be able to communicate with the $BS$, and 140km/h when communicating with the WSN. These values are in line with the speed of a typical R/C plane.

**Collision probability at MAC level.** We present simulation and analytical results obtained for the MAC protocol. These results are compared to the experimental results in Section 6.2.4.

In the implemented version of 1-hopMAC$_{v1a}$, nodes send out their $ACK$ message after having waited for a random time. If these times are too close to one another, $ACK$ messages collide. The resulting collision probability is given in (6.1) (explained in Section 4.4.2). We implement this mechanism in the home-made simulator and plot averaged-out simulation results together with the analytical results in Fig. 6.3 (how experimental data is obtained is described in Section 6.2.4).

\[
P = 1 - \left( \frac{CW - D_{ACK}}{CW} \right)^N.
\]  

(6.1)

**Simulated hop count and miss ratio.** In our simulation, 25 nodes are regularly placed on a 5×5 grid\(^1\), two neighbor nodes separated by 200m. The $MS$ traverses the network once, following a straight line, as it does in the real experiment. We call "edge" the case where the $MS$ follows

\(^1\)This constitutes a worse case than the experiment which only involves 16 nodes.
Figure 6.3: (a) The setup used to measure the experimental collision probability (see Section 6.2.4). (b) Comparing the experimental collision probability with theoretical and simulation results. Presented simulation results are averaged out over $10^6$ runs; experimental results over 8000 runs.
the outer edge of a square and "diagonal" when it crosses the area diagonally. These two models represent the shortest and longest duration the MS is connected to the network, respectively.

As the MS is connected to the network for only a short duration, miss ratio can be higher than zero. We plot the miss ratio as a function of the MS’s speed in Fig. 6.4 (a). Note that for the speed range of our R/C plane (20-30km/h), miss ratio stays zero. Fig. 6.4 (b) plots the average hop count of only the messages that eventually reach the sink.

6.2.4 Experimental Results

We wanted to run simulations and do some basic analysis before implementing our solution on real nodes for two reasons. First, simple analysis could warn us if the experiment was unfeasible (e.g. the maximum speed of the MS turning out to be extremely low). Second, simulation gives us a rough estimate of the behavior of our protocol, and allows us to fine-tune parameters.

Parameters and hardware. We use 18 Ember EM2420 nodes, each equipped with a CC2420 radio chip and an AtMega128 micro-controller (see description and picture in Section 3.3.3). Programming is done using the Think component-based language [Ozcan et al., 2005] which turns out to be very user friendly and efficient. Programming the nodes has been a rather straightforward experience. 16 nodes are mounted into little boxes, and placed on top of 1m high poles which are stuck into the field. From that height, and with transmission power set to -25dBm, each node has a maximum communication range of 10-25m. One node plays the role of BS. It is connected to its development board from which it displays lines of text on a laptop’s screen through a serial cable. This enables us to monitor the activity of the BS node in real-time. The MS node is no different from the others, only it is wrapped into protective material before being mounted onto a R/C plane.

Energy consumption of the 1hopMAC protocol. To follow the execution of an implemented protocol on a node, an easy way is to use an oscilloscope to display the power used by a node as a function of time. The measured values are presented in Fig. 6.5 (b). This figure gives us useful information:

- the radio module has a major impact on the total energy budget of a node;
- sending, receiving and idle listening consume approximately the same amount of power;
- we verify our implementation by noting that the ACK messages are sent at different instants depending on the value of the node’s virtual distance to the sink.

By measuring the current values, we extract the energy consumption of the different radio states, and the energy consumed by the different phases in the 1-hopMAC protocol (Table 6.2). \(E_{Tx}\) refers to the energy needed by a node to send a message. The receiver of this message spends \(E_{Rx}\) while any other neighbor (not the intended destination) spends \(E_{comp}\). The neighbor which is not the intended receiver does not need to receive the data. We see that sending a message costs about 2-3 times more energy than receiving one. This has a major impact on upper-layer protocol design as having a dense network does not jeopardize energy-efficiency.

Collision probability at MAC level. We verify the theoretical and simulation-based values for collision probability at MAC level by experimentation. We therefore adopt the setting depicted in Fig. 6.3 (a). A host computer is linked to 6 nodes through wired links. Through these wired links, it sends randomly and uniformly chosen virtual distance values to each of the five neighbor
Figure 6.4: Miss ratio (a) and hop count (b) when the MS traversed the deployment area following a straight line. These simulation results were averaged over $10^4$ runs.
Figure 6.5: (a) explaining the 1-hopMAC protocol. The horizontal axis represents time. A rectangle above (resp. under) the time axis means that the node’s radio is transmitting (resp. receiving). When there is no rectangle, the radio is off. (b) Power consumption versus time during Phase 4 with two communicating nodes. Measurements for a transmitter and the receiver are presented in the upper and lower parts, resp. The experiment is repeated for a virtual distance to the sink equal to 0, 3 and 5, hence the three groups of figures. Note that the first part of the preamble is truncated to ease readability.

<table>
<thead>
<tr>
<th></th>
<th>Consumption (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{\text{sleep}}$</td>
<td>8.02 mW</td>
</tr>
<tr>
<td>$P_{\text{idle listen}}$</td>
<td>65.83 mW</td>
</tr>
<tr>
<td>$P_{\text{Tx}}$</td>
<td>66.16 mW</td>
</tr>
<tr>
<td>$P_{\text{Rx}}$</td>
<td>70.69 mW</td>
</tr>
<tr>
<td>$E_{\text{Tx}}$</td>
<td>3.50 mJ</td>
</tr>
<tr>
<td>$E_{\text{comp}}$</td>
<td>1.55 mJ</td>
</tr>
<tr>
<td>$E_{\text{Rx}}$</td>
<td>1.80 mJ</td>
</tr>
</tbody>
</table>

Table 6.2: Consumption of the individual radio states and 1-hopMAC phases (transmission power set to 0dBm)
CHAPTER 6. CROSS-LAYER INTEGRATION AND EXPERIMENTAL STUDIES

nodes of \( S \), and asks \( S \) to send out a request. It then monitors which neighbor \( S \) elects as the node \( S \) thinks has the smallest virtual distance. The communicated virtual distances are used by the neighbor nodes to determine the backoff time before sending the acknowledgment. If \( S \) does not choose the node with smallest virtual distance, it means the acknowledgment message coming from this node has collided with another acknowledgment message. We average the successes over 8000 runs and obtain the experimental collision probability depicted in Fig. 6.3 (b).

Proof-of-concept experiment with virtual positions. The experimentation was carried out during summer 2007 at Alpe d’Huez, a ski resort in the French Alps. The experiment ran smoothly, and the laptop connected to the base station recorded the information contained in the answer messages transmitted by the \( MS \). We asked the requested node to send the list of its neighbors (see Fig. 6.6). This showed that, due to random propagation conditions, electromagnetically (signal strength) close neighbors are not necessarily geographically close.

Due to the limited availability of the R/C plane, we could not collect sufficient data on the routing protocol to be plotted against the simulated values. Also, and most surprisingly, the element which caused some transmissions to fail was not the use of an exotic virtual coordinate-based routing protocol, but the collisions at the MAC layer. We had set the contention window size to 30\( ms \), yielding a collision probability of about 8\% according to Fig. 6.3 (b). This value turns out to be too high, especially with paths of 4-5 hops (in this case probability of success drops to \((1 - 0.08)^5 = 66\%\)). For this reason, we developed the 1-hopMAC\(_{v2}\) protocol for the Sense&Sensitivity experiment presented in the next section.

We have to agree with [Langendoen et al., 2006] that Murphy loves WSN experiments. After a few hours of flight and data collection, the R/C plane got out of control and crashed, putting an end to our experiment. This crash was, however, not related in any way to the wireless nodes, but to a remote-control failure. Despite its sudden end, this experiment has fulfilled its goals, namely:

- we have implemented a complete energy-efficient self-organizing communication architecture;
- we have shown by analysis, simulation and experimentation that it is functional and efficient;
• we have proven that virtual coordinates are a valid routing option, which function even under the extreme case of a fast moving mobile sink.

6.2.5 Lessons Learnt

From a purely research point of view, and as stated in the previous section, this experiment serves as a proof-of-concept experiment for routing using virtual positions. Successfully passing the test of an experiment under the extreme case of a fast moving sink node is, we believe, very valuable. Because of its robustness, routing using virtual positions can be applied in many scenarios. Most importantly, it gives the user an "install and forget" experience, which is very different from existing solutions.

From the standpoint of leading an experiment, a lot of lessons were learnt. Of course, a good step by step preparation is essential and helps pushing the moment Murphy’s law comes into play further away. Another very valuable lesson is that an experiment is not the ultimate step. In particular, the results it produces are not "better" or "more valuable" than the ones produced by analysis and simulation, as long as one knows what assumptions one makes. This is the case for example for routing protocol efficiency which can simply not be assessed by experimentation as (1) during the experiment parameters can change and (2) good results come from many runs (e.g. $10^6$) for which an experiment is simply too slow. As a consequence, analysis, simulation and experimentation should be considered complementary.

From a WSN point of view, this experiment has shown once more the major importance of the MAC protocol. With low-power nodes and lossy links, the intelligence needs to be shifted from upper layers down to the MAC layer. This trend is somewhat different from traditional (wired) networking. We believe this is one of the keys to deeply understand communication in WSNs.

The extremely dynamic nature of the experiment prevented us from extracting scientifically valuable data related to the routing layer. The Sense&Sensitivity experiment includes easy-to-use hardware and software, means of collecting debug information and tools to display information in real-time.

6.3 The Sense&Sensitivity Experiment

Although the proof-of-concept WiFly experiment shows the robustness of the virtual coordinate scheme, it does not provide enough scientific results, especially on the routing layer. The Sense&Sensitivity experiment comes to answer this need. Although the protocol stack is slightly modified, the main difference between the two experiments is the application scenario.

Sense&Sensitivity mimics an Urban WSN, as defined in [Dohler et al., 2008c]. 86 static nodes are deployed in a large space, each periodically reporting light and temperature measurements to several sink nodes. Measurements and network status information are collected in a database and presented through a web interface. The aim of this experiment is to study the self-organization characteristics of the communication architecture, and to show its applicability to an Urban WSN environment.

6.3.1 1-hopMAC$^v_2$ and 1D Virtual Heights

The Sense&Sensitivity experiment uses the full potential of 1-hopMAC$^v_2$, 1D virtual heights and 3rule routing. The resulting communication stack is presented in Table 6.3.

We show the interaction between protocol layers by considering node $S$ wants to send or relay a message. Fig. 6.7 shows the execution of the 1-hopMAC$^v_2$ protocol, together with the tasks of node $S'$ routing layer.
Table 6.3: Communication stack used for Sense&Sensitivity

<table>
<thead>
<tr>
<th>Application</th>
<th>Urban WSN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Routing</strong></td>
<td>3rule routing</td>
</tr>
<tr>
<td></td>
<td>1D virtual heights</td>
</tr>
<tr>
<td><strong>Medium access control</strong></td>
<td>1-hopMAC(_{v/2})</td>
</tr>
<tr>
<td><strong>Physical layer</strong></td>
<td>86 WSN430 modules</td>
</tr>
</tbody>
</table>

Figure 6.7: The chronograph of 1-hopMAC\(_{v/2}\) represented together with the tasks of node S’s routing layer. The shaded red areas represent the exchange of the virtual heights. The exact tasks of the routing layer are cut into periods T1, T2 and T3, explained in Section 6.3.1.
• **T1.** Node $S$ starts by sending a series of microframes. At the same time, it extracts the sequence of already traversed nodes from the message it has to relay, and populates the $sent_to$ and $received_from$ lists used by the 3rule routing protocol described in Section 5.3.2. It also creates the empty $neighbors$ list which will be filled as $ACK$ messages are received.

• **T2.** Node $S$ receives $ACK$ messages, each containing the identifier and virtual heights of the neighbors. This information is stored in the $neighbors$ list as they arrive.

• **T3.** Node $S$’ virtual height is updated by the minimum of its neighbors’ incremented by one. The 3rule routing protocol is applied, which uses the $sent_to$, $received_from$ and $neighbors$ lists to identify the next hop. The identifier of the resulting next hop is written in the header of the $DATA$ message. The $DATA$ message is sent.

Note that there is no $MinVirtual$ constraint when using 1D virtual heights. Node $S$ does not need to send its updated virtual height to its neighbor, which do not have to check whether they are virtually too close (i.e. there is no $T4$ period as in the WiFly communication stack).

### 6.3.2 Overview of the Experiment

This section details the hardware and system architecture as well as the implementation parameters and packet formats. Results are presented in Section 6.3.3.

**Hardware.** 86 WSN430 nodes are used (see description and picture in Section 3.3.3). These nodes contain a MSP430 microcontroller and a CC1100 radio chip communicating on the 868MHz frequency band.

The MSP430F1611 [MSP430, 2007] microcontroller is one of the most powerful in the MSP430 family. Based on a 16bit RISC (Reduced Instruction Set Computer) architecture, it embeds 48KB of flash memory and 10KB of RAM. Five low power modes enable the programmer to control exactly which modules should stay on, fine-tuning the energy consumption. A Direct Memory Access (DMA) module is able to read/write to/from flash directly into RAM memory without the help of the Central Processing Unit (CPU). A SPI bus interconnects the MSP430 with the CC1100 radio chip.

The CC1100 [CC1100, 2008] radio chip is set to communicate in the 800-928 MHz ISM band. It can be tuned to transmit at powers up to $+10$dBm, at a data rate between 1.2 and 500 Kbit/s, and using one of 5 modulation schemes (2-FSK, 2-GFSK, 2-MSK, OOK and 2-ASK). Other features which are extremely interesting to keep the energy budget low include the capability of verifying a packet’s integrity through its Cyclic Redundancy Check (CRC) and the Wake-On-Radio (WOR) functionality. When WOR is activated, the radio chip periodically samples the channel and sends an interrupt to wake up the microcontroller when it hears a preamble. This allows the microcontroller to sleep when the network sits idle as preamble sampling is performed by the radio chip.

Additional components are mounted on WSN430 board, such as 3 Light Emitting Diodes (LED), two light sensors (visible and infrared) and a temperature sensor. The board is powered by a rechargeable 4V 800mAh Lithium-Polymer battery.

**Implementation details.** Tables 6.4, 6.5 and 6.6 show the format of the frames, packets and messages generated by the MAC, routing and application layers, respectively. The length of a complete data packet (including all headers) in bytes is hence $27 + 2 \cdot h + 3 \cdot n$, where $h$ is the number of hops it has traveled through, and $n$ the number of neighbors of the source node. In typical applications, this yields lengths of data packets between 32 and 77 bytes.
CHAPTER 6. CROSS-LAYER INTEGRATION AND EXPERIMENTAL STUDIES

The first byte of the 1-hopMAC\textsubscript{v2} frames is used to differentiate the different frame types. In the newCW packet format, "CW Length" is a parameter which sets the length of the contention window which follows. In practice, this is always 11ms, as shown in Table 6.7. \(X_0\) and \(X_1\) refer to the virtual coordinates of a node. Note that we are using 1D virtual heights, so only \(X_0\) is used by default. Counter is a number set to \(86^2\) for the first micro-frame, and which is decremented at each new micro-frame in the preamble. This is used by the receiver to know when the preamble ends (see description of 1-hopMAC in Chapter 4).

The field "Destination" in the routing layer packet is used to identify the destination sink node. This is possible if different sinks use a different set of 1D virtual heights, as shown in Fig. 5.20. If the connectivity graph is very dynamic, the 3rule routing protocol may end up not finding a path to destination. In this case, the sequence of traversed nodes is emptied at the current node and the message is resent. We allow up to 3 resets for a given message. The field \#resets is set to 3 and decremented when a reset happens. When it hits 0, the message is dropped.

The application header is filled at the source node and is left unchanged until the message reaches the sink. The information it contains is used for accounting purposes only. In particular, the virtual heights \(X_0\) and \(X_1\) of the source node are put in the application header for debug purposes only. \#CWs is the number of contention windows used by 1-hopMAC\textsubscript{v2} at the source node; \#neighbors is the number of discovered neighbors.

Table 6.7 details the parameters used in the Sense&Sensitivity experiment. In particular, the contention window size was set to minimize the collision probability as detailed in Section 4.5.2.

\footnote{The fact this value is also the number of nodes in the network is pure coincidence.}
CHAPTER 6. CROSS-LAYER INTEGRATION AND EXPERIMENTAL STUDIES

Table 6.5: Routing layer packet format

<table>
<thead>
<tr>
<th>Source address</th>
<th>Seq. Num.</th>
<th>Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Light</td>
<td>IR light</td>
<td>Voltage</td>
</tr>
<tr>
<td>$X_0$</td>
<td>$X_1$</td>
<td>#CWs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>#neighbors</td>
</tr>
</tbody>
</table>

Table 6.6: Application layer message format

The check duration is relatively high, but can be easily tuned to a smaller value. This is considered future optimizations.

System architecture. The sink nodes are connected to a gateway (a customized Linksys NSLU2 Network Storage Link) through a serial cable; the gateway communicates with the data server through the Ethernet LAN (see Fig. 6.8). Communication between the gateway and the server is done using XML formatted data encapsulated in HTTP datagrams. An example XML data is provided in Listing 6.1. It tells the server that message of sequence number 22 from node $ab:12$ has just been received at sink node $8f:3f$. Node $ab:12$ has three neighbors $be:89,12:48$ and $d6:56$. The message has traveled 3 hops ($ab:12 \rightarrow 56:33 \rightarrow 5a:aa \rightarrow 8f:3f$). The XML data also informs the server of the last read sensor values.

6.3.3 Experimental Results

Each node is tuned to generate a message every 7.5 min on average. With 86 nodes deployed, the sink node receives a message on average every 5s. 720 messages are thus collected per hour, 17200 per day. All the data contained in the messages is stored in a database for later inspection.

We chose to present results in a bottom-up approach, starting by the deployment and energy consumption, before going up the communication layers: propagation, MAC, routing and application.

Deployment. The virtual coordinate self-organization scheme enables a true deploy-and-forget experience. Once programmed and switched on, the 86 nodes are put in a box and scattered on the premises of France Telecom R&D in Meylan, France. Although the first batch is deployed
<table>
<thead>
<tr>
<th>Description</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmission power</td>
<td>-5dBm</td>
</tr>
<tr>
<td>Battery capacity</td>
<td>11.6KJ</td>
</tr>
<tr>
<td>Check Interval (preamble sampling)</td>
<td>128ms</td>
</tr>
<tr>
<td>Check duration</td>
<td>2ms</td>
</tr>
<tr>
<td>Micro-frame spacing</td>
<td>1.5ms</td>
</tr>
<tr>
<td>Number of micro-frames</td>
<td>86</td>
</tr>
<tr>
<td>Contention window size</td>
<td>11ms</td>
</tr>
<tr>
<td>Duration of a CCA</td>
<td>400ns</td>
</tr>
<tr>
<td>Duration of an ACK</td>
<td>1.8ms</td>
</tr>
<tr>
<td>Minimum duration between successive generated messages</td>
<td>5min</td>
</tr>
<tr>
<td>Maximum duration between successive generated messages</td>
<td>10min</td>
</tr>
</tbody>
</table>

Table 6.7: Parameters used for the Sense&Sensitivity experiment.

Figure 6.8: System architecture. A server collects all the data in a database for later analysis.
randomly, extra nodes have been added afterward to strengthen the links between buildings. Out of the 86, 68 are attached to the indoor ceilings and 18 are bagged and mounted on 1m high poles outdoors. Fig. 6.9 pinpoints their location over an area of roughly 140 by 250 meters. Similarly, once the sink nodes are up and running, they can be placed – together with their gateway – anywhere where there is wired LAN connectivity (to send the received data to the database). Adding or removing sensor nodes and sink nodes does not require any configuration. The relocation of a sink node is hence a matter of unplugging and replugging the gateway.

Energy consumption. By connecting an oscilloscope onto a resistor mounted at the battery, it is possible to visualize the power consumption as a function of time. This is the power consumption for the whole board, i.e. the sum of the power consumption of its individual components.

Fig. 6.10 (a) zooms on the instant when an idle node samples the channel. With its radio off, the node consumes $150\mu A$. Before sampling the channel for $2ms$, the node needs $1.2ms$ to start and calibrate the radio. One such samples costs about $0.15mJ$. As this happens every $128ms$, as long as the network sits idle, the node consumes about $1mJ/s$, i.e. $1mW$.

Fig. 6.10 (b) focuses on the node receiving data. As annotated on the screenshot, the node listens to a micro-frame and is able to send its $ACK$ message in the first contention window. It stays asleep for the second contention window and receives the data. A node consumes about $1mJ$ to receive a packet.

Fig. 6.11 focuses on the transmitting node. Fig. 6.11 (a), after performing a $CCA$, it sends a series of microframes, one every $1.5ms$. Fig. 6.11 (b) shifts to the period after the microframes, and shows the sequence of contention windows, received $ACK$ messages and sent $DATA$ messages. Note that the transmitting node’s radio is always active between the first micro-frame and the end of the data transmission. A node consumes about $9.5mJ$ to receive a packet.

If we imagine a node sends a message every hour, the network has an expected lifetime of about 4 months. This value is rather low compared to commercial solutions. Increasing lifetime involves lowering the time it takes to sample the channel, a subject for future optimizations.

---

1By looking at the data sheets of the individual components in the board, one can see that this value can be brought down to about $1.5\mu A$ ($900nA$ for the CC1100 and $600nA$ for the MSP430 is Low Power Mode 3 – LPM3). Such optimization is left for future work.
Figure 6.9: The locations of the 86 wireless nodes mounted on the ceiling of the different building at France Telecom R&D.

(a) Periodic channel sampling (this happens every 128ms).

(b) Receiving a preamble, sending an ACK and receiving the DATA.

Figure 6.10: Energy consumption of a receiving node over time (as read off the oscilloscope).
RSSI vs distance. Complex computer assisted propagation models can predict the reception power at a given location in a specific setting. This precision comes at a high price in computational power and time, which makes this approach not suitable for network simulation. In the next paragraphs, we observe the behavior of the propagation medium and extract a simple propagation model which can easily be used for simulation.

For each packet exchanged in the WSN, we measure and store its RSS value. We plot the RSS value as a function of distance between nodes in Fig. 6.12. Experimental data points are not aligned because of shadowing. Shadowing causes RSS values to follow a Gaussian distribution when RSS is in dB (or a lognormal distribution when RSS is in mW). For easier manipulation, we simplify this observation by considering RSS values are uniformly distributed in a range of 40dB below the Friis model.

Link stability. Too see how stable a link is over time (i.e. how stable reception power over that link is), we plot Fig.6.13 (a) which shows the variation of the Received Signal Strength (RSS) value over time. The three links depicted are representative of the 120 bidirectional links in the network. This is shown in Fig. 6.13 (b) which plots the standard deviation of the RSS reading in dBm as a function of this RSS value. It shows that, independently from the RSS value, deviation is small, average deviation is 1.04dBm. The link is stable over time. This means that once a link is established, it stays stable over time.

Note that, 68 out of 86 nodes were attached to the ceiling of the different buildings. One could suppose that this location makes the wireless links less vulnerable to people walking around, i.e. had they be located lower, standard deviation of RSS values could have been higher.

Link probability. The RSS over a link is an indirect measurement of its quality, a direct measurement is the associated Packet Error Rate (PER), the percentage of packets lost when sent over that link. A perfect link has a PER equal to 0; a non-existing link a PER of 1. We call link probability 1-PER. We plot link probability as a function of its RSS in Fig. 6.14. The experimental data (crosses) verifies the theoretical model presented in Section 3.2.2 and Fig. 3.2.

A realistic propagation model. Complex propagation models can predict precisely the reception signal strength at the cost of computation power and time. For simulation, where a large number of runs need be performed for results to be worthwhile, this is not practical. Based on the
CHAPTER 6. CROSS-LAYER INTEGRATION AND EXPERIMENTAL STUDIES

Figure 6.12: Relationship between RSS and distance obtained by experiment in an indoor environment (see Section 6.3 for details).

Figure 6.13: Once a link is established, its RSS value is stable over long periods of time.
observations on RSS vs. distance, link stability and link probability, we can extract a simple yet realistic propagation model. This can easily be implemented and used for simulation.

Links appear and disappear over time. The dynamic nature of a link can be captured by modeling graph edges as probabilistic, i.e. only a fraction of the packets transmitted by the node on one side of that edge are successfully received by the node on the other end. By using the PER vs. RSS model in Fig. 6.14, it is possible to assign a probability to each edge. Because the RSS of a link is stable over time (see Fig. 6.13), the RSS value of a link needs only to be calculated once when simulation starts. The RSS is determined as a function of distance between nodes, by picking an RSS value randomly and uniformly distributed between the Friis model and 40dB below. The resulting graph is depicted in Fig. 6.15; link probability is color-coded.

To sum up, a weighted graph is obtained as follows. First, a number of nodes are scattered in the simulated area. Then, for each pair of links, an RSS value is determined. Finally, the RSS value is converted into a link probability.

**Neighborhood evolution.** The RSS value is stable over time, and so is the link probability. As an example, a link with a PER of 0.9 stays a bad link for the whole duration of experiment, with only 10% of sent packets received by the neighbor node. This means that neighborhoods change over time, as some neighbors are discovered by the neighborhood discovery protocol only once every 10 neighborhood discoveries. This is confirmed by the experimental data plotted in Fig. 6.16, which shows the evolution of the number of neighbors of three representative nodes over a time span of 18 hours. In some cases, the number of neighbors goes from 4 to 10 neighbors from one neighborhood discovery to the next.

Neighborhood evolution is a natural consequence of the propagation model described previously, and is often neglected when designing communication protocols. Although it is possible to take into account only those neighbors which have a high RSS value (thus a low PER) to ensure graph stability, we argue that this may artificially remove links with ensure network connectivity.
Figure 6.15: A probabilistic graph representation of a WSN, generated using the realistic propagation model extracted from experimental data. This graph is quick to generate and is hence very suitable for simulation.

Figure 6.16: The number of neighbors of the nodes evolves significantly over time. These 3 nodes are representative for the whole network.
Figure 6.17: In 1-hopMAC$_{v2}$, the number of contention windows needed increases with the number of neighbors.

As we describe later on, during our experiment, several groups of nodes are connected to the network through a small number of weak links. By filtering our weak links, we would disconnect this group. Depending on the application, this may not be an option.

**MAC layer.** 1-hopMAC$_{v2}$ opens up new contention windows as long as neighbors are answering. The more neighbors, the more contention windows are needed to accommodate their ACK messages. This is shown in Fig. 6.17. Note that 2 contention windows are needed at least, as the last window is always empty.

**Connectivity graph.** The web interface used for the Sense&Sensitivity experiment presents the connectivity graph of the network in real-time. A snapshot of this graph is presented in Fig. 6.18. Each node has on average 4.8 neighbors. France Telecom R&D in Grenoble is organized as a number of interconnected square buildings. Outside walls of these buildings are entirely covered by windows which are coated with a thin layer of metal to reflect sunlight. Electromagnetic signals have a very hard time traversing this coating, which is the reason why the majority of the links leaving a building do so through the junctions.

The 7 nodes placed on the Western-most part are connected to the network by few very poor links. Although only a subset of the sent packets are eventually received by the sink nodes, if the MAC protocol had filtered out weak links, these nodes would have been completely disconnected.

**Average hop count and virtual heights.** The routing protocol needs to find a path on the connectivity graph between each node and one of the sink nodes. By default, we deploy 2 sink nodes which share a single 1D virtual height. This means that a node sends its data implicitly to the topologically closest sink node. Fig. 6.19 shows the spatial evolution of the virtual heights of the nodes by contour lines. From the virtual height values of the individual nodes, a plotting algorithm draws lines where the virtual height is constant (much like slopes are indicated on a hiking map). The closer the lines, the steeper the slope created by the virtual heights.
Figure 6.18: Snapshot of the connectivity graph on Wednesday 27 August 2008 at 09:31:42. Circles represent nodes; the 2 red squares represent sink nodes. The number written are the identifiers of the nodes; the size of the nodes represent their current battery voltage. Orange lines interconnect neighbor nodes, the stronger those lines, the stronger the RSSI. Links are unidirectional when the orange line stops halfway between two nodes. The green line represents the path taken by the last packet (lower right).
Fig. 6.19 shows how the values of the virtual height increase as we move away from either of the two sink nodes (represented by black squares). From this reading, we can see that data from nodes on the Northern part flows to the Northern sink, the rest to the Southern sink. As there are more nodes in the Northern part, the Northern sink receives more traffic. In the 18 last hours of the experiment, 18895 messages were received, 13823 (73.2%) by the Northern sink, 5072 (26.8%) by the Southern sink.

The sinks receive on average one message every 5s. As these messages come from different nodes, they travel different number of hops. We therefore average the number of hops over a 2min sliding window and plot the result in Fig. 6.20. The average hop count is 5.35 with two sink nodes.

**Removing the second sink node.** The virtual coordinate-based self-organization scheme enables sink nodes to be added and removed without configuration. When a sink is removed, the virtual heights rearrange around the remaining sink nodes. Removing a sink node necessarily increases the average path length, as nodes which previously reported to the removed sink now need to send their messages to another, further away, sink.

Fig. 6.21 shows how the average hop count changes as a sink is removed. Hop count jumps from an average 5.35 to 9.04 because the sink is, on average, further away. The figure also shows that the network self-organizes around the remaining sink node in a matter of minutes.

Fig. 6.22 shows how the virtual heights have reorganized around the remaining sink node. Note that the maximum virtual height is now 13.
Figure 6.20: Hop count averaged over a 2min sliding window with two sink nodes sharing the same 1D virtual height (see Fig. 6.19). The average hop count is 5.35.

Figure 6.21: When the second sink is removed, the network organizes around the remaining sink. Hop count jumps from an average 5.35 to 9.04 because the sink is, on average, further away.
Figure 6.22: Contour lines show the virtual heights when nodes reorganize around the remaining sink node represented by a black square.
Adding a sink node on a new set of virtual heights. Comparing Fig. 6.23 with Fig. 6.22 shows the importance of the sink location. Both show the contour lines created by the virtual heights when a single sink is used in the network, yet Fig. 6.23 shows a maximum virtual height of 9 which is lower than 13 in Fig. 6.22.

Fig. 6.23 was obtained by adding a third sink node on a second, independent, 1D virtual height (transmitted in the so far unused field $X_1$ of 1-hopMACv2 ACK messages, see Fig. 6.4). There are now three sink nodes in the network: two sink nodes we call $A$ and $B$ which share $X_0$, the first 1D virtual height (as depicted in Fig. 6.20), and the sink node we call $C$ on $X_1$, another independent 1D virtual height.

Such a setting (which was already described and shown in Fig. 5.20) can be used for a node to choose to which sink it sends data. If 3rule routing uses virtual height $X_0$, the message goes to either sink node $A$ or $B$. If it uses $X_1$, the message reaches $C$. If a node simply picks the smallest virtual height of the two, its packet rolls down to the sink node among $A$, $B$ or $C$ which is closest. This is the scenario we use; Fig. 6.24 shows that adding a third sink lowers the average hop count from 5.35 to 4.56.

6.3.4 Lessons Learnt

Setting up a WSN testbed can be seen as an output and the ultimate goal of designing a communication architecture. It also serves also as an input as it helps the researcher understand real-world constraints.

The Sense&Sensitivity experiment has proven that the virtual coordinate-based self-organization scheme is a valid solution for an urban WSN setting. It has successfully passed the extreme test.
of adding and removing both nodes and sink nodes. We believe this approach to be attractive because it gives the user a deploy-and-forget experience, while remaining energy-efficient. To our knowledge, this level of flexibility is not offered by other commercial or academic solutions.

This experiment has shown once again the major importance of the MAC protocol in WSNs. From an energy-efficiency point of view, the expected lifetime offered by the current implementation of 1-hopMAC$^v_2$ is somewhat below expectations. We believe it is more an implementation rather than a design issue and we are confident lifetime can be significantly increased by making channel sampling more efficient. From a topological point of view, the data we have collected about propagation has enabled us to come up with a simple and realistic model. Because nodes may have weak links to potential neighbor nodes, the neighborhood discovered by 1-hopMAC$^v_2$ can vary significantly over time. Although the virtual coordinate scheme is robust enough to adapt to these continuous topological changes, things could be improved. Methods such as link hysteresis could be used to stabilize neighborhood. This should be done intelligently to avoid removing weak links which are needed to keep the network connected. This is an important issue which is worthwhile investigating in future work.

### 6.4 Summary

Although 1-hopMAC, 3rule routing and the virtual coordinate-based self-organization scheme were presented separately, they are meant to function together. Because there are two variants on both 1-hopMAC (1-hopMAC$^v_1$, 1-hopMAC$^v_2$) and the virtual coordinate scheme (1D, 2D), those protocols can be glued together in different ways. Regardless of these variations, the resulting communication architecture provides energy-efficiency and self-organization characteristics. Because it is inherently on-demand, this communication architecture is particularly suited for low-power, large scale and low throughput applications such as urban WSNs.

WiFly is the first of the two presented experiments. This proof-of-concept setting proves that the virtual coordinate approach is valid even under the extreme case of a fast moving sink node. The extreme dynamics of the resulting network has not allowed us to extract exploitable data on
routing. It has, however, shown the applicability of 1-hopMAC.

Sense&Validity uses our communication architecture in a radically different setting. This large scale 86-node network was deployed in a set of buildings in order to mimic an urban WSN deployment. The complete log of all the data has generated interesting results in the different communication layers involved. We were able to create a simple yet realistic propagation model which captures the probabilistic nature of the links between nodes while taking into account the stability of their signal strength. Results on the MAC layer have shown the relationship between the number of neighbors and the number of contention windows opened by 1-hopMAC. Perhaps the most interesting result concerns the self-organization capability of the communication architecture. This extremely flexible communication architecture allows nodes to be deployed without any configuration. Furthermore, as topological changes are rapidly reflected in the node’s virtual coordinates, adding, removing and relocating nodes is a matter of seconds and does not need any configuration. This also holds for sink nodes, as nodes implicitly send their data messages to the topologically closest sink node. By adding and removing sink nodes, we were able to witness how the flow of data spreads to the remaining sink nodes without affecting the functionality of the network. We think the deploy-and-forget nature of our communication architecture makes it suitable for real deployments.
Conclusions and Future Work

6.5 Contributions of This Thesis Work

This thesis has focused on energy-efficient self-organization in wireless sensor networks, with a particular focus on MAC and routing protocols, as well as on self-organization schemes.

To be able to communicate globally, a node needs to know its neighborhood. Whereas, traditionally, nodes therefore periodically exchange Hello messages, we have shown that this is not energy-efficient in low-throughput networks. We have proposed the 1-hopMAC protocol which adopts an innovative on-demand approach, while shifting neighborhood discovery down to the medium access control (MAC) layer for energy-efficiency.

The resulting novel 1-hopMAC protocol, when it has a message to send, broadcasts a request to which its neighbors answer. This freshly created neighbor list is shifted up to the routing layer which can take a routing decision. 1-hopMAC is based on preamble sampling. This ingenious MAC technique achieves ultra-low power consumption in low-throughput networks by having nodes sample the channel in an unsynchronized way.

Two versions of 1-hopMAC were proposed. 1-hopMAC\textsubscript{v1} assumes that the node closest to the destination (i.e. the next hop node when using a greedy routing approach) answers first. Based on this assumption, 1-hopMAC\textsubscript{v1} dynamically switches between variants so as to reduce the global energy consumption. By tuning the mapping function, we were able to reduce the collision probability by over 40%, which translates in a significant reduction in energy consumption. 1-hopMAC\textsubscript{v2} retrieves the list of all the neighbors, and is thus usable in a wider spectrum of applications. Whereas 1-hopMAC\textsubscript{v1} optimizes energy-efficiency by tuning the mapping function, 1-hopMAC\textsubscript{v2} optimizes latency by tuning the contention windows length. Successive contention windows are opened to ensure all neighbor nodes can answer.

Preamble-sampling techniques were compared analytically to contention based MAC protocols with common active periods, and frame-based MAC protocols. They showed to outweigh both other techniques in energy consumption for low-throughput networks such as urban WSNs.

Self-organization has long been synonymous to clustering techniques. We have shown that these techniques make little sense in a fully-homogeneous setting because they do not increase throughput, while being complex to implement and energy-hungry in a dynamic network. For these reasons, we have chosen to focus on geographic routing techniques.

We discussed how geographical routing is very applicable to WSNs because of its stateless, distributed and localized nature. Moreover, no structures need to be built or maintained, yielding a simple and energy-efficient approach. We have shown how current solutions use hard-to-meet assumptions, and how they fail when nodes do not know their position with infinite accuracy, or when the unit disk graph assumption is not met.

We have proposed to use the sequence of already traversed nodes to help the routing protocol decide which neighbor should be next on the multi-hop route to destination. This novel method guarantees delivery on any arbitrary stable graph. We have shown that the 3rule routing protocol offers a hop count comparable to existing geographic routing solutions while functioning over a
larger spectrum of topologies.

Because having each node know its position is clearly impractical, we have proposed to use geographical routing over virtual coordinates. These coordinates reflect the node’s location in the topology rather than its position in space. Virtual coordinates are continuously updated in an overhead-free manner as messages flow through the network, providing reactivity and robustness to topological changes. We have proven that virtual coordinates organize in such a way that a geographic routing protocol running on top of them discovers optimal paths in terms of number of hops. This self-organization scheme outweighs all other routing technique. We discussed how this novel virtual coordinate-based self-organization scheme allows nodes or sink nodes to be deleted, removed and relocated without any configuration.

Although presented separately, we described how 1-hopMAC, 3rule routing and the virtual coordinate scheme can be combined, forming an energy-efficient, cross-layered and self-organizing communication architecture.

The proof-of-concept WiFly experiment has shown the validity of the virtual coordinate solution under the extreme case of a fast moving mobile sink node. The Sense&Sensitivity experiment mimicked an urban WSN deployment. This experiment involved 86 nodes deployed in four different buildings and forming a single WSN. After showing how the wireless propagation characteristics cause the neighborhood of the nodes to vary significantly, we have conducted experiments to witness how the data rearranges as sink nodes were added and removed from the network.

### 6.6 Future Work

The thesis work has opened many doors. In this section, we present the issues we believe are important and which we consider are worthwhile subjects for future work.

1-hopMAC\(_{v1}\) assumes that the neighbor node which answers first is the next hop. The study presented in this thesis aimed at reducing the collision probability by tuning the mapping function. This work has since then been extended in [Bettstetter et al., 2008] to the case where time is slotted. The authors find an optimal function which triggers a collision probability lower than the one attained in our initial proposal. We plan to apply these new results to 1-hopMAC\(_{v1}\); simulation and experimental data could be extracted to verify the analytical results.

Most MAC protocol optimizations aim at reducing collision probability, the assumption being that reducing collisions also reduces energy consumption. Some applications, however, are more concerned with timeliness than energy constraints. We believe that real-time communication in WSNs is an important topic which has not received sufficient attention. We also believe that the MAC protocol plays a key role in the timeliness of data transmission. We think it would be interesting to conduct studies on MAC protocols which focus on timeliness rather than energy-efficiency. A second step towards a hard real-time communication architecture is the definition of a routing scheme capable to guaranteeing hard real-time constraints. This topic, we believe, has been largely overlooked so far.

Another Quality-of-Service indicator is reliability. Experimental results have shown that frame-based MAC protocols can achieve high end-to-end reliability. Yet, because of their fully scheduled and frame-based MAC nature, those protocols lack the energy-efficiency and extreme flexibility of preamble-sampling. It would be interesting to look at MAC protocols from an angle different from energy-efficiency. A question we believe is important to answer is What are the QoS limits (latency, reliability, etc.) attainable by energy-efficient medium access techniques such as preamble-sampling?

The virtual coordinate routing scheme is interesting in that it is self-organizing and distributed. Although 2D virtual positions can be used efficiently for routing, they do not reflect the nodes’ real positions. Therefore, they can not be used to location-stamp messages flowing through the
network; something very important in most applications. The 2D virtual position self-organization scheme could be extended in order to obtain hybrid coordinates. These coordinates could be related to the nodes’ real coordinates, while providing an efficient basis for geographic routing protocols.

We see two ways of exploring hybrid coordinates. The first way assumes that nodes already know their real coordinates. Distributed algorithms can be used to detect the nodes which are located around a void area, and to update these hybrid coordinates so that a greedy forwarding technique does not fail. The second way would be to start from virtual coordinates, which can then be modulated using additional ranging information (e.g. obtained by RF ranging techniques). Because this latter approach would require location-aware anchor nodes, it would somewhat resemble relative coordinate routing. The main difference is that hybrid coordinates would be updated continuously through local interaction between neighbor nodes, rather than globally through the periodic creation of gradients.

An important question to answer is which approach is more efficient: having hybrid coordinates which serve at the same time as topological coordinates (for routing) and geographic coordinates (for the application), or having two sets of coordinates which are set up independently.

We believe that WSNs have grown in maturity. Although the market is largely fragmented in a number of proprietary solutions, the ongoing standardization efforts by bodies such as the IETF is a good step forward. Customers are interested by a seamless integration of WSN technology in their existing networks. Therefore, we are particularly interested in the results coming from the IETF 6LoWPAN working group which proposes end-to-end connectivity between the Internet and individual sensors through IPv6 over IEEE802.15.4 integration. We would like to integrate this aspect in future proposals of communication protocols and software implementations.

Many important lessons were learnt in this thesis work from the experimental studies. Hardware and software solutions have become widespread and cheap, making experimental evaluation an easy task. Our point of view is that WSNs have been addressed by different people from different angles. While theoretical results are essential and form a vital basis, we believe that these results should be combined more often with real world experiments.

As an example, evaluating the efficiency of a network-wide emerging behavior such as multi-hop routing protocols is complex when done by experimentation. For experimental results to be acceptable by the research community, best practices, tools and techniques need to be agreed upon.
Appendix
A.1 List of Publications

Patents

Thomas Watteyne, Mischa Dohler, Isabelle Augé-Blum, Stéphane Ubéda "Determining a nodes coordinate in a network of nodes," *patent belongs to Orange Labs*, filed in France on 19 November 2007.


Book Chapters


IETF Contribution


International Journal


International Conferences


Thomas Watteyne, Isabelle Augé-Blum, Mischa Dohler, Dominique Barthel "Geographic Forwarding in Wireless Sensor Networks with Loose Position-Awareness," *Annual IEEE Interna-
tional Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Athens, Greece, 3-7 September 2007.


Tutorials


Research Reports


Posters and Work-in-Progress session


A.2 Abstract [French]

Un capteur sans fils est un petit dispositif électronique capable de mesurer une valeur physique (température, lumière, etc.) de l’environnement et de communiquer par voie hertzienne. Du fait de la taille d’un réseau (pouvant être constitué de centaines de capteurs), la communication avec le nœud collecteur – appelé puits – se fait en mode ad-hoc multi-sauts, puisqu’un capteur ne peut communiquer directement qu’avec un sous-ensemble réduit de capteurs, se trouvant à portée radio.

Les principaux enjeux de ces réseaux sont d’économiser l’énergie (ressource limitée car embarquée dans le capteur) et de définir les protocoles d’auto-organisation permettant aux capteurs de communiquer avec le puits et au réseau de s’adapter aux changements de topologie sans intervention humaine. Concernant la consommation énergétique des capteurs, la puce radio du capteur est la partie la plus gourmande en énergie. Réduire l’utilisation de celle-ci réduit la consommation énergétique du capteur, mais rend plus complexe la communication entre capteurs. Il existe donc un compromis entre efficacité en énergie, et facilité de communication – donc efficacité de l’auto-organisation.

Les contributions majeures de cette thèse sont:

• Nous proposons un protocole d’accès au médium qui évite le maintien des tables de voisinage pour chaque nœud (ce qui consomme de l’énergie). Notre protocole ne construit cette table qu’à la demande, ce qui permet d’être robuste aux changements de topologies (fréquentes à cause des liens radio). Couplée avec des mécanismes d’accès au médium efficaces en énergie de la littérature, cette technique est robuste et permet une réduction importante de la consommation énergétique des capteurs;

• Nous utilisons des coordonnées virtuelles comme base pour l’auto-organisation du réseau de capteurs. Chaque capteur est muni de coordonnées virtuelles. Un protocole de routage est ensuite capable de trouver localement un chemin multi-sauts vers un puits. La mise à jour continue de ces coordonnées garantit la robustesse du réseau face aux changements de l’environnement. Nous montrons que cette technique présente les caractéristiques d’auto-organisation recherchées;

• Nous combinons ces propositions pour former architecture de communication trans-couches. Celle-ci permet d’auto-organiser le réseau de manière efficace en énergie.

Nos résultats se basent sur une étude analytique, par simulation et expérimentations. Ils montrent l’efficacité de nos propositions dans le champ d’application qui nous intéresse. Deux implémentations, regroupant l’ensemble des techniques proposées, sont également présentées.

Mots clés: réseau sans fil ad-hoc multi-sauts, réseau de capteurs, protocole MAC, auto-organisation, efficacité en énergie, étude analytique et expérimentale.
Bibliography


FOLIO ADMINISTRATIF

THESE SOUTENUE DEVANT L'INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE LYON

NOM : WATTEYNE 
Prénoms : Thomas André Jules Rik Marie Jozef


NATURE : Doctorat Numéro d'ordre : 2008-ISAL-0082

Ecole doctorale : Ecole Doctorale Informatique Information et Société (EDIIS)

Spécialité : Télécommunications

Cote B.I.U. - Lyon : T 50/210/19 / et bis CLASSE :

RESUME : Un capteur sans fil est un petit dispositif électronique capable de mesurer une valeur physique (température, lumière, etc.) de l’environnement et de communiquer par voie hertzienne. Du fait de la taille d’un réseau (pouvant être constitué de centaines de capteurs), la communication avec le nœud collecteur – appelé puits – se fait en mode ad-hoc multi-sauts, puisqu’un capteur ne peut communiquer directement qu’avec un sous-ensemble réduit de capteurs, se trouvant à portée radio.

Les principaux enjeux de ces réseaux sont d’économiser l’énergie (ressource limitée car embarquée dans le capteur) et de définir les protocoles d’auto-organisation permettant aux capteurs de communiquer avec le puits et au réseau de s’adapter aux changements de topologie sans intervention humaine. Concernant la consommation énergétique des capteurs, la puissance radio du capteur est la partie la plus gourmande en énergie. Réduire l’utilisation de celle-ci réduit la consommation énergétique du capteur, mais rend plus complexe la communication entre capteurs. Il existe donc un compromis entre efficacité en énergie, et facilité de communication – donc efficacité de l’auto-organisation.

Les contributions majeures de cette thèse sont :
- Nous proposons un protocole d’accès au médium qui évite le maintien des tables de voisinage pour chaque nœud (ce qui consomme de l’énergie). Notre protocole ne construit cette table qu’à la demande, ce qui permet d’être robuste aux changements de topologies (fréquentes à cause des liens radio). Couplée avec des mécanismes d’accès au médium efficaces en énergie de la littérature, cette technique est robuste et permet une réduction importante de la consommation énergétique des capteurs ;
- Nous utilisons des coordonnées virtuelles comme base pour l’auto-organisation du réseau de capteurs. Chaque capteur est muni de coordonnées virtuelles. Un protocole de routage est ensuite capable de trouver localement un chemin multi-sauts vers un puits. La mise à jour continue de ces coordonnées garantit la robustesse du réseau face aux changements de l’environnement. Nous montrons que cette technique présente les caractéristiques d’auto-organisation recherchées ;
- Nous combinons ces propositions pour former architecture de communication transcouches. Celle-ci permet d’auto-organiser le réseau de manière efficace en énergie.

Nos résultats se basent sur une étude analytique, par simulation et expérimentations. Ils montrent l’efficacité de nos propositions dans le champ d’application qui nous intéresse. Deux implantations, regroupant l’ensemble des techniques proposées, sont également présentées.

MOTS-CLES : réseau sans fil ad-hoc multi-sauts, réseau de capteurs, protocole MAC, auto-organisation, efficacité en énergie, étude analytique et expérimentale.

Laboratoire (s) de recherche : Laboratoire CITI, INSA de Lyon

Directeur de thèse : Isabelle Augé-Blum, Stéphane Ubéda

Président de jury : David Simplot-Ryl

Composition du jury :
Anne-Marie Kermarrec, INRIA Senior Researcher, IRISA/INRIA-Rennes (Relecteur)
Marcelo Dias de Amorim, Chargé de Recherche CNRS, Univ. Pierre et Marie Curie (Paris 6) (Relecteur)
Isabelle Guérin-Lassous, Professeur à l’Université Lyon 1 (Examineur)
David Simplot-Ryl : Professeur à l'Université Lille 1 (Examineur)
Isabelle Augé-Blum, Maître de Conférences à l'INSA de Lyon (Co-Directeur de thèse)
Stéphane Ubéda, Professeur à l'INSA de Lyon (Co-Directeur de thèse)
Dominique Barthel, Principal Research Scientist à Orange Labs, Grenoble (Co-Encadrant de thèse)