View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Duisburg-Essen Publications Online

INDOOR WIRELESS METERING NETWORKS

A COLLECTION OF ALGORITHMS ENABLING LOW POWER / LOW DUTY-CYCLE OPERATIONS

DISSERTATION

to obtain the academic grade
doctor rerum naturalium
(dr. rer. nat.)
in Computer Science

Submitted to the
Faculty of Economics
Institute for Computer Science and Business Information Systems
University of Duisburg-Essen

by
Dipl.-Ing. Nicola Altan
born on Oct. 16, 1970 in Montecchio Magg. (VI), Italy

Reviewers:

1. Prof. Dr. Erwin Rathgeb
2. Prof. Dr. Bruno Müller-Clostermann

Submitted on: 03.09.2008

Date of Disputation: 20.01.2009

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Arbeit selbständig ohne fremde Hilfe verfaßt und nur die angegebene Literatur und Hilfsmittel verwendet zu haben.

Nicola Altan

Abstract

Wireless Metering Networks (WMN), a special class of Wireless Sensor Networks (WSN), consisting of a large number of tiny inexpensive sensor nodes are a viable solution for many problems in the field of building automation especially if the expected lifetime of the network permits to synchronize the network maintenance with the schedule for routine maintenance of the building. In order to meet the resulting energy constraints, the nodes have to operate according to an extremely low duty cycle schedule.

The existence of an energy efficient MAC Layer protocol, the adoption of a robust time synchronization mechanism and the implementation of effective network discovery and maintenance strategies are key elements for the success of a WMN project.

The main goal of this work was the development of a set of algorithms and protocols which enable the low energy / low power operation in the considered family of WMNs. The development and validation of a propagation model reproducing the characteristics of the indoor radio environment was a necessary step in order to obtain appropriate instruments for the evaluation of the quality of the proposed solutions.

The author suggests a simple localized heuristic algorithm which permits the integration of all sensor nodes into a tree-like failure tolerant routing structure and also provides some basic continuous adaptation capabilities of the network structure.
A subsequent extension of the basic algorithm makes the network able of self healing.

An innovative approach to the solution of the synchronization problem based on a reformulation of the original problem into an estimation problem permitted the development of an efficient time synchronization mechanism. This mechanism, which makes an opportunistic usage of the beacon signals generated by the MAC layer protocol, permits an effective reduction of the synchronization error between directly communicating nodes and, indirectly, introduces a global synchronization among all nodes.

All the proposed solutions have been developed for a specific network class. However, since the presence of a low duty cycle scheduling, the adoption of a beacon enabled MAC protocol and the presence of limited hardware resources are quite general assumptions, the author feels confident about the applicability of the proposed solution to a much wider spectrum of problems.

Kurzfassung

Die Bezeichnung Wireless Metering Network (WMN) identifiziert eine spezifische Klasse von drahtlosen Sensornetzwerken. Solche Netze bestehen aus einer Vielzahl von kleinen, kostengünstigen batteriebetriebenen Knoten und bieten eine mögliche Lösung für unterschiedliche Aufgaben in der Gebäudeautomatisierung, vorausgesetzt dass die erwartete Lebensdauer des Netzes mindestens 10 Jahre beträgt, um die Netzwerkwartung im selben Raster mit den Gebäudewartungsarbeiten planen zu können. Die starken Energieeinschränkungen erfordern die Einführung von Energiesparmaßnahmen und insbesondere die Auswahl einer durch einen extrem geringen Arbeitszyklus charakterisierten Aktivierungsstrategie.

Schlüsselelemente für den Erfolg eines WMN-Projektes sind die Existenz eines energieeffizienten MAC-Protokolls, der Einsatz eines robusten Zeitsynchronisationsmechanismus und die Implementierung von effizienten Strategien für die Netzwerkinitialisierung und die Netzwerkwartung.

Hauptziel dieser Arbeit war die Entwicklung von Algorithmen und Protokollen, mit denen der energieeffiziente Betrieb einer spezifischen Familie von WSN ermöglicht wird. Die Entwicklung und die Validierung eines Ausbreitungsmodells für den Indoor-Funkkanal war ein erforderlicher Schritt, um die Untersuchung der entwickelten Verfahren zu ermöglichen.

Das erste im Rahmen des Projektes entstandene Ergebnis war ein heuristischer, robuster verteilter Algorithmus, der eine energieeffiziente Integration aller Sensorknoten und die Bildung einer robusten baumförmigen Routingstruktur ermöglicht. Derselbe Algorithmus ermöglicht eine begrenzte Anpassung der Netzstruktur an die wechselnden Charakteristiken des Funkkanals. Einfache Erweiterungen des ursprünglichen Algorithmus wurden hinzugefügt, um die Selbstheilungsfähigkeiten des Netzes zu verbessern.

Ein auf einer neuen Formulierung des Synchronisationsproblems basierendes Verfahren wurde entwickelt. Es gewährleistet eine energieeffiziente und robuste Zeitsynchronisation zwischen Nachbarnknoten und, indirekt, die Synchronisation aller Netzelemente.

Obwohl die vorgeschlagenen Lösungen für eine spezifische Netzkategorie entwickelt wurden, ist der Autor überzeugt, dass sich die Lösungsansätze auf ein weites Spektrum von Problemen anwenden lassen.

Contents

Abstract v

Kurzfassung vii

Contents ix

List of abbreviations xiii

List of Figures xvii

List of Tables xix

1 Introduction 1
- 1.1 Motivation 1
- 1.2 Problem description 2
- 1.3 Structure of the work 4

2 Introduction to Wireless Sensor Networks 5
- 2.1 Wireless Sensor networks - an overview 5
  - 2.1.1 A pervasive computing scenario 5
  - 2.1.2 Examples of sensor network applications 6
- 2.2 Exploring the design space 7
  - 2.2.1 Interaction pattern 8
  - 2.2.2 Requirements 8
  - 2.2.3 Enabling mechanisms 9
- 2.3 WSNs and MANETs are different 11

3 Architectural aspects 15
- 3.1 Hardware components 15
  - 3.1.1 The processing unit 16
  - 3.1.2 The communication device 18
  - 3.1.3 The power supply 20
- 3.2 Software architecture 22
  - 3.2.1 The Operating System 24
- 3.3 Protocol stack 26
  - 3.3.1 An ideal solution 26
  - 3.3.2 Standardization 27

4 An indoor WSN for metering applications **33**
4.1 An evolutionary tale . . . 34
4.2 Boundary constraints . . . 34
4.3 The MAC layer protocol . . . 35
4.3.1 Considerations on the properties of a MAC protocol . . . 35
4.3.2 A low energy / low duty cycle MAC . . . 37
4.4 The life-cycle of a WMN . . . 40

5 A simulation model for Indoor WSNs **43**
5.1 Different approaches to the Wireless Sensor Network (WSN) study . . . 43
5.1.1 Implementation . . . 43
5.1.2 Theoretical approach . . . 44
5.1.3 Heuristic approach (Simulation) . . . 45
5.2 Considerations for the propagation model . . . 46
5.3 *Islands* model . . . 47
5.3.1 Implementation . . . 49
5.3.2 Model validation . . . 50
5.3.3 Further considerations . . . 51
5.4 Simulation Framework . . . 52
5.4.1 The classes *BasicModule* and *BasicUpperLayer* . . . 53

6 Network bootstrap **55**
6.1 An overview . . . 55
6.1.1 Related works . . . 57
6.2 A heuristic algorithm . . . 58
6.2.1 Problem definition . . . 58
6.2.2 Dealing with unidirectional links . . . 59
6.2.3 Asymptotical results . . . 62
6.2.4 Simulation study . . . 63
6.2.5 Simulation results . . . 64
6.2.6 Further considerations . . . 70

7 Network self configuration and self healing **73**
7.1 Self-healing extensions . . . 73
7.1.1 Requirements and assumptions . . . 73
7.1.2 Basic self-healing algorithm . . . 74
7.1.3 Adaptation after the convergence . . . 75
7.1.4 Simulation Study . . . 76
7.1.5 Further considerations . . . 81

8 Time synchronization **83**
8.1 Introduction . . . 83
8.1.1 Origin of the frequency offset . . . 84
8.2 Existing protocols . . . 85
8.2.1 The Network Time Protocol (NTP) . . . 85
8.2.2 WSN synchronization algorithms . . . 87
8.3 Synchronization in a beacon enabled networks . . . 90
8.3.1 Problem description . . . 90
8.3.2 Proposed WSN synchronization algorithm . . . 91

8.3.3 Simulation 94
8.3.4 On the implementation on low-resource devices 100
8.3.5 On the reinitialization of the KF 101
8.3.6 Mutual synchronization of the clock references 102
8.3.7 Further considerations 103

9 Network behavior with all mechanisms activated 105
9.1 A 3-D node placement 105
9.2 Sensitivity to the variation of the propagation conditions 106
9.3 Sensitivity to the variation of the node density 107
9.4 On the guard period selection 108

10 Conclusions 111
10.1 The propagation model 111
10.2 Network bootstrap 112
10.3 Self-healing 112
10.4 Time synchronization 112
10.5 Final consideration 113

A The Kalman Filter 115
A.1 The equations of the Kalman Filter (KF) 115
A.2 The KF equations in presence of colored noise 117

Bibliography 119

List of abbreviations

6LoWPAN	IPv6 over Low-Power Wireless PAN
ADC	Analog Digital Converter
AODV	Ad-hoc On Demand Distance Vector
API	Application Programming Interface
ARQ	Automatic Repeat-reQuest
ASE	Absolute Synchronization Error
ASK	Amplitude Shift Keying
ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BI	Beacon Interval
BST	Breadth-First Spanning Tree
CDMA	Code Division Multiple Access
CFP	Contention Free Period
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
DARPA	Defense Advanced Research Projects Agency
DC	Direct Current
DLL	Data Link Layer
DSP	Digital Signal Processor
DSSS	Direct Sequence Spread Spectrum
EEPROM	Electrically Erasable Programmable Read-Only Memory
EMC	Electromagnetic Compatibility

EMA	Exponential Moving Average
FCS	Frame Check Sequence
FDMA	Frequency Division Multiple Access
FIFO	First In First Out
FLL	Frequency Locked Loop
FPGA	Field-Programmable Gate Array
FPU	Floating-Point Unit
FSK	Frequency Shift Keying
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HERA	Hierarchical Routing Algorithm
HF	High Frequency
HW	Hardware
KF	Kalman Filter
IES	Information Exchange Service
IP	Internet Protocol
IPv6	IP version 6
ISM	Industrial, Scientific and Medical
LNA	Low Noise Amplifier
LOS	Line of Sight
LR	Long Range
LTS	Lightweight Time Synchronization protocol
MAC	Medium Access Control
MANET	Mobile Ad Hoc Network
MASE	Mean Absolute Synchronization Error
MHR	MAC Header
MPDU	MAC Packet Data Unit
MSDU	MAC Service Data Unit
MST	Minimum Spanning Tree

MTU	Maximum Transmit Unit
NG	Neighbor Graph
NLME	Network Layer Management Entry
NTP	Network Time Protocol
OS	Operating System
PA	Power Amplifier
PAN	Personal Area Network
PDA	Personal Digital Assistant
PHY	Physical Layer
PLL	Phase-Locked Loop
PPDU	PHY Packet Data Unit
PSDU	PHY Service Data Unit
QoS	Quality of Service
RAM	Random Access Memory
RBS	Reference Broadcast Synchronization
RF	Radio Frequency
RFC	Request for Comments
RG	Radio coverage Graph
ROM	Read-Only Memory
RSSI	Radio Signal Strength Indicator
SAP	Service Access Point
SINR	Signal to Interference Noise Ratio
SR	Short Range
SW	Software
TDMA	Time Division Multiple Access
TPSN	Timing-sync protocol for sensor networks
UDG	Unit Disk Graph
UTC	Coordinated Universal Time
VM	Virtual Machine

WMN	Wireless Metering Network
WSN	Wireless Sensor Network
ZDO	ZigBee Device Object

List of Figures

1.1 Comparison between the existing and the envisaged WSN based solution 2

3.1 Components of a sensor nodes 16
3.2 Transceiver's structure 18
3.3 Battery equivalent circuit 21
3.4 Pulse discharge of lithium coin cell 22
3.5 Sensor node software architecture 23
3.6 TinyOS component's structure 25
3.7 A possible cross-layering enabled stack 26
3.8 The ZigBee protocol stack 27

4.1 MAC channel access 38
4.2 Contention based channel access 39
4.3 Phases of the network life 40

5.1 Propagation models used for the theoretical analysis of WSN algorithms 45
5.2 Example of indoor node placement 46
5.3 The origin of the *Islands* model 48
5.4 The propagation model 49
5.5 Rooms and measurement points arrangement 50
5.6 Structure of a node 52
5.7 The derivation of the class *BasicUpperLayer* 53

6.1 A simple NG 55
6.2 Relative frequency of connected networks in dependency on different duty cycle values 56
6.3 Example of NG 59
6.4 Example of the structure generated by the algorithm 1 by removing the constraint on n 61
6.5 Example of the structure generated by the algorithm 1 for $n = 5$ 61
6.6 Node placement 63
6.7 Impact of the number of synchronized neighbors n ($m = \infty$, $p = 0.6$) . . 65
6.8 Impact of the number of neighbors initially stored m ($n = 5$, $p = 0.6$) . . 66
6.9 Impact of the update probability p ($n = 5$, $m = 10$) 67
6.10 Update generation over time ($n = 5$, $m = 10$) 67
6.11 Convergence time vs. depth of the tree structure ($n = 5$, $m = 10$, $p = 0.6$) 69
6.12 Topological observations ($m = 10$, $p = 0.6$) 70

7.1 Schematical description of the scan process with $n = 3$ and $u = 1$ 74

7.2 Example: Node C loses the direct connection to A (a). Node B notices the event and generates an update which permits the reintegration of C (b) . . . 75
7.3 Network setup for different number of directly communicating nodes . . 77
7.4 Network behavior if the 10 most important relay nodes fail. (100 nodes distributed over 5 aligned rooms) . . . 78
7.5 Network behavior in presence of a progressive erosion of the number of active nodes (3D building structure with the sink placed in a corner) . . 80

8.1 NTP data exchange . . . 86
8.2 Logical structure of NTP . . . 86
8.3 NTP errors cumulative distribution function . . . 87
8.4 The effect of wave superimposition in [73] . . . 90
8.5 Schematic representation of the blackbox approach . . . 91
8.6 Network structure used for the simulation study . . . 96
8.7 Behavior for jitter error model (beacon interval $360s$) . . . 97
8.8 Influence of prefiltering in presence of jitter . . . 97
8.9 Influence of prefiltering on the standard deviation of the estimated beacon interval, in presence of different clock error models . . . 98
8.10 Dependence of the standard deviation of the beacon interval estimation on the network length l . . . 98
8.11 Behavior in presence of jitter for increasing number of nodes at each level (columns) . . . 99
8.12 Standard deviation of the estimated beacon interval – 20x15 nodes grid network, $\sigma_{jitter} = 0.1T_{beacon}$ . . . 100
8.13 Influence of EMA prefiltering on the standard deviation of the estimated beacon interval . . . 101
8.14 KF reinitialization in presence of jitter . . . 102
8.15 Mutual synchronization . . . 103

9.1 Dependence on the attenuation between adjacent rooms . . . 106
9.2 Dependence on the number of nodes in each room . . . 107
9.3 Sensitivity to the variation scale factor S_{ASE} . . . 109

A.1 The system model used for the derivation of the KF equations . . . 116

List of Tables

3.1 Examples of microcontroller performances 17
3.2 Energy densities for various battery families 21

5.1 Approaches to the WSN study 44
5.2 Typical attenuation values for 2.4 GHz signals 47
5.3 Measured and estimated path loss 51

6.1 Updates per node and beacon generated during the whole simulation . . 68
6.2 Frequency of partially connected structures 69

7.1 Duration of the failover process 79
7.2 Connectivity breakdown due to network erosion 81

8.1 Time syncronization performances for different clock error models . . . 99
8.2 Evolution of the KF reinitialization rate 102

Chapter 1

Introduction

Each time a new technology moves the first steps from the research labs towards a commercial utilization, a natural approach consists in trying to apply it to some well known problems.

The Wireless Sensor Networks (WSNs) do not represent an exception to this usual development path. Many companies are currently analyzing the opportunities which this relatively new technology offers. Since at this time is still unclear which applications benefit from the WSN approach, the attempts cover a wide set of possible applications (see section 2.1.2).

In any case, since the integration of WSNs into commercial products is just moving the first steps, it has to be considered experimental. The developers are not used to deal with the particularities of these new systems and they need support from the scientific community.

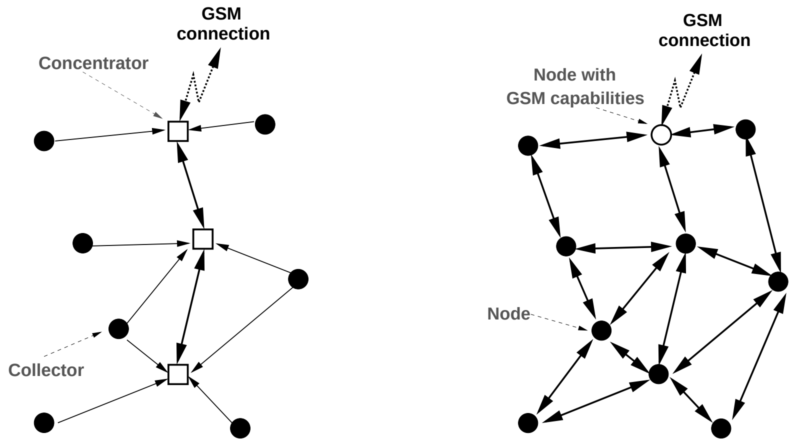
1.1 Motivation

Two years ago, a European company based in Germany active in the field of building management was facing the need of developing a new platform for the data collection, especially because the production of some components (microcontroller and transmitter) used in their existing data collection system had been discontinued.

The old systems consisted of a heterogeneous network of many weak data collection nodes providing periodical measurements (at least once a day) of water and heating consumption. Some much more powerful concentrator nodes were in charge of collecting and propagating the measurements to a central data processing facility (Fig 1.1(a)).

The collector nodes, equipped with a simple transmitter, were unable of bidirectional communication. The periodically collected data was broadcasted towards the few concentrator nodes. The concentrators were able of bidirectional communication and interacted by building a “backbone network”. The concentrators were continuously active in order to receive the data transmitted by the collector nodes. Direct consequence of the continuous transceiver activity was the necessity of providing the concentrator nodes with much larger energy sources than the collectors.

Manufacturing and warehousing of two different kinds of nodes was identified as a source of possible inefficiency. Moreover, the unidirectional communication between collectors and concentrator did not permit the remote management of the sensing nodes and made the adaptation of the network in case of nodes failure and modification of



(a) Existing solution (b) WSN based solution

Figure 1.1: *Comparison between the existing and the envisaged WSN based solution*

the propagation environment difficult. Both these facts required a new approach to the data collection problem.

The development of new hardware for collector and concentrator nodes and in particular the adoption of the same bidirectional transceiver for both components offered the opportunity of replacing the old paradigm by a new model of data collection networks.

Thanks to the new transceiver and the relatively more powerful processor, the developers envisaged the possibility of having a self configuring network consisting of homogenous nodes which are able of self organization and which build a network with redundant links (Fig. 1.1(b)).

In order to check the feasibility of the envisaged solution, the company asked for the collaboration of the Computer Networking Technology group at the university of Duisburg-Essen. The resulting two-year project was aimed at developing and validating a set of algorithms particularly tailored for the considered application and able to cope with the strict constraints of technological as well as economic nature (see section 4.2).

Simulation had been adopted to validate the behavior of the proposed solutions both because of the unavailability of a HW platform (unfortunely, the first prototypes became available after the end of this project) and because of the good compromise between flexibility and quality of the results.

Some of the main results of this profitable cooperation have been patented and will be implemented into real systems.

1.2 Problem description

A network of simple battery powered collector nodes, mainly consisting of commodity components, has to work unattended for at least 10 years and has to propagate at least one measurement per node and day to a concentrator node. Moreover, the network has to be capable of self initialization and self healing.

A single lithium coin cell is the primary energy source of each node. Only half of the stored energy can be used for data communication purposes. The energy constrains and the long expected operational life impose the adoption of an extremely efficient energy management and an activation strategy characterized by an extremely low duty cycle.

The goal of reducing the development and debugging costs suggested to reuse part of the preexisting code. In particular, a new Medium Access Control (MAC) protocol had to be developed on the basis of the preexisting one which had been used for the communication between concentrator nodes.

Preliminary theoretical considerations highlighted the existence of some critical aspects with respect to low power / low duty cycle operations which the energy constraints impose to the nodes. Hence, the development of efficient solutions for the following critical problems has been defined as main project goal.

Network Initialization After the deployment each node has to find the possible communication partners. A *greedy* localized algorithm has to promote the creation of a routing tree rooted at the concentrator. The concentrator is a specialized node, which propagates the data by using a different telecommunication technology (like GSM or UMTS). Because of failure tolerance and load balancing considerations the routing structure has to include redundant links. Since the node installation is a supervised process carried out by a human operator, the nodes might rely on information provided by some helping device in order to carry out the first initialization stages.

Failure Recovery During its operational life, which spans a quite extended time interval, a network has to cope with the modification of the propagation environment, the presence of disturbing signals and eventually with the malfunctioning of some nodes. Moreover, nodes might be activated some time after the end of network deployment and the network has to be able to integrate these new elements into the routing structure. The responsibility to fulfill these requirements is given to a failure recovery mechanism which, even in the extreme case of a total network destruction, has to rebuild the network autonomously. The failure recovery mechanism has to be compatible with the energy constraints and the low duty cycle activation strategy.

Time Synchronization The interval between two consecutive activity periods of a node can be as big as some minutes. Since each node is equipped with a free running clock which is characterized by a non ideal behavior[1] and, moreover, since environmental changes, e.g. temperature changes, may impose additional clock errors, it is obvious the necessity of a mechanism to reduce the errors in the time estimation. The time synchronization algorithm has to cope with the presence of asymmetrical links and, if possible, has to generate only a minimal communication overhead.

The proposed results, which are original contributions of the author, have been developed by considering the specific metering application. However, since the presence of a beacon enabled MAC and the adoption of a low duty cycle activation strategy are common to many other networks, these are likely applicable over a much wider range of sensor network scenarios.

---

[1] A typical clock error of 20 ppm, e.g., might results in a difference of 7.2 ms in the estimation of a 6 minute interval carried out by two different nodes.

1.3 Structure of the work

A brief introduction of the WSN topic and the analysis of some commonly accepted solutions can be found in the chapters 2 and 3.

The following chapter 4 offers a description of the boundary constraints to this specific project and shows some original modifications of the MAC protocol which have been proposed in order to improve the efficiency in case of potentially dense networks (section 4.3.2).

The Chapter 5 is devoted to the description of an innovative propagation model which has been developed with the purpose of reproducing the peculiarities of the indoor network operations in the simulation.

Analysis of the single problems identified in section 1.2 along with the proposed solutions and the results of the respective simulation based validation are presented in the following chapters (chapters 6, 7 and 8, respectively).

The chapter 9 provides an analysis of the overall behavior of a network in the case of simultaneous activation of all the proposed mechanisms.

The chapter 10 contains some conclusive considerations about the whole work.

Chapter 2

Introduction to Wireless Sensor Network

2.1 Wireless Sensor networks - an overview

2.1.1 A pervasive computing scenario

Over the last years an impressive growth of the capabilities of computing and communication devices has been observed. Nowadays it is obvious how easily a huge mount of data can be accessed and processed, but these are at best indirectly related to the physical environment. At the same time, but in a less evident way, tiny dedicated computing devices have been embedded in an ever growing number of products. Nowadays it is difficult to find a refrigerator or a washer which does not integrate a microprocessor. Typically, such *embedded systems* do not require an interaction with a human operator but are rather required to work autonomously. The number of such devices is increasing, and there is a tendency to include processing units not only into larger appliances, but also into simple and even disposable goods. In the near future, we will be surrounded by computation devices which collect and process information and interact with the human users. The user will not perceive the presence of these technologies but will have the feeling of a direct interaction with the surrounding physical world. A key role for the realization of this vision is the capability of all computing devices to communicate.

Even though networks of sensors and actuators can be realized using the existing wired communication systems, this seems not to be a viable approach except for a restricted number of applications. On one side the deployment of a wired network is associated with non negligible costs and on the other side the presence of wires prevents the mobility of the connected devices.

Recently a new class of networks has been considered. The generic name *Wireless Sensor Network* (WSN) [1] has been used in order to identify all that networks whose nodes are able to interact with the surrounding physical environment and which adopt a wireless technology for the communication (see e.g. [1–3]).

Typically, the nodes of a WSN have to cooperate in order to fulfill a specific task, as a single node is unable to do this alone. Versatility is the very strength of such a

---

[1] Even though a network may include besides the sensors some specialized nodes which act on the surrounding environment (actors), the name Wireless Sensor Network (WSN) has been preferred to others like *Wireless Sensor and Actor Network*.

technology, which may be employed to solve a lot of very different real world problems. Therefore, there is no well defined set of requirements which univocally identifies a WSN and there is also no single technical solutions which cover the whole design space. For this reason the study of the WSN can not be based on a unified, well structured and universally accepted theoretical framework.

2.1.2 Examples of sensor network applications

The proponents of WSN based approaches claim the ability of this technology to simplify the solution of many existing problems and to open the path to completely new applications.

The following list gives a short overview of some possible application areas of the WSNs, which can be found in the literature (see e.g. [1,4–6]). The enumeration is probably incomplete; in fact it is aimed rather at showing the flexibility of the technology than at providing an extensive description of all possible application fields.

Military applications Since the beginning, the WSNs did attract the attention of the military. In fact, a self configuring network consisting of a large number of disposable elements seems to be a really interesting structure to collect information in inhospitable and inaccessible areas. As noted by Kumar and Chong [5], the development of WSN is a natural evolution of the earlier studies on sensor networks started in the 1980s and 1990s. In these years military sensor networks appeared for the first time on the battlefield with the purpose of providing a detailed description of the operation area. The enabling factors for this new approach have been the availability of inexpensive low power processors and the developments in the wireless telecommunication technologies. An example of a WSN based vehicle tracking system can be found in [7]. Most of the WSN researches in the USA have been largely sponsored by Defense Advanced Research Projects Agency (DARPA).

Disaster Relief Application Historically the idea of developing a self-configuring communication network able to operate autonomously in an inhospitable environment was one starting point for the research on Mobile Ad Hoc Networks (MANETs). The idea of using a WSN in a heavily damaged area is a logical further development of that initial approach. Monitoring a wildfire, measuring the contamination after a chemical incident or finding the position of persons which have been swept away by an avalanche may be interesting application areas for this technology. Some of these applications have commonalities with military applications, where the nodes should detect, for example, enemy troops instead of wildfire.

Environment Control A continuous monitoring of air quality or the observation of the movements of stones or snow masses in order to forecast landslips and avalanches may be carried out by a WSN. Long time unattended, wire free operations are the most important aspects for this kind of applications. Clearly, number of nodes depends on the size of the monitored area and on the characteristic of the particular event. A distance of many kilometers between neighboring nodes is not unusual for systems devoted to the avalanche prediction.

Intelligent Buildings Heating, air conditioning and lighting are responsible for non negligible energetic costs. These costs could be reduced by making the different appliances and control units aware of the current values of some related physical parameters. Modern buildings are equipped with wired sensors in order to permit optimal energy utilization, So, for instance, the air conditioning can be automatically turned off in presence of an open window. A WSN could be used in old buildings in order to enable the same degree of automation, without requiring the high expenses associated with the cabling process. In this type of applications, if the nodes are battery operated (e.g. if the nodes are retrofitted in an existing building), the importance of the energetic efficiency of the single nodes can be extremely high. In fact the lifetime requirements can be very high - up to a dozen of years of unattended operation - in order to fit the node substitution into the normal building maintenance schedule.

Facility Management Another possible application field is the management of facilities larger than a single building. WSNs can be used for keyless access control purposes or maybe to keep traces of the movement of possible intruders.

Medicine and Healthcare Parallel to the improvement of the medicinal science, we observe an ever-increasing necessity of monitoring patient's functions even over long time periods. Since the usage of wired devices has a negative impact on the ability of the patient to move freely, there is an increasing interest in developing wireless wearable medical devices.

Logistics The simplest application in this field consist of equipping goods with simple tags that allow simple tracking of these objects during transportation or that facilitate inventory tracking in stores or warehouses. More complex scenarios can be developed by adding awareness of the surrounding environment to the nodes, e.g. in order to prevent the storage of chemical products which may cause dangerous reactions in the same warehouse.

Self-Healing mine fields, rescue of avalanche victims and cold chain managements are additional examples of WSN applications, which can be found in an introductory article of Römer and Mattern [8], along with some architectural considerations.

2.2 Exploring the design space

As seen before, a WSN based approach may apply to the solution of many different problems. In spite of the variety of possible network structures, it is possible to identify a subset of the network characteristics which is shared by almost all the networks.

With respect to the data flow it is possible to divide the nodes in two groups: **sensors** and **sinks**. The sensors are that elements which are able to collect data by observing the surrounding environment, while the sinks are the nodes where the measurements should be delivered to. Usually there are much more sensors than sinks, and sinks may be specialized nodes belonging to the network or may be a sort of gateway toward a different network.

2.2.1 Interaction pattern

According to [4], the observation of the interaction between sensors and sinks allows highlighting some typical behavior patterns. Some of the most important patterns are reported in the following list.

Periodic data collection Sensors may be used to report values measured periodically. The reporting period and the observation time are application dependent. They can be either known right from the beginning or dynamically adjusted in response to some external event.

Event Detection In this case the data generation is triggered by the occurrence of an event, of which the nodes known they have to report it to the sink. While simple events, like the breaking of a pane of glass, normally require the intervention of a single node, more complicated events may require the intervention of many neighboring nodes in order to obtain a correct interpretation of the collected data.

Tracking Once an event (e.g. the appearance of an intruder) has been detected, it can be useful to follow its subsequent progression, by tracing the movements of the event source (e.g. the intruder). In such a scenario the nodes can cooperate in order to report the actual position of the observed object and maybe to provide an estimation of its direction and speed.

Map Generation Sometimes an application requires the knowledge of the state of some variable over the whole observed areas (playground) and at each point in time. In such a case the whole network behaves like an interpolator, which approximates an unknown function of position and time, by using the data series collected by the individual sensors. The accuracy of the approximation depends on the reporting rate and node density and hence there is a trade-off against the energy consumption.

2.2.2 Requirements

Since the range of possible sensor network applications is wide spread, it is not possible to define a set of requirements which are common to all WSNs. The following list is aimed at providing an overview of common characteristics observed in many different sensor networks.

Data Centric Approach Traditional communication networks have been developed with the specific goal of moving data between some nodes. In a WSN, the data propagation is, on the contrary, only a (non secondary) aspect which is necessary to achieve the goal of providing valuable information with respect to a given task. Geographical and temporal information may be required in order to correctly evaluate a sensed event. A completely new approach to networking, along with new interfaces and new service models is necessary in order to put the data itself into the focus of the network operation.

Operational Life Even though in some cases the nodes may be powered via wires from a central power supply, the mainstream idea assumes having autonomously self powered systems which may operate unattended even in the absence of any sort of infrastructure. Since the nodes operate with a limited supply of energy, achieving the expected network lifetime can become a critical task which can heavily influence the feasibility of a project. In some cases the adoption of small energy scavenging systems [9](e.g. solar cells or thermoelectric devices) may be a viable solution to extend the network lifetime by partially recharging the batteries. Energy saving strategies may have a direct trade-off against the quality of services. Hence it is necessary to identify some strategies to balance these conflicting aspects. There is no univocal definition of the operational lifetime. Some authors consider the time until the first node ceases to operate, some others consider the time until the network gets partitioned, yet another alterative may be based on the extension of the observed area.

Fault Tolerance Nodes may run out of energy, might be damaged or the radio channel may be disturbed for a long interval. Therefore, it is necessary for a WSN to deal with such error conditions. Several strategies might be used to improve the fault tolerance. The most common approaches include redundant deployment and continuous network optimization.

Scalability On one side a network may include a large number of nodes, on the other side the node density might experience a noticeable variability depending on the different applications. The architectures and protocols used have to be able to deal with these two aspects by scaling with the number of nodes and by adapting to the variations of the node density. It has to be noted that even in the same network the node density might vary from region to region (e.g. non uniform deployment) or/and with the time (e.g. the nodes get energy depleted).

Maintainability A WSN and its environment are continuously changing and each network element has to adapt to the new conditions. It has to monitor itself and adapt the operation strategies according to its health status (e.g. a node may choose to propagate fewer data in order to prolong its lifetime when the residual energy reaches a given threshold). The network could also be able to interact with a maintenance system. Another aspect of the maintainability involves the ability of the nodes of being reprogrammed in order to change their tasks or to correct some error. The *programmability* requires the adoption of dedicated mechanisms at all levels of the protocol architecture.

2.2.3 Enabling mechanisms

As stated before, a WSN differs from all previous types of communication networks. The well known protocols and architectures, which have been developed during the past decades, do not fit all the characteristics specific the WSN. New concepts, new architectures as well as new protocols have to be developed to satisfy the requirements of this new network class.

Exploring the boundaries of the project space may be a good starting point for the selection of the mechanisms needed by a WSN. Often the developers need to find a compromise between contradictory goals at each level of the system development. Examples of these trade-offs may be: energy saving vs. result accuracy, network lifetime vs. lifetime of the single nodes, algorithm efficiency vs. hardware constrains, etc. From this point of view it is clear that a WSN is not a general purpose network and that it is impossible to find a solution which fits to all possible projects. On the other hand, a development and implementation completely from-scratch would not be economically feasible in many cases. Fortunately, there are an increasing number of results for each of the following issues which may be successfully integrated in almost every new project.

The mechanisms described in the following list can be typically found in almost every WSN.

Energy Awareness / Energy Efficient Operation As said before, the energy constraints have a large impact on the feasibility of a WSN project. First, the developers need to adopt strategies which are able to reduce the energy required for the data transmission between neighboring nodes. At the same time the network as a whole should be aware of the energy state and try to evenly distribute the work over all nodes. In addition, another common concern is the efficient acquisition of the required information.

Multihop Communication Given that the nodes communicate using a wireless technology, it is clear that a direct communication among all nodes is infeasible in many cases. In particular, the direct communication over long distances requires the generation of strong radio signals. This is, on one side, problematic in presence of extremely constrained energy resources and on the other side inefficient from the point of view of the interference in a dense network. The previous considerations suggest the necessity of adopting a multihop communication structure where the nodes are able to operate like a router by retransmitting the received data.

Data Centric Communication A traditional network is a (sparse) structure devoted to the transfer of data between two specific devices; each one of them is identified by at least a network address. According to the address the network nodes propagate the data regardless of their meaning. The WSNs are based on a different approach. They are aimed at providing some valuable data collected from the surrounding environment. Therefore, the data transport loses its importance in favor of the data self. Moreover, in presence of a dense network structure, which may be characterized by an high degree of redundancy , it makes no sense trying to identify the specific node which has provided the data, whereas it might be interesting to know the position of the event. In such a context it is necessary to move from an *address-centric* approach to a *data-centric* one. The mechanism used in order to ask for some data will probably resemble the mechanism used to interact with a database [10]. As example we can consider an air quality monitoring application, where it is more interesting to ask for the pollution in a given geographic region than to ask a specific node for its measurements.

Self-configuration The nodes belonging to a WSN have to configure most of their operational parameters autonomously. Because of the number of involved nodes and the deployment strategies, it is inconceivable that an operator would manage each single node in order to ensure the network operation. The long lasting lifetime forces the network to deal with modifications of the surrounding environment and with the progressive erosion of the node population. Similarly, when the network operator decides to retrofit some nodes, e.g. in order to replace the depleted ones, the new elements have to be able to interoperate with the existing ones.

Collaboration Sometimes the observations done by a single node are not sufficient in order to univocally identify an event. In such a case several sensors have to cooperate in order to provide a more complete description of the observed event. Even though the data evaluation may be done at the sink, performing that function during the data propagation might result in a more efficient utilization of the resources. As example the average temperature over an observed area might be computed by aggregating the raw measurements, while these propagate towards the sink. In such a way it is possible to reduce the overall data transmission costs.

In addiction to these mechanisms there are some design guidelines the developer of a network should consider. One of them is the principle of *locality*, which has to be applied in order to ensure in particular the scalability. Since most of the nodes are really limited in resources like memory, energy and computational power, all the implemented algorithms have to rely only on a extremely limited amount of stored information, at most concerning the direct neighbors.

Implementing all the proposed mechanisms may be a particularly challenging task. Some of them require a deep reconsideration of the existing paradigms. As example an energy efficient network protocol stack unlikely fits into the well known OSI network communication model, which does not envisage the presence of “shortcuts” between the different protocol layers.

2.3 Wireless Sensor Networks (WSNs) and Mobile Ad Hoc Networks (MANETs) are different

Based on the previous description it is natural for the reader to compare the WSN with another class of self-configuring and infrastructure less networks which have been studied extensively in the past, the Mobile Ad Hoc Networks (MANETs). In general, an ad-hoc network is a structure, which is set up with the precise purpose of satisfying a quickly appearing communication need. An example for this network type may be the networking of few laptops in a meeting room, where the lifetime of the network is upperbounded by the meeting duration. It is clear that in such a case the simplicity of network setup and operation, and hence the self-configuration, play a critical role in the acceptance of the technology. The MANET is an extension of the original idea of ad-hoc network, which has been obtained by adding the requirements of multihop communication and node mobility support. As example a MANET may be used in order to provide a data network to support the operation of emergency services in case of absence or malfunctioning of the communication infrastructures. In such a scenario multihop communication is required in order to extend the network coverage, while the mobility support is necessary in order to “follow” all the persons involved in the operations (firemen, policemen, etc.). This is a very active research area, and there are many good books on this theme, like per example the works of Perkins [11].

Even though there are some similarities between MANET and WSN, from a macroscopic point of view, it has to be noted that at the same time there are many differences, which prevent an unified treatment of the two kinds of networks. The following list is aimed at highlighting some important differences between the two network families.

Equipment, Application and Data The applications running in a MANET go beyond the simple data collection. In fact, these networks are aimed at satisfying the communication needs of a group of human users. The terminal may be fairly powerful as a PDA or a notebook and even though these are battery powered, the amount of available energy is significantly bigger than in the typical sensor nodes. The characteristics of the data traffic and the Quality of Service (QoS) requirements might show a wide variability ranging from a moderate quantity of data, exchanged in a delay tolerant fashion, for a web based application to a fairly big amount of delay sensitive data belonging to an audio or video stream.

Network Size It is reasonable to assume that the number of active nodes in a MANET scales with the number of users contemporarily present in the deployment area and this number not easily grow above some hundreds. In a WSN the number of nodes depends mainly on the extension of the observed area, on the characteristics of the nodes and on the required redundancy and it is not difficult to imagine networks with many thousands of nodes.

Human Interaction vs. Environment Interaction The characteristic of the traffic on a MANET, which consists mainly in a mix of well known services like Web, voice, mail, etc., will show similarities to the traffic observed in the global internet. The WSNs interact with the surrounding environment and are likely to exhibit a low data rate over a large time scale, even though a bursty distribution of the traffic can not be excluded. It might happen that almost simultaneously many sensors try to report their observation in response to a sudden variation of the environmental conditions.

Simplicity As stated before a sensor node should be a simple and cheap element. Hardware constraints (processor, memory, batteries, etc.) impose the adoption of simple and efficient mechanisms. These constraints do not apply to MANETs, where the nodes are mostly state of the art computation devices.

Mobility The concept of MANET emphasizes the idea of having plenty of users / terminals which are changing their position continuously. Consequently, the network has to adapt the routing process in order to follow the node movement.

Generally, most of the nodes belonging to a WSN do not change their position after the deployment phase. This fact does not exclude the existence of a certain network dynamic which, however, depends mainly on the modification of the environment, the presence of disturbing signals and malfunctioning of the nodes. It is even possible that some nodes (actuators) or the sink experience some sort of mobility, but this is not the behavior of the majority of the nodes. Hence, the algorithm used in a MANET in order to address the mobility may result in a overkill for the needs of a WSN.

Furthermore, WSNs may have to deal with a completely new mobility aspect, which does not have any correspondence in the previous networks and which is a direct consequence of the data centric approach. If a WSN is used in order to observe a physical phenomenon, a single event may cause some local processing in order to correctly understand the event. If the origin of the phenomenon moves over the observation area, the network has to be able to follow the event by propagating those partial results which are needed in order to carry out the required computations.

Because of the peculiarities of the WSNs, it is evident the impossibility of setting the study of these networks in the framework of some well settled research area. Consequently, WSN researches are carried out in an autonomous way and often require new and original approaches.

Chapter 3

Architectural aspects

It is not possible to talk about the Wireless Sensor Networks (WSNs) and the challenges of this technology without first analyzing the peculiarities of the different node architectures. Even though many of the earlier works on WSN focused the analysis of the sensor architecture (e.g. [1, 3, 6]), there is no common understanding of the capabilities of the single network elements.

In this section the author wants to analyze the characteristics of the various building blocks, with reference to some well known implementations, in order to point out his idea of a sensor node. After a first section which provides a concise description of the hardware components, this chapter continues with the description of some proposed solutions for the network protocol stack and the software architecture. The following discussion is partially based on the descriptions which can be found in the books of Callaway [12] and Karl [4].

3.1 Hardware components

The designer of a new sensor node has to deal with many and sometime conflicting goals. Hence, the resulting product will be the result of many compromises, for example between energy consumption and computation capabilities, or between production costs and available energy. The specific application imposes the requirements that the nodes might have to be small in size, long lasting, cheap, etc. Sometimes the requirements might be really challenging, like in [13] where the node was aimed at being cheaper than 1$ and at having a total energy dissipation upper bounded by $100\,\mu W$. However, regardless of the specific characteristics of the underlying hardware, almost all sensors (and actuators) share the same overall structure [3] sketched in Fig. 3.1.

A processing unit, consisting of a processor and some memory, processes all the data and controls the node activities. The processor is capable of executing arbitrary code. The storage might consist of different types of memory for programs and data.

The sensing unit and the optional actuator provide the interfaces of the node to the physical world. Their capabilities are strictly dependent on the specific application.

The transceiver enables the wireless data exchange between neighboring nodes and hence it is the key element in order to turn a set of nodes into a network.

Usually, the nodes have to operate autonomously, independently of the presence of infrastructure. In such a case the primary power supply consists mainly of some form of batteries. Energy scavenging devices may be used in order to partially reload

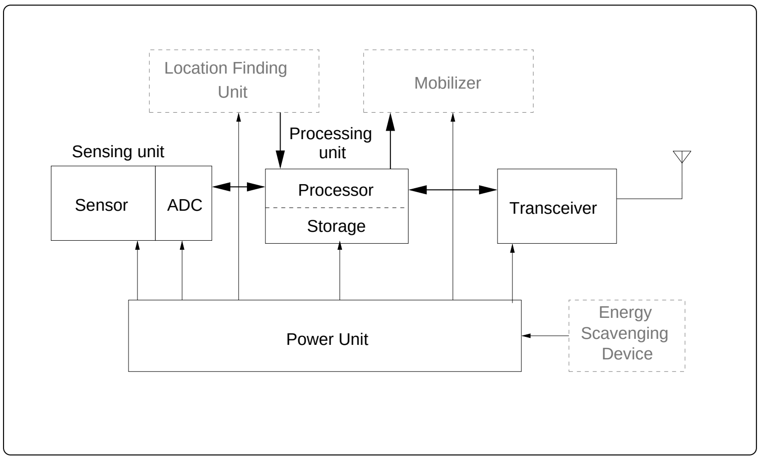


Figure 3.1: *Components of a sensor nodes [3]*

a rechargeable battery and, hence, to extend the node life.

3.1.1 The processing unit

The processing unit is the core of the sensor node. It is responsible for managing all node activities. It controls the data acquisition, processes the data, handles the communication with the neighboring nodes and operates the actuator. The processor has to execute different tasks ranging from time-critical signal processing activities to execution of some application program.

The easiest, but probably not the most efficient, solution consists in using a general purpose microprocessor. Unfortunately, these powerful and flexible elements are over dimensioned for the requirements of a sensor node and moreover are quite inefficient from the point of view of the energy consumption.

Fortunately, there is a class of much simpler processors, specially tailored for embedded system applications, which are known by the generic name *microcontroller*. These processors have some interesting characteristics very suitable for the realization of sensor nodes. They are characterized by low energy consumption and sometimes by the ability of throttling down the clock rate in order to further reduce the current absorption. A further reduction of the energy costs can be obtained by putting the processor into a *Sleep* state, which is characterized by the turning off of part of the internal circuits[1]. Often microcontrollers have built in memory and/or integrated Analog Digital Converters (ADCs) for the direct connection with sensing devices. Compared to a microprocessor, a microcontroller shows some (minor) drawbacks like the absence of a Floating-Point Unit (FPU) which requires the modification of many algorithms in order to work only with integer arithmetic. Another short coming is the absence of memory management functionalities which makes the usage of protected or virtual memory difficult, if not impossible.

The utilization of off-the-shelf processing units is surely an advisable option, in cases where the development costs have to be strictly controlled or no extreme efficiency is required. However, in some cases, especially if the efficiency has to play a major role, it

---

[1] E.g. the Frequency Locked Loop (FLL) can be turned off, which result in a reduction of the clock precision.

Table 3.1: *Examples of microcontroller performance [14]*

	ATMega128L	MSP
Word size	8 $bits$	16 $bits$
Max Frequency	8 MHz	6 MHz
Power Down	8 μA	1.8 μA
Idle (1MHz)	0.5 mA	55 μA
Idle (8MHz)	4 mA	440 μA
Active (1MHz)	2 mA	240 μA
Active (8MHz)	8 mA	1.9 mA
Wakeup	2 ms	6 μs

could be interesting to have some functionality directly implemented in the hardware. In this case the utilization of Field-Programmable Gate Arrays (FPGAs) or Application Specific Integrated Circuits (ASICs) might be a viable solution. The former are logical circuits which can be reconfigured in order to adapt to changing sets of requirements while the latter are dedicated processors, which are designed for specific purposes. By embracing one of these solutions a developer must consider a new trade-off between the loss in flexibility and the improvement of the energy efficiency. An increase of the hardware development costs has also to be taken into account.

A paper of Lynch and O'Reilly [14] addressed the problem of selecting the appropriate microcontroller for a sensor node. The authors did compare the performance of different commonly used microcontrollers with respect to energy consumption considering different clock rates, wakeup times, I/O capabilities and power save functionalities. The work pointed out the big variability of the performance parameters among the different microprocessor families and how some features, like the ability of dynamically throttle down the clock rate, can be exploited in order to optimize the energy consumption. As example the values for the Texas Instruments MSP family and for the Amtel ATMega128L (booth microcontrollers are employed in widely used sensor network development kits) are reported in Table 3.1. The big difference in the wakeup time seems to be mainly due to the different technologies adopted for the clock circuit.

The microcontroller needs some memory in order to store data and programs. The usual solution consists in providing a quite limited Random Access Memory (RAM) (in order to store intermediate sensor reading, packets from other nodes, computation results and so on) and some more Read-Only Memory (ROM) or, more typically, Electrically Erasable Programmable Read-Only Memory (EEPROM) or flash memory to store the program code. The RAM is fast, but it loses its content if power supply is interrupted, while the much slower EEPROM and flash memory are able to conserve the stored information independent from the power supply. The most evident difference between the flash memory and the EEPROM is that the former permits the modification of the stored information in blocks while on the letter these actions can be carried out only one byte a time. The flash memory might be used in order to temporarily store the state of the RAM when the power supply is turned off. The selection of the optimal amount of memory is strongly dependent on the specific application. As example the well studied *Mica Mote* is equipped with 4 KBytes RAM, 128 KBytes flash memory and 4 KBytes EEPROM [15].

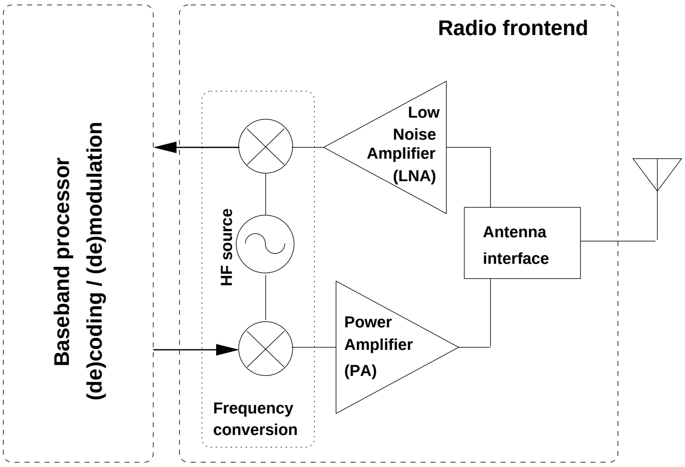


Figure 3.2: *Transceiver's structure [4]*

3.1.2 The communication device - Radio Frequency (RF) transceivers and antennas

Since the single nodes are supposed to communicate with each other in order to build a WSN, the transceiver is a mandatory component of each WSN element.

This device is responsible of converting a data stream coming from the microcontroller into a Radio Frequency (RF) modulated signal and sending it over the air. Symmetrically a transceiver has to be able to convert a RF modulated signal into a bit stream and pass it to the microcontroller. Usually because of the characteristics of the RF medium a transceiver accesses the medium in a half-duplex fashion; otherwise it would observe only its own transmitted signal.

The transceiver can be schematically divided in two distinct parts (Fig. 3.2): a *baseband processing unit*, which communicates directly with the microcontroller, and a *radio frontend* which moves the baseband signals to the actual radio frequency band.

An analysis of low power implementations of the RF front-end would require a separate treatment. In any case, it has to be noted that the energy consumption of a transceiver is largely due to its RF part. The components which are mainly responsible for the power drainage are the Power Amplifier (PA), the Low Noise Amplifier (LNA) and the local oscillator. The PA amplifies the High Frequency (HF) signals to the level required for the transmission. The LNA amplifies the received signals up to a level suitable for further evaluation, ideally without introducing additional noise. The last element, which might be implemented with a voltage controlled oscillator, in combination with a frequency mixer builds the frequency conversion bloc. There are other elements, like filters, belonging to the RF front-end, but their contribution to the total energy consumption is minimal.

During the network operation the transceiver may be put in one of the states *Transmit, Receive, Idle* or *Sleep.* The meaning of the first two states is obvious. During the *Transmit* state, the transmit part of the transceiver is active and the antenna radiates energy. Similarly, when in the *Receive* state, the receive part of the transceiver is active.

The *Idle* state identifies a transceiver which is ready to receive but is not currently receiving anything. In this state some parts of the receiver can be switched off and activated only after the detection of a signal.

In order to save even more energy, a significant part of the transceiver can be switched off. In this *Sleep* state the transceiver can neither send nor receive. The change from the *Sleep* to the *Idle* (and then *Send* or *Receive*) can not occur instantaneously. The *Recovery time* is the interval needed in order to resume the transceiver operations. The energy consumed during the (re)activation process is called *startup energy*. Sometimes the transceiver offers many different *Sleep* states by progressively turning off an increasing number of elements[2].

Connected with the RF part of the transceiver, the antenna directly influences the shape of the *communication area* of a node. This passive component transforms an electrical signal into a variable electromagnetic field and vice versa. It is probably the most underappreciated component of a sensor node. Often the developer has to find a trade-off between the performance of the antenna and the node packaging requirements. If the antenna has to be placed on the inside of the packaging, there are mainly three options which can be considered.

Circuit Board Trace This quite inexpensive solution is advantageous if low profile network nodes are required. This kind of antenna typically shows poor performance which is different from node to node because of the tolerances inherent in the circuit board manufacturing process.

Wire or Metal Strip Antenna These elements can be placed as a component on the circuit board. They have improved performance compared to the circuit board traces, even though performance degradation due to coupling effects with the other components can not be excluded. Wire antennas seem to provide reasonable compromise between cost and efficiency.

Specialized Ceramic Antenna Components These elements are much more expensive than a wire antenna but they have the advantage of being smaller in size and can be robotically inserted and soldered. However, they are available only for the most widely used frequency bands (e.g. the 915 MHz and the 2450 MHz Industrial, Scientific and Medical (ISM) bands)

A comprehensive analysis of the problems related to the antenna selection can be found in Chap. 8 of [12].

The energy required to transmit (or receive) a single bit has often been chosen as metric for the efficiency of the transceiver. The data transmission (or reception) is one of the most energy expensive operations in a sensor node[3].

The frequency band used for the communication has to be chosen according to the application requirements and the existing regulatory norms. It has to be taken in account that for a simple antenna (like a monopole or a dipole) an increment of the central frequency corresponds, on one side, to the need of increasing the transmit power in order to cover the same distance and, on the other side, to the possibility of using shorter radiant element. In fact, the optimal size of a simple antenna is proportional

---

[2] E.g. the Chipcon CC1100 provides 4 different sleep states [16], which are associated with current consumptions ranging from 160 μA to 400 nA.

[3] E.g. 0.3 $\mu J/bit$ is the estimated efficiency of a Chipcon CC1100 transceiver computed on the basis of the information provided by the datasheet by considering a Medium Access Control (MAC) frames of 64 Bytes, a data rate of 250 kBuad and the transmission power of 0 dBm. This is the same amount of energy a microcontroller Amtel ATMega128L clocked at 8 MHz powered a 3V uses for ∼90 clock cycles.

to the wavelength of the carrier signal. Typically the transceivers used for WSNs operate in the ISM bands $433.05 - 434.79\ MHz$, $868 - 868.6\ MHz$, $902 - 928\ MHz$ (e.g. MICA Motes / ZigBee) or $2.400 - 2.500\, GHz$ (ZigBee) which permit the usage of monopole antennas, whose length varies from about 15cm for the first band to about 3.2cm for last band. Actually there are many research activities aimed at developing short range transceivers operating in the millimeter frequency band [17]. This solution is particularly interesting because it permits the integration of all sensor components, with the exception of the power supply, on a single chip.

Some transceivers offer different carrier frequencies (*channels*) in each band, which can be selected in order to alleviate congestion problems.

The bandwidth of the transmitted signal (the portion of radio spectrum around the carrier frequency, which contains useful information) together with the modulation and coding scheme determines the maximum data rate. In most cases the gross data rate ranges from few tens to some hundreds kilobits per second and is hence fairly less than in broadband wireless communication systems, but usually satisfactory for WSNs.

The adopted modulation scheme is typically a fairly simple on/off-keying, (ASK, FSK or similar), while the set of available coding schemes varies from model to model.

Some transceivers offer the possibility of directly controlling the transmission power which can be selected from a discrete number of different power levels. The maximum output power is generally given by the regulations. The modification of the output power can be used for *topology control* purposes [18].

Since many MAC protocols sense the channel before scheduling the activities, it is important that the receiver is able to provide the required information. Some receivers provide an indication of the intensity of the received signal. On the basis of this information, reported in the Radio Signal Strength Indicator (RSSI), it is possible to compute for example a rough estimation of the node's position (e.g. [19]).

3.1.3 The power supply

Since sensor nodes are supposed to run unattended and autonomous for a long time, the availability of appropriate energy sources is a crucial aspect for a WSN project. The mainstream strategy consists of using batteries as primary energy sources and possibly some *scavenging* device to replenish the consumed energy. The work of Shad Roundy et al. [20] gives a concise overview of the different solutions which might be adopted for both the batteries and the generators.

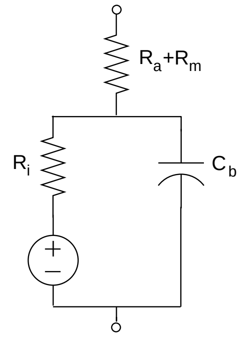
A battery is the most widely used power source. It could be a non rechargeable one (*primary* battery), or, in presence of a scavenging device, a rechargeable one (*secondary* battery).

A battery is nothing else than an electro-chemical storage of energy which is able to release this energy by doing electrical work. From a historical point of view, batteries have been the first kind of electricity source used by man and have been steadily improved over the years. The different families of batteries available today differ mainly in the materials used for electrodes and electrolytes.

Form a macroscopic point of view the different families are characterized by the different capability to store energy. The *energy density* (J/cm^3) is the metric used to measure this property. Table 3.2 reports the typical energy density values observed in commercial cells. Following the trend of having small sensor nodes it is obvious that batteries characterized by a higher energy density should be preferred in order to keep

Table 3.2: *Energy densities for various rechargeable and non rechargeable batteries [20]*

Non rechargeable			
Chemistry	Zinc-air	Lithium	Alkaline
Energy (J/cm^3)	3780	2880	1200
Rechargeable			
Chemistry	Lithium	NiMHd	NiCd
Energy (J/cm^3)	1080	860	650



R_m is the resistance of the metallic path through the cell including the terminals, electrodes and inter-connections.

R_a is the resistance of the electrochemical path including the electrolyte and the separator.

C_b is the capacitance of the parallel plates which form the electrodes of the cell.

R_i is the non-linear contact resistance between the plate or electrode and the electrolyte.

Figure 3.3: *Battery equivalent circuit*

the nodes small in size.

A more careful examination of the behavior of a battery suggests the existence of further selection criteria which are based on the electrical and chemical properties of the battery. Since the expected node life can span a long time interval, we want to consider the aging processes. Electrochemical reactions are responsible for the *self-discharge* effect. A battery loses part of the stored energy even though it is not connected to a load. The discharging rate depends on the temperature; in fact an increment of the temperature produces an acceleration of the chemical processes. The lithium batteries are quite stable from this point of view, their discharging rate[4] is less than 1% per year at 20°C and less than 2% at 30°C, while the zinc-air cells can not be considered for long lasting operations, their discharge rate of ~2% per year at 20°C increases rapidly with the temperature reaching 10% at 30°C and more than 50% at 50°C.

From the electrical point of view, a battery can be modeled with the equivalent circuit of fig. 3.3. When current flows through the cell there is a voltage drop across the internal resistance of the cell which decreases the terminal voltage of the cell during discharge and increases the voltage needed to charge the cell thus reducing its effective capacity as well as decreasing its charge/discharge efficiency. The *internal resistance* of the battery is influenced, among other things, by age, temperature and energy level. The manufacturers report an impedance between 10 Ω and 20 Ω for a fresh lithium coin cell (this value increases with time and discharge).

---

[4] The data have been obtained from the data sheets of the DURACELL products

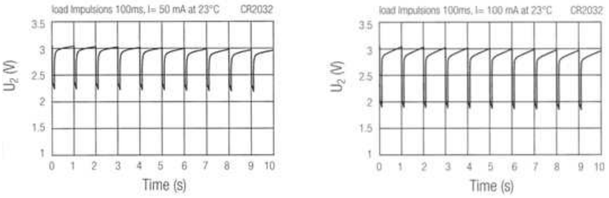


Figure 3.4: *Pulse discharge of lithium coin cell CR2032* [5]

The *maximum discharge current* depends, among others things, on the geometry of the electrodes and on the characteristics of the electrolytes. When a device tries to absorb more current than the given threshold, the voltage drops and when it falls below the minimum required in order to drive the circuits, the node might cease working. On one side the peak absorption has to be considered at the time of the battery selection. On the other side, in presence of absorption peaks, it is advisable to adopt an activity schedule which permits a partial recovery of the batteries. The described effect is represented by the graphs in Fig. 3.4.

In some cases, it is possible to directly connect the batteries to the load by assuming sufficient stability of the output voltage over the whole node life. Generally, however, because of the different characteristics of analog and digital circuits, and in order to keep the provided voltage stable, despite of the discharging of the batteries, a *voltage converter* is inserted between battery and load. These devices can be divided in the two families of the linear and the switching converter[6]. The former are down converters based on an active component (transistor) used as variable resistance, the latter are based on a reactive component, capacitor or inductor which is switched alternatively in and out. The energy stored in the reactive element is used to produce the output voltage, which can be higher or lower than the input voltage. Because of the switching activity, this converters are sources of Electromagnetic Compatibility (EMC) problems and require the adoption of a noise filter. Especially for the switching converters there is an almost constant power consumption (proportional to the maximum current capability of the converter) which is due to the converter itself.

In case of a bursty activation strategy of the computation and communication components of a sensor node, the current drainage due to the converter might thwart the improvement due to the optimization of the node activation strategy.

An effective strategy to reduce the current drainage of the converter, consists of permitting a *bursty* activation of this element decoupled from the operation scheduling of the other components. In such a case, the energy stored in the (large) capacitor of the EMC filter is used to feed the load as long as the converter is down. Each time the voltage reaches a lower threshold, the converter is turned on again only for the time required to recharge the capacitor and than it is turned off again.

3.2 Software architecture

At a first glance, one could think that the resource constraints impose the implementation of monolithic and dedicated software, which is responsible for the management

---

[5] From the product datasheet provided by the company RENATA batteries (*www.renata.com*)

[6] A more detailed discussion of the power management aspects might be found in Chap. 7 of [12]

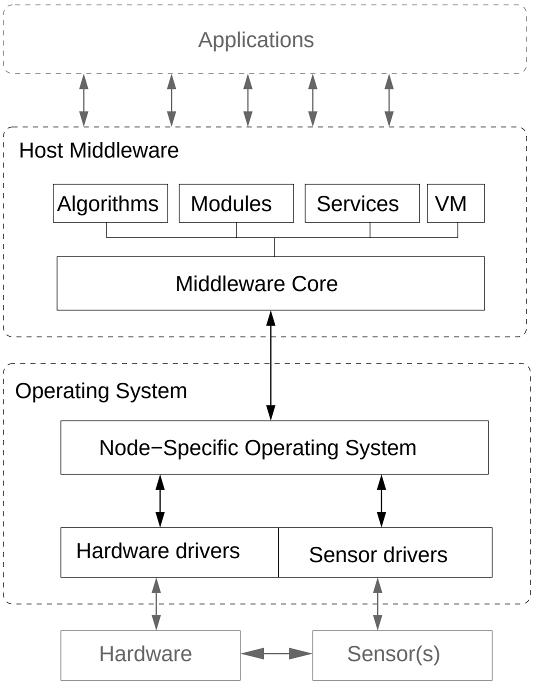


Figure 3.5: *Sensor node software architecture [21]*

of all node activities. Even though such a solution might lead to extremely efficient software, it has to be discouraged, since it is associated with high development costs and prone to failure.

Some research, in the field of the software engineering addressed the problem of the software architecture for sensor nodes [21–23]. Almost all the authors agree on the architecture shown in Fig. 3.5. This architecture uses a separation of functional blocks, in order to increase flexibility and enhance the scalability of sensor node software. The different layers/blocks offer increasing abstraction levels and allow a faster and easier modular implementation of new applications.

The Operating System (OS) layer has been divided in a *driver layer* which contains at least a sensor driver and the hardware drivers (e.g. transceiver driver, power management driver etc.) and a *node specific OS*. The OS handles device tasks, like initialization of hardware, memory management, scheduling and so forth.

The *host middleware* is a software layer devoted to the cooperation of distributed nodes. The four optional elements *Algorithms, Modules, Services, Virtual Machine (VM)* can be implemented and exchanged according to the node's task. *Modules* are additional components which increase the functionalities of the middleware (e.g. security modules). *Algorithms* describe the module behavior (e.g. the different encryption algorithms for a security module). *Services* contain the software components which are required in order to perform local and distributed services. The *VM* enables the execution of hardware independent code.

From the network perspective, the additional layer *Distributed Middleware* sits between the *Application* and the *Host Middleware.* Its purpose is the coordination of the services within the network, it supports the creation of network applications.

3.2.1 The Operating System

The traditional OSs, used in general purpose computing devices, are responsible for controlling and protecting the access to the resources and managing their allocation to different users as well as providing the support for concurrent execution of different processes and communication between these processes.

Considering the previously described characteristics of sensor nodes and sensor applications, it is clear that a sensor node can not support an OS providing all the described functionalities and, moreover, that they are all not required.

The research in the field of OSs for WSN is mainly aimed at developing a simple *execution environment* which is able to address event concurrency, provides some sort of interfaces towards the underlying hardware and provides some basic functionalities for common operations.

The simplest solution is based on the utilization of an appropriate programming model together with a well defined way to structure a protocol stack. This approach has been followed during the development of *TinyOS* [22], which represents actually one of the most widely adopted solutions.

The concurrency of different activities is one of the most important aspects which have to be addressed in an execution environment.

The well known process based approach, where the OS supports the concurrent execution of many processes on a single Central Processing Unit (CPU) is not suitable on a small weak system. In particular, mapping the single protocol functions or layers to individual processes would cause a high overhead in switching from one process to one other. Moreover, each process would require the allocation of its own stack space in the memory which is in conflict with the stringent memory constraints.

A simple and resource saving sequential, polling based, approach would also be useless because of the non negligible risk of missing important events.

The work of Hill et al. [22] did show a third way consisting in a *event-based* programming approach. Small pieces of code (*handlers*) react to the occurrence of new events (e.g. the availability of new measurements, the arrival of a data packet, etc.). The event handlers are able to interrupt every running code, except other event handlers. They simply store event-specific information in dedicated data structures. The execution time of each event handler is so short, that it runs to completion under all circumstances without noticeably disturbing the other code. The actual event processing is then carried out by other routines. In order to implement this event handling strategy, an event-based system has to distinguish between two different *contexts*: a first one for time critical handlers, which can not be interrupted, and a second one for the processing of normal code which can be interrupted. The proposed event-based solution requires a new approach to code generation which has many similarities to the code generation for parallel systems.

TinyOS, along with the programming language *nesC* [24], offers an implementation of an event-based system which partially conceals the complexity of the underlying programming paradigm and which permits a modular approach to the implementation of the node functionalities. A tiny scheduler and a graph of *components*, are the elements of a system description. The directed edges of the graphs represent the event flow. A component is a self-sufficient software element consisting of (Fig. 3.6):

a *frame* - fixed size and statically allocated memory space used to store all information required in order to operate on the specific component

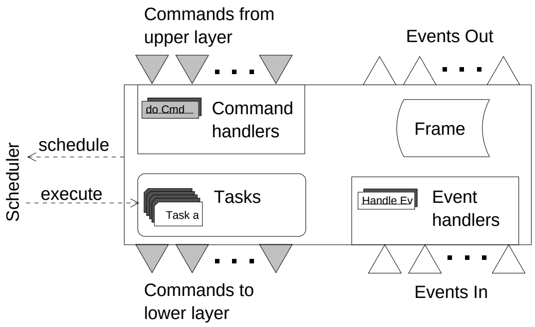


Figure 3.6: *TinyOS component's structure (adapted from [22])*

a set of *event handlers* - small preemptive Software (SW) elements which are invoked in order to deal, directly or indirectly with Hardware (HW) events. The execution time of an handler must be short and fixed

a set of *command handlers* - pieces of SW which handle the non blocking requests arriving from the upper layer component (*commands*), they act on the information stored on the frame and might schedule the execution of some tasks. A command must provide feedback to its caller in a finite time

a set of *tasks* - atomic code elements which run to completion, even though they can be preempted by events. They operate on the data stored in the frame, can issue an event to the upper layer, call a command to the lover layer and schedule new tasks.

The scheduler implements a simple First In First Out (FIFO) strategy and stores the information into a bounded size data structure. The scheduler is power aware and able to put the node to *Sleep* mode when no task is scheduled.

The combined use of commands and events permits the quasi-concurrent execution of actions which would block the system (because of the run-to-completion strategy). In order to do that, each action which requires an answer from an underlying module has to be split into two tasks. The first one starts a command to the underlying module and stores some data in the frame, the second one is scheduled when the corresponding answer arrives in the form of an event from the underlying level. This *split-phase programming* approach requires the definition of interfaces between the components which are defined with the help of some dedicated structures of the *nesC* language.

The output of the *nesC* compiler and linker is a monolithic, statically linked image of the system which can not be modified at the run-time.

In order to overcome this limitation, the *Contiki* [23] OS for WSN has been proposed. It implements an event-driven kernel and, only if a particular application explicitly requires it, an optional lightweight preemptive multithreading. The *Contiki* system consists of two parts: a *core*, which consists of kernel, libraries and program loader, and the *loaded programs*, which are loaded at run-time by the program loader. They can be either stored on the local EEPROM or can be retrieved from a remote storage by using the communication stack.

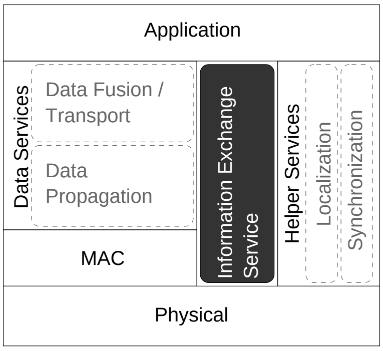


Figure 3.7: *A possible cross-layering enabled stack (adapted from [25])*

3.3 Protocol stack

3.3.1 An ideal solution

The traditional approach to communication protocol structuring is based on the layering concept: individual protocols are stacked on the top of each other, each layer using the function of the layer directly underneath. This approach proved to be advantageous in promoting modularity and reuse, in reducing the complexity and keeping the whole stack manageable. However, it is still unclear if a strictly layered approach is the most advantageous approach for the design of new protocols for WSNs.

In particular Kumar et al. [25] argue the inefficiency of the traditional paradigm and propose a new approach aimed at exploiting the possible benefits of *cross-layering*. Similar solutions have been proposed in other works (e.g. [26]).

The main idea is based on the observation that, on one side, some pieces of information are required at different levels of the protocol stack. The signal strength, measured at the physical layer, might e.g. be required by the network layer in order to select the forwarding nodes and, at the same time, by an application which estimates the node position. On the other side, the availability of information belonging to different protocol layers might allow a better utilization of the resources. The transport protocol can e.g. hold the Automatic Repeat-reQuest (ARQ) error correction down, as long as the physical layer observes the presence of disturbing signals. In such a way it is possible to save energy by reducing useless transmissions.

The introduction of cross-layering is, however, a complex process which is open to different kinds of mistake. The most common pitfalls are

absence of standard interfaces which might cause inefficiencies like the proliferation of the data elements

piecemeal evolution which leads to a spaghetti design of the protocol stacks and hence to a too complex, non verifiable and non maintainable structure

incomplete spread of the information which might cause the adoption of suboptimal strategies.

Both cited papers agree on the need of decoupling the *adaptability* requirements, which are cross-layer oriented, from the *modularity* requirements, which are functionality oriented. In order to achieve that they introduce a new module, which has been

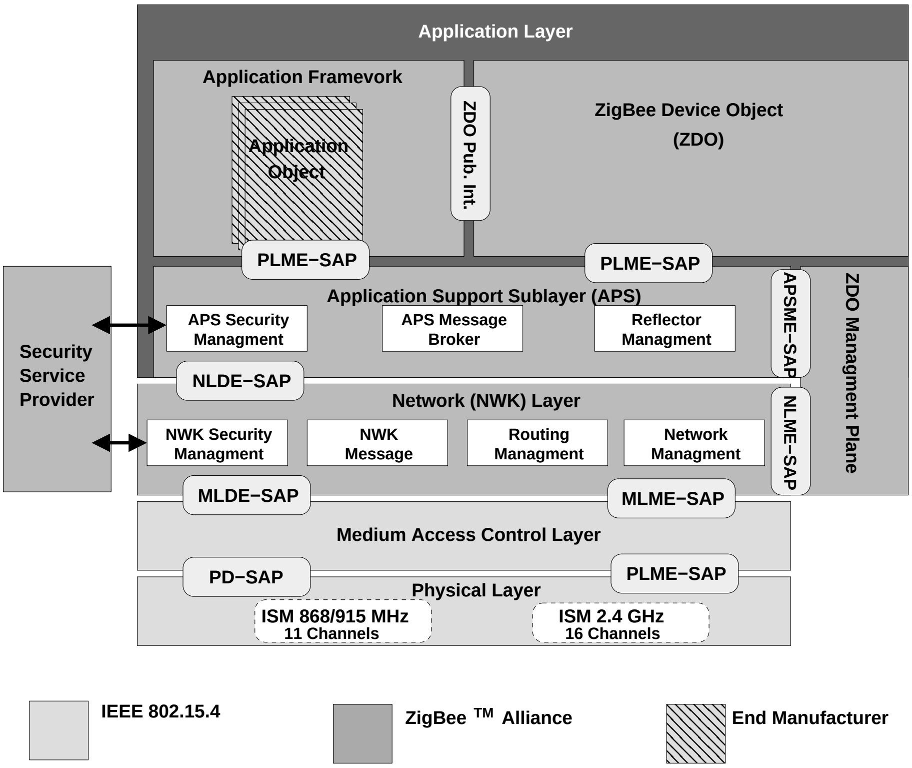


Figure 3.8: *The ZigBee protocol stack (from [27])*

called Information Exchange Service (IES) in [25]. The IES enables the cross-layering by offering a publish/subscribe interface and an event notification service. By concentrating the cross-layer management onto the IES, the protocol modules can concentrate on the own functionalities and preserve the modularity.

The resulting *sensor protocol stack*, having the structure shown in Fig. 3.7, is particularly suitable for the implementation on a event-driven OS like TinyOS. An event handler, in the module implementing the IES, reacts to a modification of a published event by notifying all their subscriber modules.

3.3.2 Standardization

Interoperability and standardization are requirements when a technology tries to move from the research labs to commercial applications. In the year 2000 two different standard groups, the *ZigBee Alliance* and the *IEEE 802 Working Group 15*, started to work on a protocol stack for low power, low data rate, wireless networks tailored to WSN and Personal Area Network (PAN) applications. The two groups joined their efforts the new IEEE task group 802.15.4 started in December of the same year with the definition of the Physical Layer (PHY) and MAC layer, while the ZigBee Alliance concentrated its efforts on the definition of the upper protocol layers. Fig. 3.8 gives a schematic representation of the results of this standardization process.

IEEE 802.15.4

Like all the IEEE 802 standard the IEEE 802.15.4 [28] encompasses only those layers up to and including portions of the Data Link Layer (DLL).

Two PHY options have been defined in order to operate in the two ISM bands 868/915 MHz (Europe/USA) and 2.4 GHz. Both PHY variants are based on a Direct Sequence Spread Spectrum (DSSS) technology and both use the same frame structure.

The two PHYs differ in the transmission rate. Systems operating in the 2.4 GHz range provide a gross data rate of 250 Kbps while systems operating in the other band provide 20 Kbps or 40 Kbps, respectively. The higher performance in the 2.4 GHz band is mainly due to the adopted modulation scheme.

When selecting the operating band, not only the data rate has to be considered but also the costs[7], the presence of interferences[8] and sensitivity / coverage aspects[9].

Since the devices have to operate in an environment containing multiple types of wireless equipment, which might cause involuntary interference, the standard is open to the implementation of a dynamic channel selection mechanism. However, the definition of the specific selection algorithm is referred to the upper protocol layers.

The basic structure used for the data exchange at the PHY layer is the PHY Packet Data Unit (PPDU). This is a relatively short piece of information which can not be longer than 133 Bytes[10]. The choice of using a short PPDU has been probably made in order to reduce the packet error rate in the considered noisy channel. It should not be a problem for the majority of the WSN applications which are expected to exchange only few data.

The current standard specifies for the receiver a sensitivity threshold of -85 dBm for the 2.4 GHz devices and -95 dBm for the other band. The transmitters have to be able to transmit at least at 0 dBm and are expected to cover 10-20 m in an indoor environment.

The MAC layer is located on the top of the PHY. It communicates with other MAC instances by exchanging MAC Packet Data Units (MPDUs) which, due to the simple and flexible structure, can accommodate the needs of the different upper layer protocols. The basic structure of a MPDU consists of a MAC Header (MHR), a MAC Service Data Unit (MSDU), which carries the data provided by the upper layer, and a footer which consists of the Frame Check Sequence (FCS)[11].

The MHR starts with a 16 bits frame control field which define among other things the frame type (*Beacon, Acknowledge, Data, Control*) and the addressing. An 8 bit field used to carry a unique frame sequence number follows the frame control field. The last field which might range between 0 and 20 Bytes contains the addressing information. In order to save space it is possible to use short 8 bit device addresses instead of the 48 bit IEEE-addresses.

The channel access happens in a contention based fashion by using a (*slotted or unslotted*) Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) proce-

---

[7] 2.4 GHz equipment is normally cheaper.

[8] The 2.4 GHz band is getting crowded, the equipment has to coexist with WLAN and Bluetooth equipment.

[9] Thanks to the lower data rate the devices operating in the 868/915 MHz band are characterized by higher sensitivity and correspondingly by a larger coverage area.

[10] A PPDU consists of a *Synchronization Header* (32 bits preamble and 8 bits Start-of-Packet delimiter), a *PHY Header*(8 bits) and a *PHY Service Data Unit*(*PSDU*, at most 127 Bytes).

[11] 16 bit Cyclic Redundancy Check (CRC) computed over the whole MPDU.

dure. When low delay is required, the network might operate in a superframe mode. A node, *PAN coordinator*, forces a superframe structure by transmitting superframe beacons at regular intervals. The interval length can be selected between 15 ms and 245 s. The beacon interval is divided into 16 equal length slots. The channel access is contention based, using the slotted variant of CSMA-CA. However, the coordinator might assign time slots to a single device, which requires low latency access. These assigned *guaranteed time slots* are located at the end of the superframe and form a *Contention Free Period (CFP)*. The duration of the CFP is communicated by the coordinator with each beacon. The nodes which do not have reserved time slot must compete for the bandwidth during the contention period and the winner has to release the channel before the start of the contention free period.

The MPDU sender can indicate whether or not an acknowledge is expected, by setting a flag in the MHR. Should that be the case, the successful reception and verification of a data or control frame is confirmed with an acknowledge frame sent immediately after the successful validation. Beacon and acknowledge frames are never acknowledged.

The MAC layer offers two Service Access Points (SAPs) to the upper layer: the *MAC Common Part sublayer (MCPS-SAP)*, used to communicate with the MAC data service and the *MAC layer management entry (MLME-SAP)*, in order to interact with the management services.

From the point of view of the network topology the 802.15.4 standard allows the creation of two basic topologies: *Star* and *Peer-to-Peer*. In the former case all the nodes are directly connected to a central element which is the only directly reachable communication partner. The latter topology considers the direct communication with all neighbor nodes.

The same standard allows the implementation of a subset of the whole protocol stack in order to accommodate the requirements of really weak devices. These *Reduced Function Devices* do not have routing capabilities and have to communicate through a device which implements the complete standard.

ZigBee

The ZigBee standard [27] defines the upper layers of a WSN protocol stack developed on top of the IEEE 802.15.4 protocol. Adjacent to the MAC Layer, the *Network Layer* ensures the proper operation of the underlying layer and provides an interface to the application layer. It supports star, tree and mesh topologies and defines three different kinds of logical device types: *Coordinator, Router* and *End Device* which are different in their functionalities.

While the *End Devices* are only able to join (or leave) the network and to send and receive packets, the nodes of the other two types participate in the routing process and in the management of the 16-bit network addresses. *Coordinators* are the only elements able to start a ZigBee network. They scan the different channels in order to select the best one. Afterwards they assign a logical network identifier (PAN-ID), which will be applied to all devices, which join the network.

The Network Layer provides also security to the network by ensuring authenticity and confidentiality for each transmission.

The routing can be based on different alternative schemes. The most interesting are the well known Ad-hoc On Demand Distance Vector (AODV), which is tailored to highly dynamic network structures, and a new and simple Hierarchical Routing Algorithm (HERA), which leverages the parent-child relationship established by the 802.15.4 topology formation procedure. The behavior of the two routing scheme has been compared in [29].

The *Application Layer* consists of many interacting sublayers. The *Application Support Sublayer* provides a reliable data transport service and is responsible, among other things, for the management of the binding tables, fragmentation and reassembly of packets, address mapping between the 64-bits IEEE extended addresses and the 16-bits network addresses, and group address definition and management. A binding is a map which correlates a source tuple (source address, source endpoint) to one or more destination tuples.

As example a network contains two temperature sensors from two different manufacturers which can be accessed through the endpoints 0x23 and 0x77 respectively. A controller discovers the two sensors and adds two entries to the binding table, one for each node, in order to be able to send a request to both sensors.

The *Application Framework* is an execution environment for *Application Objects*. These Application Objects are located on top of the application layer. They are defined by the device manufacturer and actually implement the application. Each Application Object is addressed by its corresponding endpoint. An Application Object is characterized by its ZigBee application profile which provides a description of the logical components and their interfaces. The usage of profiles provides the interoperability between devices of different manufacturers by using a common language for the data exchange and by defining a common set of processing actions.

The endpoint 0 correspond to the *ZigBee Device Object (ZDO)* which is responsible for the overall device management. The ZDO is specifically responsible for defining the operation mode of the device and for the device and service discovery. Device discovery allows for ad-hoc network operations and network self healing.

IPv6 over Low-Power Wireless PAN (6LoWPAN)

In parallel to the industry driven standardization process carried out by ZigBee, the network community started to show its interest in the new low power MAC protocol of the IEEE 802.15 family. In the summer 2007 the first Request for Comments (RFC) entitled *"6LoWPANs: Overview, Assumptions, Problem Statement and Goal"* [30] appeared.

The document is the "manifesto" of an IETF working group, which is aimed at permitting an almost transparent integration of the sensor nodes into IP-based networks with the purpose of promoting a technology which is well-known, proven to be working and based on a freely available specifications. Since the technology trend goes towards the adoption of the IP version 6 (IPv6) protocol, the working group suggests using it. The authors argue that some mechanisms like autoconfiguration and management can be profitably ported to the sensor networks, once these are able to propagate IP-packets.

However, because of some peculiarities of the lower protocols used in PANs, a straightforward implementation of IPv6 is infeasible and some adaptation mechanisms are required. As example the maximal length of an IEEE 802.15.4 frame is much smaller than with the minimum Maximum Transmit Unit (MTU) specified for the IPv6 packets.

The Standards Track RFC 4944 [31] published by the same working group proposes possible solutions to the highlighted problems. In particular, it defines an *adaptation layer*, which provides, among other things, *fragmentation capabilities*, *IP-header compression* and *stateless address autoconfiguration*.

The 6LoWPAN standard does not define any specific routing protocol, whose standardization is left to another group.

One of the benefits of the proposed approach is that it provides an Application Programming Interface (API) to the developer which reflect the well known IP networking API.

Chapter 4

An indoor WSN for metering applications

A stated before, building automation is one of the most promising areas for the commercial usage of Wireless Sensor Networks (WSNs). The possible applications span from surveillance to the monitoring of environmental parameters. In this wide range of possible applications, we focus our attention on a particular class of networks which, despite the simplicity of the provided service and the straightforward constraint definition, is a source of challenging problems.

A Wireless Metering Network (WMN) is a WSN which provides periodical readings of some physical meter/counter. The network size ranges from just a few nodes in residential buildings up to 1000 nodes in large complexes. The number of nodes can be considered almost proportional to the number of rooms for the respective building.

Such a network is typically owned and operated by a service provider, which is not identical with the owner of the building. It will typically be deployed in an already existing building such that mains operation is much too costly because a large number of power outlets would have to be retrofitted for that purpose. Therefore the nodes have to be autonomous, i.e. battery powered operation is necessary despite the fact that the network is operated in an indoor environment.

A clear requirement is that the network is capable of long lasting unattended operation. In fact, the physical access to the sensor nodes after the deployment is limited since each time arrangements have to be made with the possibly numerous tenants. Also, physical access to the sensor nodes is a significant cost factor because a technician has to be sent there.

All the following assumptions about boundary constraints, HW-components, parameter choices and protocols are directly derived from the specifications which emerged during a project done in cooperation with an industry partner and aimed at developing a new commercial multi hop long lasting wireless sensor network for metering applications.

The original contributions of the author to the success of this project include the improvements of the preexisting Medium Access Control (MAC) protocol (see section 4.3.2) and the development of some new protocols which provide time synchronization (see chapter 8), network initialization (see chapter 6) and continuous network optimization (see chapter 7) capabilities.

4.1 An evolutionary tale

Metering applications are not new: electricity, water and heating consumption are constantly measured with dedicated equipment and the data are used for billing purposes.

The older meters were simple standalone systems equipped with some sort of indicator and required human intervention in order to record the collected data. The newer meters tend to incorporate some sort of short range communication device in order to automate, at least partially, the reading process. It is not uncommon to observe *walk-by* solutions where the human operator, who carries a reading device, passes by the metering appliances and the reading device collect the data from the meters autonomously.

In order to reduce the costs, a further evolution tries to eliminate the need for the presence of a human operator. The first approach in this direction consists in building a simple wireless network characterized by a star topology and based on the direct communication of each meter with a central unit (*concentrator*). The concentrator, which is either mains operated or equipped with much larger batteries, collects the data from the single nodes and periodically propagates these towards the data-processing center of the metering network operator by using some well known telecommunication technology like GSM or GPRS. If the size of an installation prevents the direct communication of all nodes with a single concentrator, additional concentrators are activated and interconnected in order to build a kind of backbone network (see Fig. 1.1(b)).

The major drawbacks of this approach consists in having two different kinds of nodes, meters and concentrators, the latter of which are rather expensive devices, and in the existence of "*single points of failure*"(the concentrators).

A logical evolution of this architecture consists in building a self configuring network of identical (at least from the point of view of energy and communication resources) and autonomously operating nodes. All meters have to be able to propagate both the own data and the data received from the neighbor nodes towards the concentrator. The concentrator, like in the previous case, transmits the information using another communication technology.

This multihopping approach is aimed at improving the network resiliency and at the same time at reducing the network management costs.

4.2 Boundary constraints

The business plan imposes a lower limit to the expected network life and an upper limit to the production cost for each single node. Practical considerations suggest the adoption of off-the-shelf component for its implementation. The sum of these constraints reduces the degrees of freedom for the project development.

Since physical access to the sensor nodes after the deployment is limited, the maintenance operations (battery or node substitution) should fit into the scheduling for the building maintenance, which is typically characterized by ten year periods.

The batteries represent a non negligible component of the initial cost, at the same time they also contribute to the determination of the node size. A lithium coin cell is the power supply typically adopted for metering applications and represents a good trade-off between cost, self discharge and energy density.

Once the hardware components have been chosen, it is possible to obtain a rough estimation of the energy budget and consequently of the energy saving strategies the nodes have to implement. As example we consider nodes powered by a battery having a capacity of 1000 mAh. The half of the available energy can be used to drive the transceiver, which absorbs about 16 mA during the active phase and about 400 nA in *Sleep* mode. The transceiver is assumed to provide a gross data rate of 112 kBuad. The processing unit consists of an 16 bit microcontroller equipped with 2 KBytes RAM and 32 KBytes ROM. The controller, together with the sensor and the rest of the electronics, is responsible of the consumption of the other half of the available energy.

The expected network life is ten years[1], which leads to an upper limit for the transceiver's duty-cycle of about $2.5 \cdot 10^{-4}$.

The coexistence of analog and digital circuits which require different input voltages imposes the utilization of a voltage converter. The characteristics of the batteries require the utilization of the capacitor of the voltage converter as secondary energy source (see 3.1.3) and upper bound the duration of a transceiver's activity period to 20 ms approximately. Moreover, in order to allow a partial recovery of the batteries, the mean interval between two consecutive activity periods has to be at least 6 s.

A first simple evaluation of the reported values suggests the existence of two major problems. First, the low duty cycle and the relatively short activity period mandate the adoption of some form of clock synchronization in order to reduce the length of the guard period[2]. Second, the low average data rate requires an efficient utilization of the communication channel. In fact, the data rate of a hypothetical bottleneck link near to the concentrator is upperbounded by 329 KBytes of application data per day. Hence, the average data rate between each node and the concentrator will be lower than a few hundreds bytes per day.

4.3 The MAC layer protocol

The development of an efficient MAC protocol is the first step in order to address the previously described issues.

4.3.1 Considerations on the properties of a MAC protocol

MAC layer protocol design is a research area more than 30 years old. The older works address the performance requirements of throughput efficiency, stability, fairness, low access delay, low transmission delay as well as low overhead. The overhead at the MAC layer is the sum of many components, which might include the per–packet overhead (MAC headers and trailers), the packet retransmission overhead due to collisions, and the additional overhead due to exchange of management frames.

A first taxonomy, based on the strategy used to access the channel, allows identifying the three main families:

Fixed assignment protocols The resources are divided between the participating nodes in such a way that each node can use its resources exclusively and without the risk of collisions. The assignment is done in a long-term fashion (for an interval much longer than the duration of a data burst). In order to react to

---

[1] The importance of meeting the requirement suggests to consider a 10%-20% longer interval.

[2] A receiver node activates the transceiver in advance with respect to the expected data arrival time in order to accommodate for possible synchronization errors. The *guard period* is the interval between the transceiver activation and the expected data reception time.

topology changes, there are some dedicated protocols aimed at the renegotiation of the resource allocation. Typical examples for this class of protocols are Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA) and Code Division Multiple Access (CDMA).

Demand assignment protocols In this case the exclusive allocation of resources to a node is done on a short-term basis (typically the duration of a data burst). The management of the resource allocation can be either centralized or distributed. In both cases a node sends out a request for the resource allocation to a management node which, in case of a successful allocation, responds with a confirmation message containing the description of the allocated resources. The deallocation might be implicit, which means the management node observes the channel and, upon detection of unused allocated resources, removes the reservation. The transmission of the reservation requests is often done by using a contention based random access protocol.

Random access protocols In this case each node tries to access the channel without explicit coordination with the surrounding nodes. The access protocol operates in a fully distributed manner. Often, the algorithms incorporate some random element in order to stochastically reduce the collision probability.

One of the oldest and best studied representatives of this class of protocols is the ALOHA protocol, which had been developed at the University of Hawaii some decades ago [32].

A node which implements the simplest unslotted variant of ALOHA starts a transmission as soon as the data arrives from the upper layer –regardless of the activity of the other nodes. When the data is received correctly, the receiver acknowledges the received frame by replying with a specific packet. The absence of an acknowledgment frame indicates a packet loss and in this case the transmitter reschedules the packet transmission. Clearly the absolute absence of any coordination causes inefficient channel utilization. It is known that the efficiency (the number of successfully transferred data per time unit normalized to the nominal data rate) of a data transmission based on the ALOHA mechanism is upperbounded by 0.18.

All the CSMA based protocols belong to the same family of random access protocols. However in this case the nodes try not to disturb ongoing transmission.

Before accessing the channel, a node observes the medium (carrier sense) and starts the transmission only if the channel is idle. Otherwise it delays the transmission for an amount of time dependent on the specific algorithm.

Since the physical layer directly influences the performance of the MAC protocol, the peculiarities of the radio channel have to be taken into account during the development of a MAC protocol for wireless networks. As example some wireless protocols (e.g. IEEE 802.11) address explicitly the existence of hidden nodes by implementing a virtual carrier sensing based on the exchange of RCS/CTS packets.

There is a wide literature on the MAC protocols for WSNs. An overview can be found in [4, 6, 12, 33]. All the authors agree that the implementation on wireless sensors imposes some additional constraints. So the low energy consumption becomes an additional performance parameter and at the same time, the implementation on weak devices imposes additional requirements for low complexity operations.

Among other things, the energy consumption might be reduced by adopting a *low duty cycle* operation strategy. The main idea is to reduce the time a transceiver spends in the *Idle* mode. In an ideal case the node leaves the *Sleep* state only for the time required to send or receive a frame.

This ideal case can be approached by using a *wakeup radio* system [34]. This approach is based on the observation that detecting the presence of a radio signal requires much less energy than decoding it. A wakeup receiver is a simple circuit, typically implemented in a dedicated piece of hardware, which generates a trigger signal each time a proper radio signal is detected. This trigger signal causes the activation of the transceiver in order to receive the following data transmission. Each station transmits a wakeup signal before sending the data in order to make the receiver aware of the subsequent network activities.

Much more common solutions are based on *periodic wakeup* schemes. In this case the node activates the transceiver periodically and only for relatively a short period. The sender knows the receiver's activation strategy and is, at least weakly, synchronized with it and sends the data out during an active period of the intended receiver node.

Because of the synchronization errors and the inaccuracy of the local clocks, the nodes have to be activated in advance with respect to the scheduled activation period. Since this guard period is a source of useless energy expenses it has to be made as short as possible.

By choosing a small duty cycle, the transceiver is in *Sleep* mode most of the time, avoiding the energy consumption due to idle listening. The duty cycle has to be chosen small enough in order to keep the energy savings bigger than the additional expenses due to the *start-up* costs.

It has to be noted that an extension of the sleep period causes an increment of the per-hop delay. In fact each node has to wait on average half a sleep period before propagating the data. Moreover, the adoption of a low duty cycle concentrates the network activities on small time windows, which might cause significant competition in heavily loaded networks.

4.3.2 A low energy / low duty cycle MAC

With respect to the development of a new project it is natural to check when some standard solution will be able to fulfill the requirements. Nowadays, the IEEE 802.15.4 [28] is the unique standard MAC protocol for short range low power / low data rate communication (see 3.3.2).

Unfortunately, this protocol does not fit the operational constraints well. In particular the contention based Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) access mechanism does not permit an efficient channel utilization if the offered load approaches the channel capacity which might happen in large networks. In fact, the collision probability increases with the number of competing nodes, and approaches 30% if the offered load is about 50% of the cannel capability [35]. The low latency, contention free and reservation based access mechanism which requires the existence of coordinator nodes would be useful in order to meet the requirements. Unfortunely this optional part of the standard has been mostly neglected. Moreover, in a dense network the probability of beacon loss due to interference increases quickly with the number of coordinator nodes [36] and might have a negative impact on the performances of large networks.

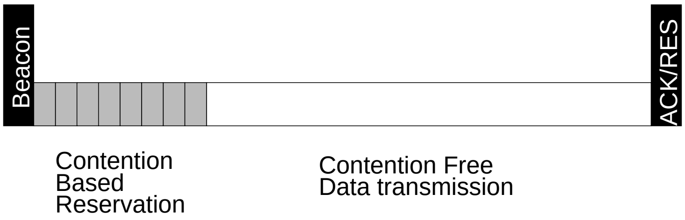

Beacon
ACK/RES
Contention Based Reservation
Contention Free Data transmission

Figure 4.1: *MAC protocol: channel access strategy*

Once the need of a new MAC protocol, specifically tailored for the considered metering application, had been highlighted, it appeared natural to start the development on the base of the pre-existent and well tested protocol, which had been used for the communication between concentrator nodes in the old network. This protocol was capable of extremely low duty cycle operation and *on-demand* resources allocation in a fully distributed manner. The simple modification of the channel reservation procedure was the only modification needed, to improve the channel utilization in a dense network and hence to meet the previously described requirements.

In the original protocol, the channel access is carried out according to a fixed scheme which combines contention based and reservation based channel access (Fig. 4.1) into a single activity period. A receiver node periodically (each T_{beacon}) turns on the transceiver and signals its activation by sending out a beacon signal. The beacon signal carries – among other things – the number (N_{AP}) of the actual activity period, which varies between zero and $N_{MAX_{AP}} - 1$ where $N_{MAX_{AP}}$ is the maximum number of access periods in a *superframe* structure. The following chapter 8 shows how the periodicity of the beacon generation might be conveniently leveraged in order to provide at least a local clock synchronization for all nodes.

A contention period of fixed length starts just after the beacon. During this interval, which is divided into a fixed number of slots, the nodes try to send their reservation requests by using a random access mechanism derived from the slotted ALOHA principle.

The subsequent contention free period is reserved for the transmission of a single frame from a neighbor node, which got the permission to transmit during a previous activity period, to the beacon sender.

The ACK signal following the contention free period contains the acknowledgment for the received frame along with a list of reservations for the winners of the initial contention phase.

If no message is received during the contention free period, the receiver removes the reservation for the corresponding access period.

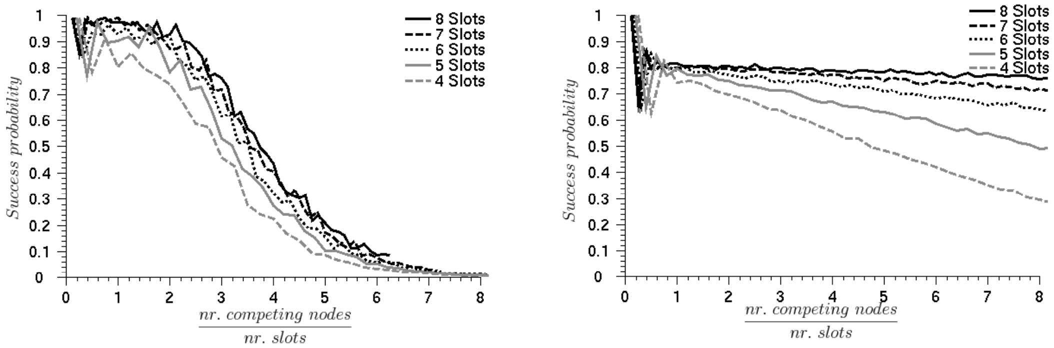
The total duration of the data exchange is upperbounded by the maximum length of an activity period.

It has to be noted that because of the structure of the MAC protocol, the reception of a beacon signal is a **mandatory precondition** for the data transmission.

It is worth spending a few words for the description of the access strategy used during the contention period. Ignoring the presence of capture effects[3], a reservation can be carried out successfully only if the corresponding requests do not collide with other signals at the receiver. If each competing node simply draws the slot it uses to

---

[3] The strongest of many interfering signals is correctly decoded, in spite of the collision event.



(a) Open Loop

(b) Closed Loop

Figure 4.2: *Dependency of the success probability on the offered load, during the contention based phase*

send the reservation request randomly, the probability that at least one of the requests does not collide is a non increasing function of the offered load[4]. It starts close one if the load is lower than 1 and falls rapidly below 10% if the load becomes bigger than 6. This behavior can be observed in Fig. 4.2(a) which shows the results collected for different slot numbers by using a Monte-Carlo simulation.

The reason for this behavior can clearly be identified to be the inability of the channel access algorithm to adapt itself to the load conditions because there is no feedback mechanism. Direct consequences of the low success probability are increasing channel access delays and, at the same time, waste of energy due to useless transmissions.

In order to make the channel access algorithm aware of the actual load condition and hence to solve the highlighted problem, the author proposed a solution which is based on the simple observation that the detection of collisions can be conveniently carried out at the receiving node. This node is, in fact, able to detect the presence of a radio signal during each slot and to identify the presence of corrupted data (collisions). The proposed improvement consists in letting the receiver adjust the channel access probability of the competing nodes.

A short 3 bit field (CH_ACC), allocated in the beacon frame, provides the beacon receivers (a superset of the competing nodes) with a feedback signal. The CH_ACC value is set by the beacon sender according to the observations done during the previous contention phase. All the competing nodes, have to evaluate the CH_ACC value and to defer the generation of the reservation request to a following activity period with probability $(1 - 2^{-CH\_ACC})$.

At the beginning CH_ACC is set to 0. Each time the number of slots containing corrupted data is higher than the number of unused slots the CH_ACC is incremented by one unit and it is decremented if the number of free slots exceeds the number of collisions.

The effectiveness of the proposed solution has been analyzed by simulation. The curves in Fig 4.2(b) show that this simple feedback strategy increases the success probability during the contention phase significantly and, hence, that it has a positive influence on the mean access delay and the energy expenses.

---

[4] The number of competing transmissions normalized to number of slots.

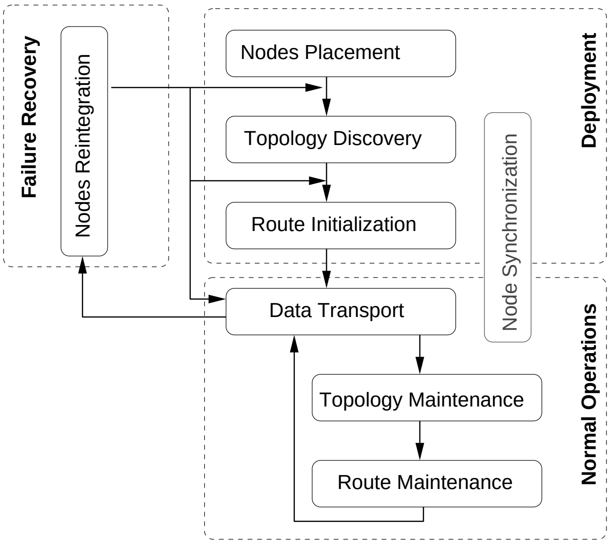


Figure 4.3: *Phases of the network life*

Moreover, the almost linear decrement of the success probability of the single request in dependence of the load suggests that the proposed mechanism achieves almost constant channel utilization.

4.4 The life-cycle of a Wireless Metering Network

The life of a sensor network can be ideally divided into the three main phases *deployment, normal operation* and *failure recovery.* Each phase is characterized by the execution of specific tasks. With respect to the special class of the WMNs, Fig. 4.3 gives a schematic representation of the interaction between the different mechanisms.

The following description tries to highlight the characteristics of the different phases and to mark the differences between the WMN case and the commonly accepted assumptions for the WSNs.

Deployment This phase includes all the steps which are necessary in order to transform a set of nodes into a functioning network. *Nodes placement and activation, communication setup and routing activation* are steps belonging to this phase. While the original idea of WSN considers the node placement as a random process, the installation of the meter nodes is a supervised process where a technician individually installs each element at a location predetermined by the type of service to be provided.

The time required to select the neighbor nodes and to set up the routing will be of secondary importance if the installation process is unattended. On the contrary, in the case of a supervised installation process, an additional visit of the technician to check if the network is fully operational would be a significant cost factor. This results in the requirement that the self-configuration of the WMN has to be finalized as quickly as possible after the deployment of the last sensor node such that an immediate verification is possible - several hours until convergence are not tolerable.

Even though, we assume that the service personnel has only limited knowledge of networking and hence the nodes have to be capable of self configuration. It is also realistic to assume that some external device might assist the nodes during the deployment phase by, e.g., providing a list of the already active nodes.

The chapter 6 is devoted to the analysis of the topology discovery process and the initialization of the routing process. During these two steps, each node selects the subset of its neighbor nodes which might be used to propagate the data toward the network concentrator. The resulting network structure is a tree rooted at the concentrator with redundant links in order to improve the robustness.

Normal operation Hopefully a network spends most of its life in this phase. All the nodes are able to communicate with the concentrator (sink), they collect measurements and propagate the data toward the sink.

A continuous adaptation/optimization process adapts network topology and routing to the changing propagation conditions. Even though neither the nodes nor the concentrator change their position during the whole network life, modifications of the signal propagation environment can be viewed as a sort of mobility. The integration of new nodes is also part of this adaptation process.

In the specific metering application considered each node collects a new value periodically, at least once a day. The measurements taken by the same node follow a non decreasing function of the time. This permits the implementation of simple data compression algorithms (e.g. in presence of multiple measurements generated by the same node, only the most recent has to be propagated).

Failure recovery In case of long lasting interferences or malfunctioning nodes, some network elements might lose the ability to communicate with the concentrator. In this case the nodes have to cooperate in order to restore the network integrity. The proposed failover procedure is described in chapter 7

The node synchronization process has to be performed in all phases. Because of the extremely low duty cycle and the limited duration of an activity period, it is essential in order to assure the proper functioning of the MAC protocol. Chapter 8 proposes a possible energy efficient solution for the synchronization problem.

The boundaries between the three phases are blurred and it is difficult to clearly identify an event which can be used as trigger in order to detect the transition from one phase to the other one. Therefore, it is advantageous to develop algorithms which can operate continuously during the whole network life, instead of developing dedicated algorithms for the different phases.

Chapter 5

A simulation model for Indoor WSNs

Simulation is one of the possible options and probably the most widely adopted for the study of Wireless Sensor Networks (WSNs). The reasons are mainly the good trade-off between complexity and generality of the results and the possibility of analyzing the network behavior over a wide range of conditions which can not easily be reproduced in a testbed. This is, actually, the approach the author chose for the analysis of the network behavior.

After a short description of the known alternative approaches to the WSN study, the rest of the chapter is aimed at introducing our propagation model especially developed for the peculiarities of the indoor environment, and at illustrating the overall simulation framework.

5.1 Different approaches to the WSN study

In parallel to the lack of common definitions for WSNs the absence of a uniform and widely accepted approach to the study of this class of networks is also noticeable. The different research groups tried to adapt the methodology used in previous projects and this approach led to results which are not directly comparable.

The following Table 5.1, which has been presented in a recent workshop, gives a short overview of the different approaches. Starting from the theoretical approach and moving to the implementation on a testbed, the results are obtained at the price of increasing effort while the generality of the results follows an opposite trend.

5.1.1 Implementation

Looking at the history of the WSN research, it is possible to see that the earlier works had been carried out mainly by research groups active in the field of HW development. One of the first examples is the “Smart Dust” project [38] at the Electronics Research Laboratory of the University of California. This research was mainly focused on the development of hardware platforms for wireless sensor networks. The resulting availability of HW elements permitted since the beginning the study of the network behavior in testbeds and small real installations. An overview of some interesting real WSN installations can be found in a paper of Römer and Friedmann [8].

Table 5.1: *Different approaches to the WSN study (adapted from [37])*

Class	**Analysis**	**Propagation model**	**Node distribution**	**Popularity**
Implementation	Testbed	Reality	*Reality(?)*	low
Heuristic	Simulation	UDG, SINR	Random, grid,...	high
Scaling law	Theorem/ Proof	SINR and more	Random, grid	medium-low
Algorithm	Theorem/ Proof	UDG and more	Worst-case (ad-hoc)	low

Probably the most famous one is the sensor network for wildlife monitoring, which had been installed at the Great Duck Island, a wildlife reserve. The network consisting of about 100 nodes powered with batteries and solar cells, showed the feasibility of long lasting big installations in a relatively inhospitable environment.

Even though the real implementation provided a lot of interesting information on the behavior of real WSNs and highlighted some practical issues, the observations done with these lack generality. Also, especially because of the limited size of the networks, they do not provide information on the scaling capability of the implemented mechanisms.

Moreover, HW costs and development time are limiting factors for this approach and the results are usually “too specific”.

5.1.2 Theoretical approach

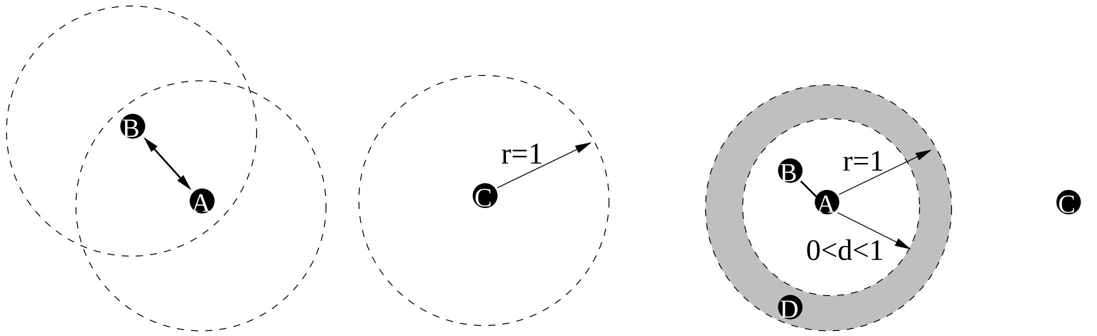
The theoretical approach has been used especially at the beginning of the WSN studies. It is particularly appealing because it permits to obtain some general results at reasonable costs and quicker than the other methodology.

Problem tractability considerations impose the adoption of strongly simplified models of the network elements and their behavior. As example, the most widely used propagation model is the Unit Disk Graph (UDG) which has been introduced, at least for the WSN study, in the work of Clark et al. [39]. It considers the communication area of a node being a disk centered at the node and with an unitary radius. According to it, two nodes are able to communicate as long as the distance between them is shorter than the disk radius (Fig. 5.1(a)). Clearly, the communication area of a real node does not have a circular symmetry and depends among other things on the node orientation, the presence of obstacles and the presence of fast fading effects[1]. However, it has been successfully used e.g. to show the complexity of some problems on the radio coverage graph. This is a graph where the edges represent the ability of direct communication between neighboring nodes.

Some authors [41] introduced the concept of *Quasi UDG* which permits to model the non ideal shape of the radio coverage area. According to this model the connectivity is certain in a disk of radius $d <= 1$, non existent outside the disk of radius 1 and undefined in the ring at distances between d and 1 from the node (Fig. 5.1(b)).

---

[1] The empirical characterization of the coverage area of a ZigBee sensor node can be found in [40].



(a) Unit Disk Graph model: A communicates directly with B but not with C

(b) Pseudo UDG model: the existence of a direct communication link between A and D is unknown

Figure 5.1: *Propagation models used for the theoretical analysis of WSN algorithms*

In both cases (UDG and Pseudo UDG) the resulting graph is planar, which is a quite strong simplification with respect to a real network.

In most cases a 2-D uniform distribution is used to model the node placement, this is in accord with the original idea of having the nodes dropped randomly on the observation area.

The theoretical approach has been successfully used to provide results in form of scaling laws and analysis of the overall behavior of the different algorithms.

Some noteworthy results obtained with this approach are e.g. the conditions for the network connectivity described in the work of Xue and Kumar [42] and the optimal synchronization algorithm proposed by Hu and Servetto [43].

The high abstraction level which is required by the theoretical approach, represents one of the major limitations of this methodology. The results in [43], e.g., apply to networks with a infinite number of nodes with infinitesimal communication range deployed on a finite area all conditions which can not be reproduced with real systems.

5.1.3 Heuristic approach (Simulation)

As reported in table 5.1 the heuristic simulation based approach is widely used to study WSNs. This approach is characterized by many appealing characteristics, e.g., it allows to investigate the behavior of huge networks and to analyze quickly the network behavior in dependence on parameter and algorithm modifications. The model development time is shorter than in the case of the implementation on real systems and the execution of many different runs with different parameters and pseudo random event sequences allows obtaining valid statistics on the collected data.

The simulation libraries and tools required for this approach have been steadily improved during the last decades and belong to the standard tools of the network scientists.

Clearly the quality of the observations, and in particular their adherence to the reality, is strongly dependent on the quality of the implemented simulation model. In the particular case of WSNs the simulation model has to consider the peculiarities of the propagation medium and its time variant characteristics.

The first investigations of WSN systems have been done by using the same models

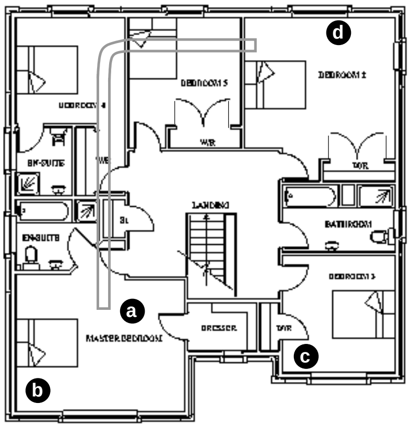


Figure 5.2: *Example of indoor node placement*

which had been developed for the Mobile Ad Hoc Networks (MANETs). These models were mainly aimed at reproducing the effects of mobility on the communication between nodes. The underlying propagation model was in the best case a simple Line of Sight (LOS) model. Unfortunately, the LOS models are unable to reproduce the effects which can affect the propagation if the nodes are close to obstacles or if the network is deployed indoor. For a more complete analysis of the problem of developing a trustworthy propagation model, the reader can refer to the following section 5.2.

5.2 Considerations for the propagation model

By observing a floor plan, like the one in Fig. 5.2, it is possible to speculate about the required characteristics of indoor propagation model and the inappropriateness of the previously described models.

We start considering the apparently simplest case represented by the direct communication between the nodes **a** and **b**. Because of the absence of obstacles in the direct communication path, we could be tempted to adopt the simple LOS propagation model [44], where the ratio betwen emitted (p_e) and received power (p_r) is inversely proportionally to an appropriate power of the distance $p_r/p_e \propto d^{-\gamma}$, where the exponent γ varies in the range $[2 \ldots 4.5]$.

Unfortunately walls and furniture reflect the radio signals. The receiver observes the superimposition of many signals arriving with different delays. The macroscopic effect of this *multipath propagation* is the appearance of sudden variations of the received signal depending on the placement of sender and receiver. Typically, the distance between a maximum and a minimum of the received signal is of the size of the half of a wavelength (about 6.25 cm for devices operating in the 2.4 GHz band).

Lymberopoulos et al. [45] try to empirically characterize the indoor propagation channel observed by a device operating in the in the 2.4 GHz band equipped with monopole antennas, for the case of receiver and transmitter located in the same room. The results show that the LOS model holds only in the proximity (nearer than 1 m) of the transmitter. As soon as the distance between sender and receiver increases, the correlation between signal strength and distance becomes weaker and for distances bigger than 2 m the intensity of the received signal (represented in a logarithmic scale)

Table 5.2: *Typical attenuation values for 2.4 GHz signals*

Material	Attenuation (dB)	Material	Attenuation (dB)
Glass windows	2.3	Brick wall	2
Office wall	6	Metal door in brick wall	12
Drywall	5-8	15 cm thick solid-core wall	15-20
Human body	3		

appears almost uniformly distributed. Furthermore, the same paper shows that the probability of having links with asymmetrical propagation characteristics increases with the distance between sender and receiver.

As next step, the reader might consider the case arising when the nodes are placed in different rooms (e.g. the path between the nodes **a** and **c**). The obstacles along the propagation path cause a further signal attenuation. The importance of this contribution to the total attenuation depends on the characteristics of the obstacles and is almost constant. Typical attenuation values are reported in Table 5.2.

A further complication is represented by the existence of unexpected propagation paths (cables, conduits, etc.). With respect to Fig. 5.2 the gray lines represent an air conditioning conduit, which offers an alternative propagation path between the nodes **a** and **d**.

Furthermore, in a large network the nodes do not remain confined to a single floor and hence an explicit support of the third dimension is mandatory for the propagation model.

Recently, the problem of providing a better propagation model for the study of MANETs has been addressed with an efficient ray-tracing based approach [46]. The path loss is explicitly computed offline only for a subset of the possible node positions and then the simulation programs computes the path loss for each placement by interpolating the previously computed values. This elegant approach seems to produce high quality descriptions of the propagation environment, but the overall computational costs are not negligible especially if the size of the building increases.

5.3 *Islands* model

In order to address the previously described problems, the author developed a new radio propagation model and an empirical procedure to convert a floor plan to a suitable set of simulation parameters. The complexity of the model has been limited, by waiving the exact knowledge of the node placement in favor of rough positioning information concerning the room the nodes are installed in. It has to be noted that, according to the observations in [45], the overall information loss is only marginal at least for the path loss estimation.

The model generation starts by decomposing the floor plan into multiple non overlapping propagation areas (*islands*). Each island is fully contained in a single room (Fig. 5.3 left). Particularly large rooms might be conveniently split in more than one

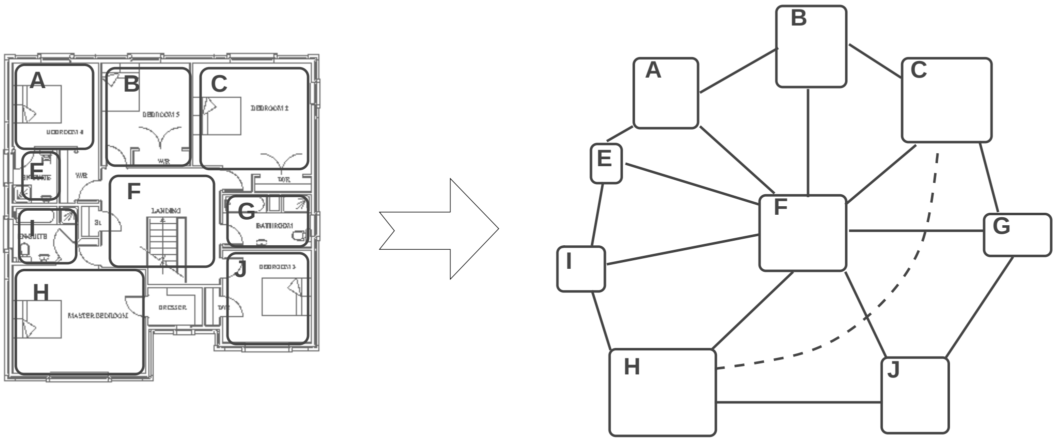


Figure 5.3: *The origin of the* Islands *model*

island.

An undirected weighted adjacency graph, having the islands as nodes and the attenuation between adjacent nodes as weights, is then generated (Fig. 5.3 right). The weight of each edge is set according to the average attenuation (in dB) measured between nodes belonging to the two islands connected by the edge. Hence the weights reflect the expected attenuation due to the obstacle in the path between the two islands. Additional edges might be added in order to explicitly consider some particular propagation path. E.g. the dashed edge represents the air conditioning conduit.

The communication between two nodes (**a** and **b**) located in the same island is affected by a path loss $L_{(a,b)}$. Its value (in dB) is given by

$$L_{(a,b)} = L_f + L_{s_{(a,b)}}(t) : \; L_{s_{(a,b)}}(t) \in \mathcal{N}(m_{(a,b)}, \sigma^2) \tag{5.1}$$

where L_f represents the constant attenuation due to the physical parameters of the equipment used and $L_{s_{(a,b)}}(t)$ is a stochastic parameter drawn according to a Gaussian distribution of mean $m_{(a,b)}$ and variance σ^2. This component approximates the effects of the variation of the propagation conditions which may arise during the interval between two consecutive packets[2]. The mean value $m_{(a,b)}$ takes into account the attenuation due to the distance between the nodes and it is drawn at the beginning of the simulation according to a uniform distribution in the interval $[L_{min}, L_{max}]$, whose extremes are chosen according to the room dimensions.

Because of the extremely low duty-cycle, the values of $L_{s_{(a,b)}}(t)$ affecting the transmission of the different packets can be assumed as independent and identically distributed. Therefore, a new value for $L_{s_{(a,b)}}(t)$ is drawn each time a packet has to be generated.

For the communication between nodes in different islands (**a** and **c**) , the model takes in account the contribution of a constant attenuation between the two propagation areas $L_{(R_a,R_c)}$ which depends on the building materials and on the distance and the stochastic contribution inside each room

$$L_{(a,c)} = L_f + L_{(R_a,R_c)} + L_{s_{(a,v)}}(t) + L_{s_{(v,c)}}(t) \tag{5.2}$$

---

[2] The interval between two packets is in the order of some minutes.

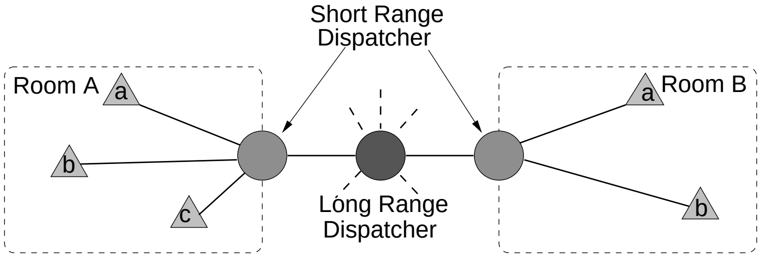


Figure 5.4: *The propagation model*

The parameter $L_{(R_a,R_c)}$ is the minimum of the sum of the attenuation components associated with the edges along the paths between the two islands[3], and $\mathbf{v}$ is a virtual element modeling the interface between the propagation area and the outer world.

5.3.1 Implementation

The *islands* model is particularly suitable for the implementation in a simulation model. Our implementation has been developed in C++ using the library *Omnet++* [48]. It consists of:

- a message structure which stores the messages coming from the nodes together with the signal level information
- a simple dispatcher module, which is responsible of modifying the signal level indication according to the parameters of the propagation path
- an interface module between the node model and the propagation model.

Two different classes derived from the common dispatcher class implement the models for the Short Range (SR – inside an island) and Long Range (LR – between different islands) communication, respectively.

All nodes belonging to the same island are directly connected with a SR–dispatcher which models at the same time the interface towards the other rooms and the propagation channel for the communication between the directly connected nodes. A single long range dispatcher connects all the islands (Fig. 5.4).

At boot time, each SR–dispatcher draws the average attenuation between each pair of directly connected nodes according to a uniform distribution

$$m_{(a,b)} \in U[L_{min}, L_{max}] : \forall a, b \in S_I, \forall I \tag{5.3}$$

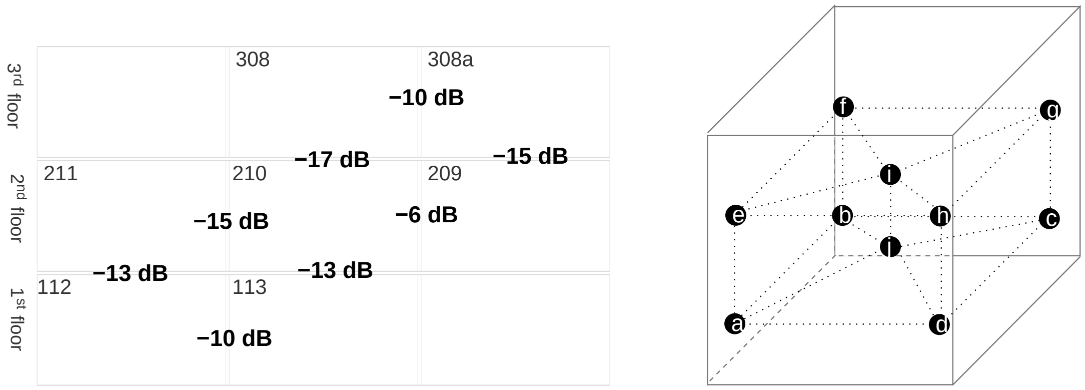
where S_I is the set of nodes in the island I, including the virtual node v.

At the same time, the LR–dispatcher reads the attenuation coefficients of the weighted adjacency graph from a configuration file, and then runs *Dijkstra's algorithm* (p. 527-532 of [49]) in order to compute the minimal attenuation path between each pair of islands.

Each dispatcher generates as many replicas of the incoming messages as the number of links minus one, updates the signal intensity fields and sends the messages out. In order to keep the number of messages to a minimum, and hence to speed up the computation, the packets are removed from the model as soon as the signal level indication

---

[3] This model is compatible with the "Attenuation Factor" (AF) model [47].



(a) Room arrangement and measurement paths

(b) Measurement points

Figure 5.5: *Rooms and measurement points arrangement*

falls below a given threshold. The SR–dispatcher draws the signal attenuation (in dB) for each outgoing packet accordingly to a normal distribution[4]. The LR–dispatcher does not carry out any complex computation; in fact it has only to look up the attenuation value for each source destination pair in the local table.

This approach requires less computation and storage than the usage of the LOS model which requires the computation of the path loss for each pair of nodes.

5.3.2 Model validation

In order to empirically validate our model, path loss measurements have been done in our office building. An IEEE 802.11b transceiver, equipped with a dipole antenna, has been used as signal source while a “Yellow Jacket” 802.11 measurement tool has been used to collect the data.

The measurements have been carried out in seven rooms distributed over three floors (Fig. 5.5(a)). Ten equipment placement points have been defined in each room (Fig. 5.5(b)). Each measurement consisted in the acquisition of ten values, one each 10-20 seconds, for each equipment placement.

A first series of measurements, aimed at observing the received signal in absence of impairments has been done in room 308. All possible combinations of sender and receiver placement have been tested. As expected, the first results agree with the observation of Lymberopoulos. The average of the measured signal levels (-47 dBm) has been computed. It represents the mean signal level measured in absence of obstacles.

The second step consisted in obtaining an estimation of the attenuation due to walls and floor/ceiling. The Radio Frequency (RF) source as been put in the middle of each room and the signal intensity has been measured in the adjacent room(s) at each measurement point. The difference between the mean received power and the previously estimated value is an approximation of the additional attenuation due to the building material (and the additional distance too). The results are reported in Fig. 5.5(a).

The last step consisted in measuring the path loss between non adjacent rooms and

---

[4] The resulting path loss is lognormal distributed, which agrees with many common propagation models.

Table 5.3: *Measured and expected average path loss (dB) in the test field*

Room nr.	308	308a	211	210	209	113	112
308		-10	-33 (*-32*)	-17	-22 (*-23*)		-27 (*-30*)
308a	-10		-39 (*-32*)	-27 (*-21*)	-15	-45 (*-44*)	-33 (*-34*)
211	-33 (*-32*)	-39 (*-32*)		-15	-19 (*-21*)	-13	-22 (*-23*)
210	-17	-27 (*-21*)	-15		-6	-22 (*-23*)	-13
209	-22 (*-23*)	-15	-19 (*-21*)	-6			-20 (*-19*)
113		-45 (*-44*)	-13	-22 (*-23*)			-10
112	-27 (*-30*)	-33 (*-34*)	-22 (*-23*)	-13	-20 (*-19*)	-10	

comparing the observations with the values which had been computed by applying the propagation model on the previously obtained attenuation values. The results are even in Table 5.3 where the values generated by using the propagation model have been put in brackets. Despite the limited set of measurements, the good agreement between the respective values indicates the viability of the proposed solution.

The distribution of the values collected in each position seems to fit a normal distribution with a standard deviation varying between 4 dB and 7 dB.

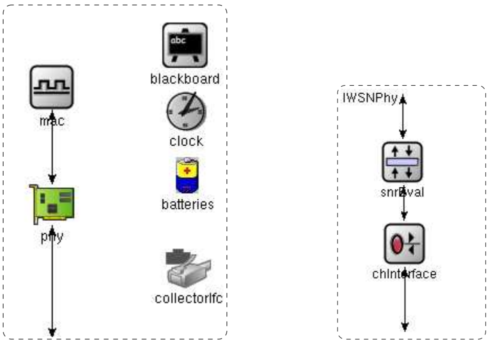
5.3.3 Further considerations

The proposed model can easily be extended in order to simulate noticeable modifications of the radio environment which might happen during the network operation time (as in consequence of opening/closing doors, etc.). It is sufficient to change the attenuation values used by the LR–dispatcher and then to rerun the shortest path algorithm.

Furthermore, it is not too difficult to use the proposed propagation model to mimic the behavior of the commonly used solutions based on either a grid or a random node distribution. The starting point is to build a network having as many islands as nodes and putting a single node into each island. The grid case can then be reproduced by setting the path loss parameters in such a way that a direct communication between non adjacent islands becomes impossible. The random case can be emulated by offline computation of the node distribution and the corresponding attenuation between each pair of nodes. These values can than be provided to the LR–dispatcher.

The lack of an explicit support for node mobility does not reduce the validity of the approach. In fact, in the considered application scenario (see chapter 4) the nodes do not change their position and only environmental changes are responsible for modifications of the network topology.

The randomization of the path loss also reproduces the asymmetry of the radio channel, which has been highlighted in [45].



(a) Internal structure (b) The module *“phy”*

Figure 5.6: *Structure of a node*

5.4 A simulation framework for indoor WSNs

Besides the propagation model, there are many other elements which might be reused in different simulation programs. Because of this reason, a simulation framework similar to the *Mobility Framework* [50] developed at the TU-Berlin has been implemented with the explicit purpose of speeding up the development of the simulation models. The framework provides the signal propagation model an easy mechanism to interact with node models and many basic modules which a developer can use to build the own models[5].

As example, Fig. 5.6(a) shows the structure of a node model based on the classes provided by the framework. All building blocks are either simple elements, obtained by derivation of a basic module, or composite element which contains other simple of composite elements.

The *“blackboard”* module, probably the only object which does not need derivation, provides the cross-layering functionalities (see section 3.3.1). It implements a data exchange structure and an event notification service which works according to a publish/subscribe strategy. This module is accessible from all elements belonging to the same node. Each module can access the blackboard in order to publish new information or subscribe to already published items. The collector interface module (*collectorIfc*) uses a blackboard module accessible from all nodes (*gobalBlackboard*) in order to supply the *globalCollector* (the element which collects the measurements and generates the statistics) with the required data.

The composite module *“phy”* (Fig. 5.6(b)) consists of two parts. The channel interface (*chInterface*) which provides the interface towards the propagation model and depends on this one, and the module *“snrEval”* which, on the contrary, is independent from the underlying propagation model.

The module *“snrEval”* simulates the transmission delay, computes the Signal to Interference Noise Ratio (SINR) associated with each received data block and decides – based on the computed SINR value– which packets have been correctly received and have to be propagated to the upper layer. This module has been implemented with

---

[5] The *Blackboard* model used for asynchronous data exchange has been taken from the *Mobility Framework*

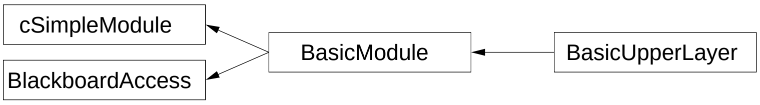


Figure 5.7: *The derivation of the class* BasicUpperLayer

the explicit purpose of simulating the *capture effect*[6] [51, 52] which might arise when two signals, characterized by a different signal strength, arrive almost simultaneously at the receiver.

5.4.1 The classes *BasicModule* and *BasicUpperLayer*

All protocol layers (or their components in case of composite modules) are objects derived from the same class *BasicUpperLayer*, whose derivation is sketched in Fig. 5.7.

This class offers a pair of equally sized gate vectors (input/output) towards the upper layer modules and a single gate pair towards the lower layer. The messages arriving from the upper layer are encapsulated and the index of the arrival port is attached to the new message. The port index is then extracted from the message arriving from the lower layer before decapsulating it and used to address the output port, if the message has to be propagated upwards.

The three handler functions *handleLowerMsg*, *handleUpperMsg* and *handleSelfMsg* are called in order to react to the arrival of new messages from the lower level, the upper level or the module itself, respectively. These three functions contain the actual protocol functionalities. These handlers together with the multistage module initialization function *initialize* and the *finish* function are, in principle, the only elements which have to be rewritten in order to implement a new protocol. The class *BasicUpperLayer* does not implement any specific protocol and only propagates the messages between upper and lower layer.

The parent module *BasicModule* uses the multi stage initialization provided by Omnet++ in order to assure the publication of the objects on the blackboard. This has to happen during the phase 0 before starting the subscription process in the phase 1. The same module retrieves the Omnet++ module *id* of the node and node's name. The former can be used as address while the second is useful in order to identify the single nodes during debugging.

The class *BlackboardAccess*[7] provides the methods used to interact with the blackboard.

---

[6] The strongest of many interfering signals is correctly decoded, in spite of the collision event.

[7] Taken from the *mobility framework*.

Chapter 6

Network bootstrap

As highlighted at the end of chapter 4, the topology discovery and the subsequent routing initialization are two fundamental steps of the network deployment phase. The successful completion of the corresponding operations is a prerequisite for the network's ability of entering the normal operational state.

This chapter is focused on the early stages of the network life. It starts with an analysis of the problems arising during the deployment phase, with a special reference to the Wireless Metering Networks (WMNs), and gives an overview of the solutions usually adopted.

After the introductory sections, the rest of the chapter is aimed at introducing and analyzing a simple distributed heuristic algorithm[1] which addresses the topology discovery and routing initialization problems in a combined fashion.

This lightweight algorithm, which has been developed by considering the HW constraints described in section 4.2, permits the integration of all sensor nodes into a tree-like failure tolerant routing structure and also provides continuous adaptation of the structure to possible changes. Simulation results suggest that the proposed algorithm converges quickly to a structure with the desired properties even in the worst case.

6.1 An overview

Just after the node placement, when the network elements start functioning, they have to to discover their position in the network topology by finding out the possible communication partners. As noted by Kumar and Chong [5] this *ad hoc network discovery* is the first step to build an operational network and may be particularly challenging especially considering the requirements stated in the previous chapter for a WMN (see section 4.2). Interestingly, in the literature most of the studies consider a running network and do not address the early power-on stages and subsequent topology maintenance at all.

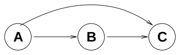
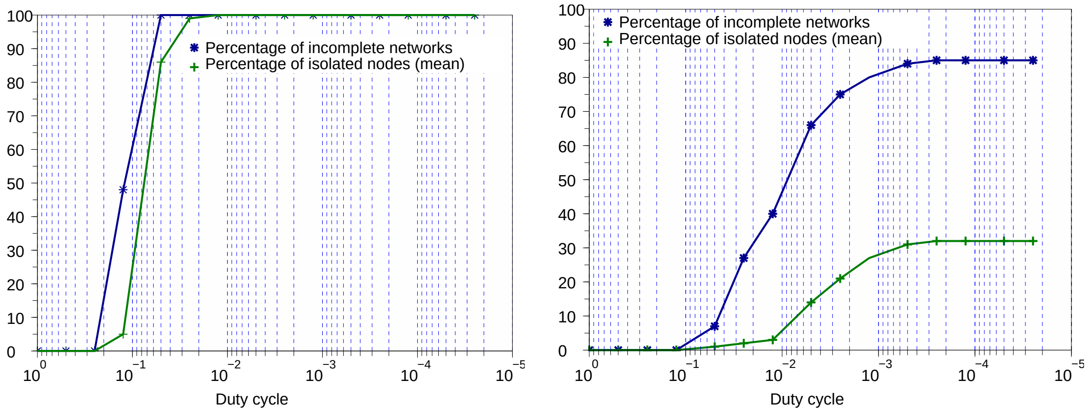


Figure 6.1: *A simple NG*

---

[1] The basic ideas of this approach have been patented in the year 2007 [53].



(a) Same duty cycle during the whole simulation run

(b) Observation of a whole beacon interval ($n = 5$)

Figure 6.2: *Relative frequency of connected networks in dependency on different duty cycle values*

Output of the topology discovery is a set of neighboring relationships which establish a correlation between each node and its possible communication partners. A Neighbor Graph (NG) is a suitable abstraction to represent all these relationships. It is a directed – hopefully connected – graph whose nodes represent the sensor nodes while the directed edges represent the node's ability of sending data from the origin node to the destination neighbor. Fig. 6.1 shows a situation where node A is able to communicate with B and C and B can communicate with C whereas C can not communicate with any node.

Once the topology has been discovered, the nodes have to cooperate in order to create a routing structure over the NG. The following normal operational state can start only once each node has been integrated into the routing structure. For a WMN having a single concentrator, the *Minimum Spanning Tree (MST)* is a possible solution for the routing problem. Moreover, if all edges are equally weighted, the MST is equivalent to the *Breadth-First Spanning Tree (BST)*. Failure tolerance considerations suggest the creation of multiple (redundant) routing paths from each node.

Unfortunely, the duty cycle significantly influences the network initialization process. If the duty cycle is high enough, the completion of the initialization procedures is a straightforward process. However, when reducing the duty cycle, the probability of obtaining a fully connected network decreases and for very small duty cycles tends towards zero. This is due to the fact that the probability that a node observes the signal generated by a specific neighbor decreases with the duty cycle.

The simulation results reported in Fig. 6.2(a) have been collected using the simulation program described in chapter 5 and the node distribution described in section 6.2.4. In particular, 100 nodes evenly distributed over 10 rooms had been powered on in a random order, during an interval of 10 beacons. Each node worked since its activation according to the chosen duty cycle. Network initialization has been made easier by assuming that the sink is already active since the beginning of the node deployment (which is not always the case in reality). The duty cycle varies from 1 to $0.8 \cdot 10^{-4}$. Hundred runs with different node placements and different node activation sequences have been done for each value taken by the duty cycle parameter.

A possible solution to the problem is allowing the node to use an higher duty cycle for some time [54]. Assuming sensor node hardware with the constraints in section 4.2, the energy costs for the observation of a complete beacon period correspond to about 1.5% of the yearly energy budget. The high duty cycle periods have, therefore, to be strictly limited.

It is easy to show that the observation of the channel for one complete beacon period is a sufficient condition for the computation of the BST, even in a network characterized by a low duty cycle, as long as all nodes have already been activated at the beginning of this beacon period and the nodes have enough resources to observe the activity of all the neighbors. Both conditions, however, are typically not met. Since it is not clear over which period the deployment process is carried out, it is difficult to ensure that the deployed nodes start their scanning cycle only when all the nodes are activated.

Furthermore, hardware constraints and energy considerations impose an upper limit on the maximum number of permanently observed neighbor nodes – even if the deployment procedure ensures that all neighbors can be detected during the scanning period. In this case it makes sense that only the neighbors producing the best signals are stored for further observation.

In the following, it will be assumed that each node observes the channel for a complete beacon period just after its own activation. Therefore, the node is able to detect all the neighbor nodes (i.e. all nodes in communication range), which are already active at its power-on time. The node then selects the beacon instants of the m strongest neighbors for further observation and stores them. In the subsequent beacon periods, the node wakes up periodically at these instants and is able to receive the signals of the selected neighbors and only those.

Fig. 6.2(b) shows the probability of obtaining a completely connected network if this activation scheme is carried out and a node can permanently observe at most $n = 5$ neighbors. The simulation study has been carried out using the same previously described simulation setup and following the same procedure.

The decrease of the relative frequency of connected networks for decreasing duty-cycle may be explained observing that the deployment of the nodes takes many beacon periods and hence some nodes develop only a partial knowledge of the surrounding environment.

Observing a complete beacon interval by using a small observation window and shifting it a bit every beacon period represents another possible strategy[2] allowing to detect all neighbors without consuming too much energy. The most important drawback of this solution is that network initialization requires a much longer time which makes the solution infeasible in the context of an economically sound installation process.

6.1.1 Related works

Even though the neighbor discovery is a problem of primary importance for a Wireless Sensor Network (WSN), it has been seldom explicitly addressed. Most of the works simply suggest to modify the low duty cycle scheduling for the time required to integrate all the nodes, like in [54].

Probably the first paper addressing the problem of a low power neighbor discovery was [55]. The authors, M. McGlynn and S. Borbash, leverage the *birthday paradox*[3],

---

[2] This strategy is analyzed in the following chapter 7.

[3] The birthday paradox is a derivation of the simple problem of finding the minimal number of

in order to permit the neighbor discovery when the nodes operate according to a low duty-cycle strategy. This strategy is simple and robust, but it is characterized by an extremely slow convergence, which makes it unsuitable for the considered WMNs.

The ZigBee [27] specification highlights the importance of the problem and states that this is a compulsory functionality of the Network Layer Management Entry (NLME) but it omits the indication of a specific solution.

A recent Internet-Draft published by the IPv6 over Low-Power Wireless PAN (6LoWPAN) Working Group [56] suggests the utilization of a modified version of the *Router Discovery Protocol*, which belongs to the TCP/IP protocol suite. This approach is effective; however, it does not address the issue of low energy consumption.

An alternative approach to the problem can be found in the work of Xue and Kumar [42]. The two authors analytically determine the minimal number of neighbors each node has to find in order to guarantee that the NG is connected. This work states that when the number of neighbors $n > 5.1774 \cdot \ln(N)$, the network with N nodes is asymptotically connected. At the end of the paper, the authors postulate that there is a critical value c close to 1 such that $\forall n \ : \ n > c \cdot \ln(N)$ the probability of having a connected network approaches 1 even for small networks. This result obtained with a specific node distribution seems to apply to a wide variety of networks.

The routing problem has then been addressed by many authors, but most of the proposed solutions overkill the requirements of the WMN analyzed in this thesis.

6.2 A heuristic algorithm

6.2.1 Problem definition

The following considerations apply to a WMN with the known constraints (see section 4.2).

In particular each node operates its transceiver according to a low duty cycle scheduling and produces at least one measurement each day. The overhead due to the generation of the maintenance traffic should not exceed 40% of the channel capacity, in order to reserve enough bandwidth for the transmission of the measurement data even over a bottleneck link. The beacon signals used by the MAC layer are generated periodically every T_{beacon} seconds. Since the beacon frames trigger the beginning of an activity period for the MAC layer, only the nodes which receive the beacon are able to send data to the sender of the beacon.

Because at the beginning all the nodes are running unsynchronized, it can be assumed that the beacon generation points of the various nodes are uniformly distributed over the beacon interval T_{beacon}.

Some theoretical results indicate the existence of a trade-off between latency and energy consumption during the power-on phase of a WSN [57]. Simple computations indicate that without changing the duty cycle in the power-on phase, network initialization can take several days. If a node can observe the channel for an interval δ each T_{beacon} the mean time T_{sync} needed to establish the synchronization with one of its n neighbors is about

---

people which have to seat around a table, such that, the probability of having at least two of them which celebrate the respective birthday on the same day of the same month is higher than 50%. The quite surprising result is 23.

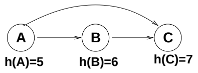


Figure 6.3: *Example of NG*

$$\frac{T_{beacon}}{2\alpha(k+2)} \leq T_{sync} \leq T_{beacon}\left(1+\frac{k}{\alpha(k+1)(k+2)}\right) \tag{6.1}$$

where $\alpha = \delta/T_{beacon}$ and k is the number of directly reachable neighbors. This means for the considered network with on average 10 neighbors per node $T_{sync} \geq 41\ hours$.

Energy and memory constraints impose an upper limit $n \geq 1$ to the out-degree of each node[4]. Therefore, each node has to select a set of at most n neighbor nodes after its activation and n will in general be smaller than the number of sensor nodes within communication range. Because each node chooses the neighbor nodes it wants to synchronize with autonomously, the neighborhood relationship has to be considered as unidirectional. In this case the out-degree of each edge of the resulting neighbor graph (NG_n) is upper bounded by n.

6.2.2 Dealing with unidirectional links

Each newly activated node observes the communication channel and builds a list including at most $m \geq n$ already active neighbor nodes. The selection criteria of the possible neighbors (in general, m is still smaller than the number of nodes in communication range of the respective sensor) is based on the average strength of the observed beacon signals.

The beacon signal carries, among other things, an indication of the current (logical) distance (h) in hops from the sink along the shortest path. At the beginning, each node has an initial value for this distance (h_0), which is bigger than the maximum number of nodes which will be active in the network[5]. The sink uses distance 0 as initial value and doesn't change it.

Each node schedules its activity periods in order to receive the beacons generated by the selected neighbors.

Each node can answer to a beacon with an update packet in order to propose itself as neighbor to the beacon's sender. This communication happens with probability $p > 0$ and only if the new connection will allow the beacon's sender to reduce its distance from the sink. Consider as example the NG in Fig. 6.3: Node A will generate an update packet only for C as B can not decrease its distance from the sink as result of this packet.

Upon reception of a beacon (or update) packet, each node re-evaluates the set of nodes it synchronizes with and computes the new distance value (cf. Algorithm 1).

In absence of a limitation on the number of neighbors n a node can synchronize with, the neighbor graph (NG_∞) can be thought of as derived from the undirected

---

[4] Number of nodes each element selects as neighbors.

[5] So it is possible to identify easily if all nodes have been integrated into the network.

graph representing the radio coverage (RG)[6] by choosing at random the direction of each link[7].

The computation of the BST is trivial in an undirected graph. The possibility of generating update packets and, therefore, of changing the direction of the edges of the neighborhood graph, makes the computation of the BST feasible over NG_∞ [8]. With respect to the node distance after convergence, there is no difference between computing the BST over the RG or over the NG_∞[9].

By introducing a constraint for the maximum number of neighbors n, the corollary 1.2 gives a sufficient condition to obtain a connected network.

Algorithm 1: Neighbors selection algorithm

```
$N$            is the set of neighbor nodes the element is synchronized with.
$n$            is the maximum number of neighbors a node can be synchronized with
$h_{local}$    is the current logical distance from the sink.
$s$            is the sender of the beacon (or update) packet.
$h_x$          is the logical distance of the node $x$ from the sink

foreach Beacon (or Update) packet do
    if $s \in N$ then
        Update the information about $s$
    else if $n >| N |$ then
        $N \leftarrow N \cup \{s\}$
    else if $\exists r \in N : h_r > h_s$ then
        $N \leftarrow N \setminus \{r\} \cup \{s\}$
    end
    // compute the new value of the distance from the sink
    if not is sink then
        $h_{local} \leftarrow min_{q \in N}(h_q) + 1$
    end
    if $h_s > h_{local} + 1$ then
        generate update with probability $p > 0$
    end
end
```

A first analysis suggests that the algorithm converges to a stable structure in a time proportional to the depth of the tree. During this time the nodes generate a number of update packets that in the observed cases is proportional to the number of nodes. The proportionality constant is dependent on the generation probability p of an update packet.

As an example the two structures obtained running the algorithm for the same node distribution with $n = \infty$ and $n = 5$ have been plotted in Fig. 6.4 and Fig. 6.5, respectively. It is obvious that the degree of connectivity is higher for the case with $n = \infty$, but also for $n = 5$ the network is fully connected and there exist some redundant

---

[6] Given a set of nodes N, which transmit continuously with the same power, the graph $RG = (N, E)$ is the communication graph such that $(u, v) \in E$ if and only if u and v are within transmitting range of each other.

[7] The direction of the edges is defined by the node activation sequence.

[8] Cf. corollary 1.1, section 6.2.3

[9] Cf. lemma 2, section 6.2.3

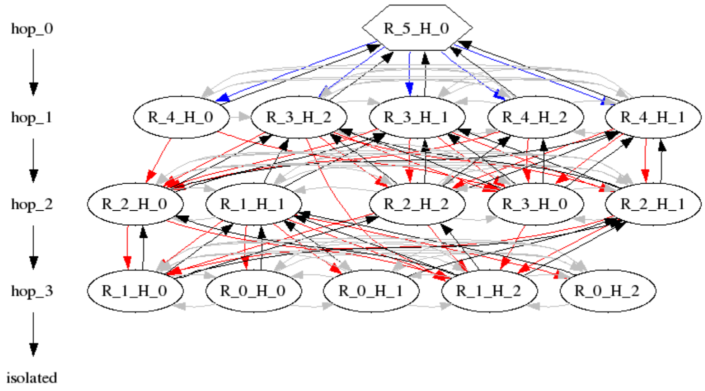


Figure 6.4: *Example of the structure generated by the algorithm 1 by removing the constraint on n*

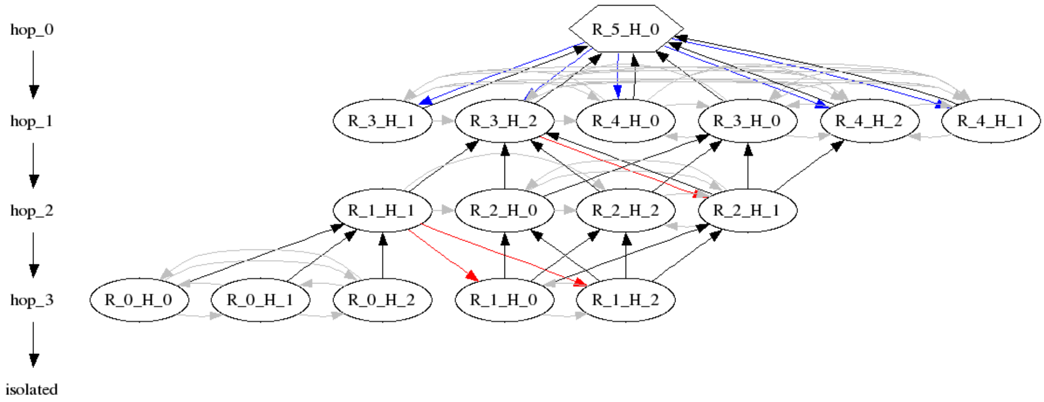


Figure 6.5: *Example of the structure generated by the algorithm 1 for $n = 5$*

uplink paths[10]. It is interesting to observe (also for $n = 5$) the presence of some edges, which are not directed to uplink nodes and, hence, which are not interesting for the routing of the data packets.

Even though the unused links can be removed to save energy, it is advantageous to preserve these in order to improve the network's capability of reacting to modifications in *RG*. In fact the proposed algorithm (with the few enhancement described in the following chapter 7) runs all the time to accommodate the changes of the radio coverage and to eventually allow the integration of new nodes. The regularity which can be observed in these small examples is even stronger in structures with much more nodes.

---

[10] Paths to nodes which are logically closer to the sink.

6.2.3 Asymptotical results

By relaxing the constraints on the number of observed neighbors, the following theoretic results show that the exchange of *update packets* with non zero probability is a sufficient condition to ensure the integration of all nodes into a tree structure rooted at the concentrator each time the RG is connected. Moreover, after the algorithm convergence, all nodes are at the same distance from the concentrator, they have in the BST computed over the Radio coverage Graph (RG).

The following propositions make use of these assumptions:

$G = (N_G, E_G)$ is a directed graph with at least one node marked as *sink* (s)

$U = (N_U, E_U) : \forall d, e \in N_U = N_G,\ (d,e) \in E_U \Leftrightarrow (d,e) \in E_G \vee (e,d) \in E_G$ is a connected undirected graph

$S_\infty(t)$ is the graph obtained at the time t from G running the algorithm 1 without constraints for the number of neighbors ($n = \infty$)

$h_t(e) \geq 0$ is the logical distance of each node $e \in N_G$ from the sink along the shortest path at the time t (the equality is satisfied only by the sink s)

no update packet gets lost

Lemma 1. *Each time two nodes (d and e) are connected by at least one edge in G, the value* $d_{e,d}(t) = \mid h_t(e) - h_t(d) \mid$ *converges for* $t \to \infty$ *with probability one to a value not greater than one (or in other words the probability* $P_{d>1}(t)$ *of the event* $\{d_{e,d}(t) > 1\}$ *goes to 0 for* $t \to \infty$*).*

Proof. $h_t(e)$ and $h_t(d)$ are non increasing and non negative functions of time defined over the integer numbers. There is a time T such that $h_T(e)$ and $h_T(d)$ reach their minimum.
If $d_{e,d}(T) \leq 1$ then $\forall t > T\ P_{d>1}(t) = 0$.
If $d_{e,d}(T) > 1$, after the reception of each beacon, the node nearer to the sink (say d) generates an update according to a Bernoulli process with success probability $p > 0$, The probability that e does not receive any update decreases with the time and is $\forall t > T,\ P_{d>1}(t) \leq (1-p)^{\lfloor \frac{t-T}{T_{beacon}} \rfloor}$, and hence $lim_{t \to \infty} P_{d>1}(t) = 0$. □

Corollary 1.1. $S_\infty(t)$ *converges to a stable structure* S_∞ *for* $t \to \infty$

Proof. After the convergence the distance between directly connected nodes becomes either 0 or 1 [11] and no more updates will be generated. □

Corollary 1.2. S_n *with* $0 < n < \infty$ *is connected each time the undirected graph* RG_n *obtained from* NG_n *by adding the reverse edge to all unidirectional links in* NG_n *is connected.*

Proof. It follows from Lemma 1 by substituting U with RG_n □

---

[11] Otherwise an update packets are generated to reverse the edges.

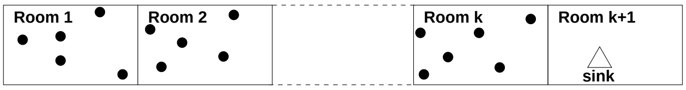


Figure 6.6: *Node placement*

Lemma 2. *The nodes of S_∞ are at the same distance from the sink that the corresponding nodes have along the edges of the BST computed over U.*

Proof. S_i is the set of the nodes of S_∞ having distance i from the sink.
B_i is the set of the nodes of U having distance i from the sink along the edges of the BST.
It is possible to suppose that first difference appears at layer i. Which means
$\exists\, e : e \in B_i, e \in S_j, i < j$.
Because $e \in B_i$, there is at least one node $s \in B_{i-1}$ and one edge $(e, s) \in E_U$ connecting the two nodes e and s in U, and hence in G.
Because of Lemma 1 the distance between the two nodes must be at most one. Hence node e can not be in S_j. ☐

6.2.4 Simulation study

Even though the asymptopotical results indicate that the proposed approach is a viable solution for the neighbor discovery and routing initialization problem, they do not give any indication on the optimal parameter selection and on the influence of it on the convergence speed. Moreover, they do not indicate a viable strategy to assure the completeness of RG_n and hence of NG_n.

A simulation study has been performed with the purpose of giving an empirical answer to the still open questions. The simulation model described in chapter 5 has been used to reproduce the propagation conditions in an indoor environment.

A series of preliminary tests permitted to identify the most critical node placements and activation strategies, which have then been used in order to highlight the worst case behavior of the proposed algorithm.

Simulation setup and metrics

It has be observed that the node placement which potentially generates the most problems corresponds to the building structure in Fig. 6.6 where the nodes are distributed over k aligned rooms and the concentrator (*sink*) is located in the room k+1.

The propagation parameters have been chosen such that the nodes in room i are able to communicate almost certainly with the nodes into the rooms $i \pm 1$ but the same nodes can not communicate with elements in the rooms $i \pm j$ with $j \geq 3$. The activation sequence of the nodes follows the room numbering, starting from room 1. A new room will be activated only when all nodes in the preceding room are already active. The consequence of this rule is that a new node has to synchronize with nodes which are already active and hence with the neighbors that are geographically at most as close to the sink as the node itself. After the sink's activation, many nodes have to choose a new neighbor in order to be integrated into the routing tree. This represents a worst case scenario for the convergence of the proposed heuristics.

Metrics

The simulation study was aimed mainly at analyzing the energy costs and the robustness of the structure the algorithm generates for different parameter values. Therefore, it has been decided to observe the average distance from the sink as parameter giving an indication of the data transport costs during the normal operation[12].

In order to describe the robustness, the k-connectivity of the graph has been considered at first. But, because of the limitation on the out-degree of each node (n) it was clear that k would be a positive integer upper bounded by n [58] and hence the parameter would not be expressive enough.

Thus, a new metric has been defined which is characterized by a more granular structure. It has been considered that during a beacon period each node generates a flow unit of traffic. This flow unit is divided evenly among all outgoing edges pointing to *uplink* nodes. The expected lost traffic (M_{FL}) under the assumption that just one link fails during a beacon period and that the failure probability is equal for each link as been chosen a robustness metric.

M_{FL} is increasing with maximum distance from the sink and is inversely proportional to the out-degree of the nodes. It holds that $0 \leq M_{FL} \leq \frac{d+1}{2}$ where d is the depth of the routing structure. The smaller M_{FL} is, the more robust is the structure.

Default parameters

The earlier simulations permitted to identify the following parameters, which represent an acceptable compromise and fulfill the hardware constraints. The combination $n = 5$ and $m = 10$ is compatible with the memory constraints and at the same time permit to operate in the region where the probability of having a connected network is not negligible [42]. The update generation probability $p = 0.6$ reflects a compromise between the overhead traffic due to updates and the time required to reach the convergence. The simulation is aimed at analyzing the impact of each single parameter on the behavior of the proposed heuristics. During the simulation the parameters imposed by the hardware, namely the beacon period ($T_{beacon} = 360\ s$) and the duty-cycle of 10^{-4}, are left unchanged.

If not stated otherwise, a test series consisted of 50 runs with different seeds per parameter value. Modifying the seed results in different values for $m_{(a,b)}$ and thus reflects different node placements within a room.

6.2.5 Simulation results

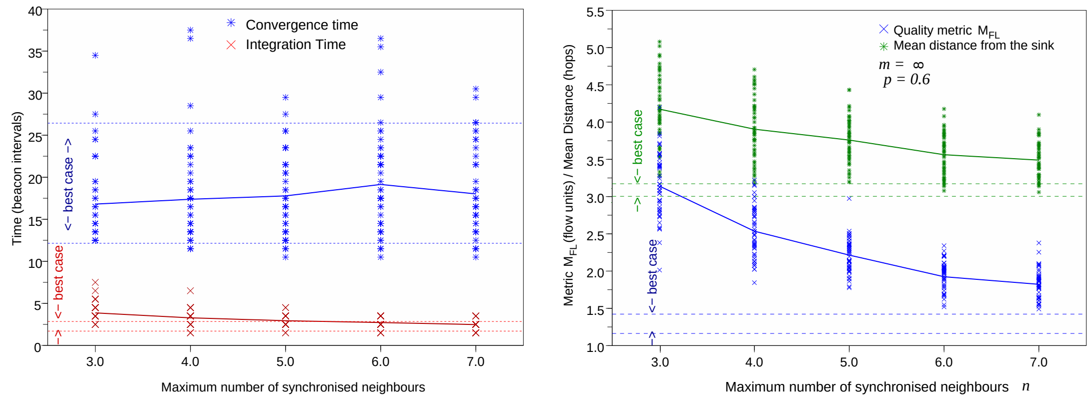
Influence of the parameter n

As the theoretical considerations already indicated, the implemented algorithm converges to a stable connected structure if there are no limitations on the maximum number of neighbors a node can synchronize with. The first test series was aimed at analyzing the influence of the parameter n on the convergence. A scenario with ten rooms, each room containing exactly five nodes, has been considered.

At activation time, each node built a list of all the neighbors which sent a beacon during the following T_{beacon} seconds ($m = \infty$).

---

[12] In absence of data aggregation the total number of packets is lower bounded by the sum of the edges along the paths between each node and the sink.



(a) Integration and convergence time after sink activation

(b) The quality parameters

Figure 6.7: *Impact of the number of synchronized neighbors n ($m = \infty$, $p = 0.6$)*

The sink was the last node which joined the network. To evaluate the results, these have been compared with the output of the simulations done with the same seeds but relaxing the constraint on n. The parameters $m = \infty$ and $p = 0.6$ did not change. The parameter n varied from 3 to 7. This range has been chosen as a compromise between the parameters suggested by Xue and Kumar [42] in order to obtain a connected network and the constraints imposed by the capabilities of the chosen hardware.

The graph in Fig. 6.7(a) shows the time to integrate all the nodes into the routing structure (upper curve) and the time until the sum of the node distances converges (lower curve). The graph shows that the time the network needs to converge to a stable structure is upper bounded by 4 hours, which is less than 10% of the lower bound given by the Eq. 6.1.

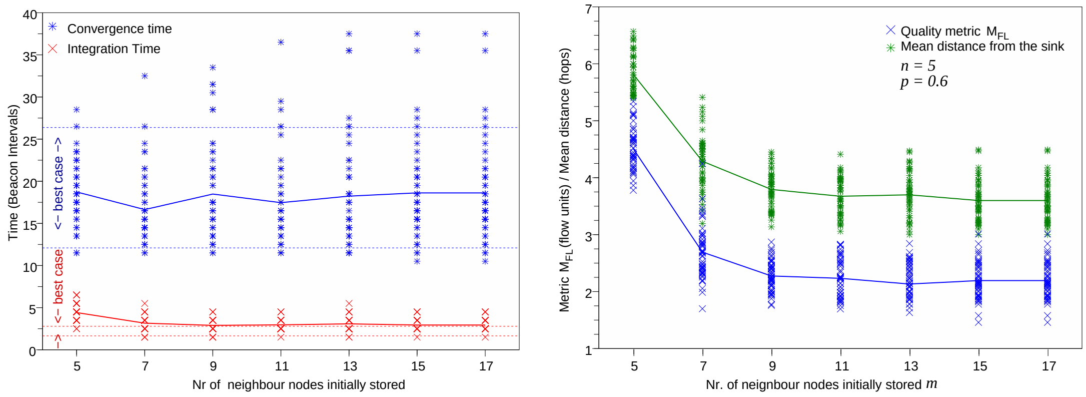
The horizontal lines represent the range of the respective parameters in the best case obtained setting $n = m = \infty$ for comparison reasons.

As expected, an increment of n moves the results toward the best case. The values of the mean distance (upper curve) from the sink and of the quality metric (lower curve) defined in the previous section have also been collected and are shown in Fig. 6.7(b) along with the lines representing the best case. For $n \leq 5$ a significant improvement of the quality parameters can be observed, whereas a saturation is approached for larger values. As an increase of n increases memory and energy consumption, $n = 5$ seems to be a reasonable value.

Influence of the strategy adopted for the selection of the possible neighbors

Because of the hardware constraints, especially because of the memory constraints, the nodes are not able to check and maintain the connectivity with all active neighbors. Therefore, at the beginning each node selects only the m best neighbors (based on the signal quality) out of all active nodes. This process can influence the behavior of the system because it may exclude some useful nodes.

During this test series, consisting of 50 runs for each value of the parameter m, each node uses the signal quality information in order to select the best m neighbors, with $m \in [n \ldots n + 12]$. The results are plotted in Fig. 6.8(a) and Fig. 6.8(b). Putting



(a) Integration and convergence time after sink activation

(b) The quality parameters

Figure 6.8: *Impact of the number of neighbors initially stored* m *(*$n = 5$*,* $p = 0.6$*)*

too strict limitations on the number of observed nodes is obviously counter productive. A node relies for its decisions on the signal quality it observes during a limited time interval, neglecting possible fluctuations of the channel properties.

Afterwards the nodes choose their neighbors according to the distance from the sink. Choosing a too small value for m implies that the node knows with high probability only nodes located in the same room and hence probably not having a better position in the routing structure.

For $m \geq n + 3$, only minimum improvement in the behavior of the algorithm has been observed.

Influence of the parameter p

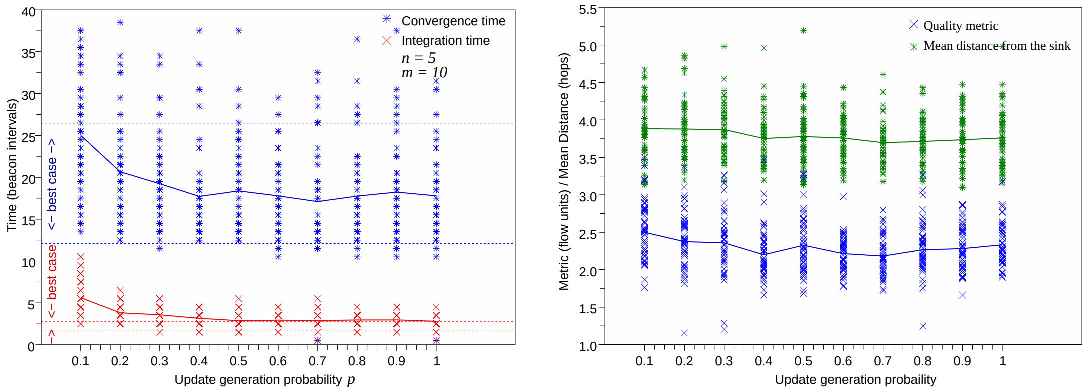
Theoretical results provide evidence that the algorithm converges for each value $p > 0$, and the overhead due to the generation of update packets will increase with p. Moreover, the updates are generated using an unslotted ALOHA mechanism and therefore a large number of collisions of updates might occur in densely populated areas. These considerations would motivate a preference for a small p. On the other hand, it is reasonable to expect that choosing p too small will have a negative impact on the convergence. To quantify these effects, a test series of 50 runs for each value of the parameter p, which varied in the interval $0.1 \ldots 1.0$, has been performed.

The results depicted in Fig. 6.9(a) suggest that for increasing values of p, the network structure converges slightly faster. The improvement seems to be negligible for $p \geq 0.5$. The quality parameters are obviously almost insensitive to variations of p and, thus, do not suggest a specific choice of this parameter. Therefore, convergence time and energy consumption are the dominant factors for the choice of p.

In order to analyze the additional energy costs resulting from the generation of the *update* packets, 100 longer runs[13] have been done for $p = 0.6$ (the value the author suggests to use) and $p = 1$ (the worst case). The observations are plotted in Fig. 6.10(a) and Fig. 6.10(b).

---

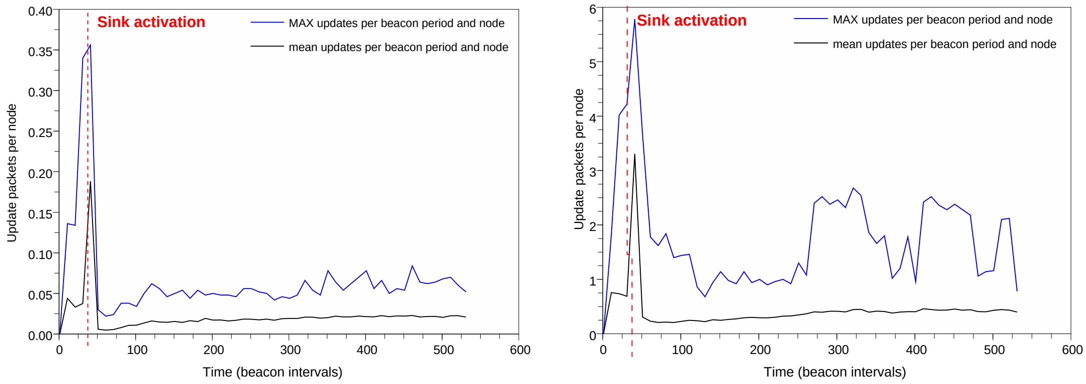
[13] 500 beacon periods after sink activation



(a) Integration and convergence time after sink activation

(b) The quality parameters

Figure 6.9: *Impact of the update probability p ($n = 5$, $m = 10$)*



(a) Updates per node and beacon period for $p = 0.6$

(b) Updates per node and beacon period for $p = 1.0$

Figure 6.10: *Update generation over time ($n = 5$, $m = 10$)*

The algorithm is running continuously for all the time. Even after the convergence of the network structure at the end the deployment phase, the nodes will generate some updates.

The graphs show the amount of update packets, which have been generated during and after the deployment. Prior to the activation of the sink ($t < t_{sink} = 33 \cdot T_{beacon}$) the updates are generated in order to build a structure rooted at the node with the minimal h_0 (temporary master - TM). After the sink has been powered on ($t > t_{sink}$) the structure has to change the direction of some links in order to point to the sink. When the mean distance reaches its minimum [14], the nodes generate only fewer updates building a background traffic.

To explain the presence of the background traffic it should be noted that the radio channel is time variant, and that a node may not react to an update, either because the transmitted packet is corrupted or because the node itself decides to discard the information.

Because of the ALOHA based generation of the update packets, it is likely that many

---

[14] $t_{conv} = (50.7 \pm 4.8) \cdot T_{beacon}$

Table 6.1: *Updates per node and beacon generated during the whole simulation*

	Since the start	After the convergence

$p = 0.6$	mean = 0.020
	stdev = 0.008
	max = 0.041
		mean = 0.010
		stdev = 0.007
		max = 0.027

$p = 1.0$	mean = 0.040
	stdev = 0.020
	max = 0.104
		mean = 0.020
		stdev = 0.014
		max = 0.070

updates collide when p approaches the value 1. The presence of collisions explains the much bigger values, which have been observed for $p = 1$.

The graphs in Fig. 6.10(a) and Fig. 6.10(b) are the result of the superimposition of all the traces recorded during the test series, so the uppermost curve represents the absolute maximum value recorded over all the runs during the respective beacon interval.

Given that each node generates a beacon packet per beacon period, the values in Tab. 6.1 represent an estimation of the additional costs. The results in table 6.1 and in Fig. 6.10(a) and Fig. 6.10(b) differ, because the data has been aggregated in a different fashion.

The first column reports the overhead due to the update packets exchanged during the whole simulation, while the second column shows only the overhead due to the update traffic after the end of the transient phase[15]. Because of the long operational life time of the network, the mean overhead due to the updates will approach the latter ($\sim 1-2\%$ of the overhead due to the generation of the beacon packets).

Presence of isolated nodes

In certain cases with pathological characteristics the proposed algorithm fails to integrate all the nodes into the routing structure. The relative frequency of these pathologic cases (with at least one isolated node) has been also investigated. The results given in Tab. 6.2 have been collected by selecting the parameters $p = 0.6$, $m = 10$ and considering 100 nodes distributed over 10 rooms.

As mentioned before, the work of Xue and Kumar [42] states that if $n > 5.2\ln(N)$, the network with N nodes is asymptotically connected. At the end of the paper, the authors postulate that there is a critical value c near to 1 such that $\forall n : n > c{\cdot}\ln(N)$ the probability of having a connected network approaches 1 even for small networks. It is reasonable to expect that the resulting structure behaves according to the observations of Xue and Kumar.

For 100 nodes the critical value of n should be around the value 5. The observations match with the expected values and for $n \geq 5$ we observe only connected networks. Test runs done with bigger networks suggest that for the particular structure we consider the critical parameter c is smaller than one.

---

[15] $t > t_{conv}$

Table 6.2: *Frequency of partially connected structures over 500 simulations, 100 nodes distributed over 10 rooms*

	Relative frequency of not connected network	Number of isolated nodes

$n = 3$	0.042	mean = 7.71
		max = 40

$n = 4$	0.004	mean = 5.5
		max = 10

$n = 5$	0	-
$n = 6$	0	-
$n = 7$	0	-

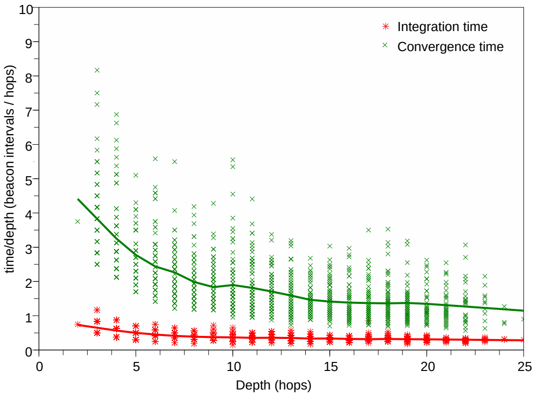


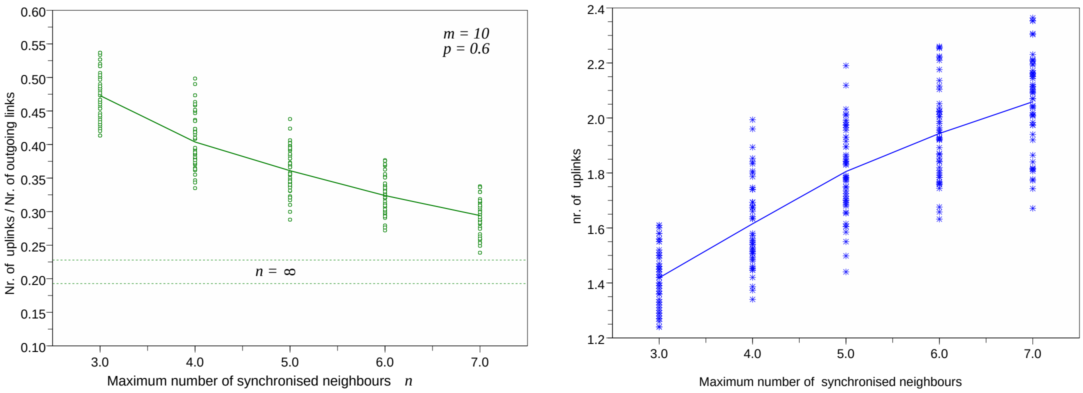
Figure 6.11: *Convergence time vs. depth of the tree structure ($n = 5$, $m = 10$, $p = 0.6$)*

Convergence as function of the depth

As said before, the proposed algorithm is expected to converge as fast as the BST algorithm (i.e. the time will be a linear function of the depth of the structure). The results plotted in Fig. 6.11 have been obtained during a simulation series where the number of rooms had been increased from 5 to 31. The figure shows the ratio between the measured time intervals (convergence and integration) and the depth of the tree. Excluding the cases where the network depth is really small and the precision of a beacon in the time measurement is too coarse, the integration time seems to be a linear function of the depth with a proportionality factor ~ 0.3 *beacon/hop*. Likewise the ratio between the convergence time and the tree depth seems to converge to the value ~ 1.3 *beacon/hop*.

Topological aspects

The ratio between the number of uplinks and the total number of outgoing links (UL-Ratio) has been plotted as function of the number of observed neighbors n (Fig. 6.12(a)) and as function of the mean number of uplinks (Fig. 6.12(b)). The UL-Ratio indicates the mean number of links a nodes could use to propagate the data towards the sink and hence indirectly the ability of distributing the data load between the nodes which



(a) UL-Ratio (b) Mean number of uplinks

Figure 6.12: *Topological observations* $(m = 10,\ p = 0.6)$

are nearer to the sink. The value (1- UL-Ratio) represents the number of links not pointing to upper nodes and hence useful only in order to improve the failure tolerance of the system.

It is interesting to note that for increasing values of n the UL-Ratio is decreasing and hence the increment of the mean number of uplinks is sublinear (Fig. 6.12(b)).

Currently, updates are only generated for nodes which have a distance of at least $h_{local} + 2$ (see Alg. 1). Including also the nodes with distance $h_{local} + 1$ would increase the UL-Ratio to 0.5. However, this would also increase the number of update packets by a factor of about ten and is therefore not recommended.

6.2.6 Further considerations

The author proposed and evaluated a heuristic algorithm for the initialization of an indoor WSN under the constraint of an extremely low duty-cycle.

He did not make any assumptions about the strategy used to collect the initial list of possible neighbors for the described scheme. The selection of a specific approach impacts mainly the total energy consumption and the convergence time, but not the quality and structure of the resulting network. The temporary usage of a higher duty cycle [54] and the periodical observation of small portions of the beacon periods are the two commonly implemented methods used to discover the neighbor nodes (see 6.1).

In a home automation scenario, it is possible to make use of an external device to provide each node with some initial information about the active neighbor nodes. This allows shifting the energy consumption for neighbor discovery away from the sensor nodes while keeping convergence time of the network low. This is a reasonable and realistic proposition because the deployment of the network is an assisted process where a field service technician installs the nodes at the predetermined positions. He can easily use a handheld device (mobile device - *MD*), which automatically communicates once and only once with a node as soon as it will be activated.

Based on this initial information each node runs a distributed algorithm autonomously in order to identify the neighbors belonging to a routing structure connecting the node and the sink.

The results show that the proposed algorithm works well in the given parameter range, which has been chosen based on the constraints imposed by a specific sensor node hardware. It converges fast, and it generates only an acceptable communication overhead. Implementing this heuristic algorithm on other systems, m and n should be chosen as big as the hardware and energetic constraints permit. Parameter n should not be smaller than 5 and m at least $n + 3$. For the parameter p it seems reasonable to suggest a value around 0.6 in order to keep the overhead resulting from the update packets low.

Even though the focus of this study was on the initialization of a newly deployed network, it is interesting to note that the proposed algorithm is able to permit the adaptation of the network structure to the problems which could arise during the expected life time, like modifications of the radio environment or presence of jamming signals. This aspect is covered in the following chapter 7.

Chapter 7

Network self configuration and self healing

A heuristic algorithm for neighbor discovery and routing initialization in a WMN has been described in the previous chapter 6. It allows the computation of the *Breadth-First Spanning Tree (BST)* used for routing purposes even in presence of unidirectional transmission links.

As stated at the end of section 4.4 a clear identification of the transition point between the different phases of the network operations is a difficult task. Because of that, it seems advantageous having algorithms which operate seamlessly in all phases.

This chapter is aimed at showing how the neighbor selection algorithm can be extended in order to provide continuous network optimization and self-healing capabilities by re-integration of temporarily isolated nodes or network segments.

7.1 An extension of the neighbor selection algorithm to address the self-healing requirements

7.1.1 Requirements and assumptions

Since the expected network life spans many years, the algorithms used during the normal operation and the failure recovery phase have to be aware of some additional constraints which might be neglected during the power-on phase.

In particular the characteristics of the primary power supply impose a lower limit d_{min} to the interval between two consecutive activity periods of the transceiver (especially when the batteries are almost depleted – see section 3.1.3).

Another constraint depends on the variability of the neighborhood relationship. Because of modifications in the radio environment, the set of possible communication partner changes during the time.

Disturbance signals might cause the temporary unavailability of some nodes. As consequence, even though a synchronization mechanism (see chapter 8) generates a common time base for the node operations, it can not be excluded that some nodes might lose the synchronization with the neighbors. Because of that the network has to be able of self-healing by reintegration of the isolated nodes.

At some point it might be necessary to add or replace some nodes, therefore it is also necessary to implement a mechanism which searches for new neighbors and which, at

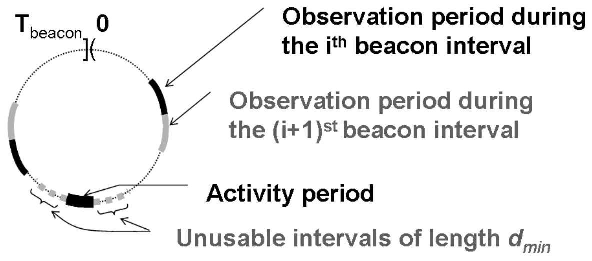

T_{beacon}
0
Observation period during the ith beacon interval
Observation period during the (i+1)st beacon interval
Activity period
Unusable intervals of length d_{min}

Figure 7.1: *Schematical description of the scan process with* $n = 3$ *and* $u = 1$

the same time, permits an adaptation to the changes in the surrounding environment.

Clearly, it is almost impossible to guarantee the availability of all nodes during the whole network life. After a node failure, the recovery algorithm has to permit the prosecution of the network activity for the surviving nodes – even in presence of multiple node failures.

7.1.2 Basic self-healing algorithm

Starting from the idea of having the neighbor selection algorithm active continuously, the following additional mechanisms have been selected to enable the network adaptation

Deletion of the stale neighborhood relationships A node has to be removed from the set (N) of the neighbor nodes an element is synchronized with, if its beacon has not been observed for a certain period $t_{inactive}$. If a node gets isolated ($N = \emptyset$), it has to reset its hop counter to the initial value h_0.

Verify the existence of a path to the sink If a group of nodes gets isolated from the sink, a count to infinity race for the logical distance of the isolated nodes starts. To avoid this, the concentrator generates a sequence number for each beacon it sends. Each node (except for the concentrator) stores the highest sequence number previously received from the uplink nodes and sends it with each own beacon. Each node resets its logical distance to the initial value if no new sequence number has been observed during the interval $t_{isolated}$.

Proactive search of new neighbors Energy permitting, each nodes has to search for previously unknown (and maybe better) neighbors to provide a continuous and fast adaptation of the routing structure to environment modifications. To find new neighbors, each node currently observing less than n neighbors scans the beacon interval by using $n - u$ (u is the number of already selected neighbors) additional short observation windows, evenly spaced over the beacon interval as shown in Fig. 7.1. The length w_{obs} of the observation windows corresponds to the longest feasible activity period. The observation windows are shifted by w_{obs} every beacon interval. As long as the distance between the boundaries of an observation window and the boundaries of one of the u scheduled activity periods is smaller than d_{min} the system does not make use of that observation window.

The maximum number of neighbors n a node synchronizes with has to be set taking in account the conflicting goals of robustness and energy efficiency. This parameter has been set to $n = 6$ for the simulation study as this values provides a reasonable compromise.

7.1.3 Adaptation capability of the network structure after convergence

When a sufficient amount of information about the surrounding nodes has been collected, the algorithm used to build the network (see chapter 6) converges quickly to a stable tree structure rooted at the sink. Each node is able to observe the beacon signals generated by at most n neighbor nodes. With respect to the distance from the sink only a subset of the selected neighbor nodes is useful for routing data as they have a smaller logical distance to the sink as the node itself. All other neighbors have at least the same distance as the node. Nodes belonging to the latter group play an important role for the fast adaptation of the network structure to changes of the environment.

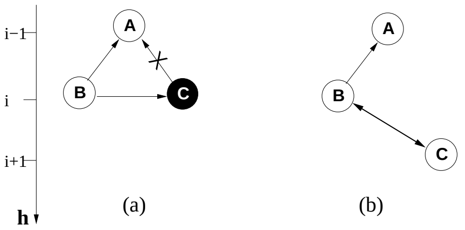


Figure 7.2: *Example: Node C loses the direct connection to A (a). Node B notices the event and generates an update which permits the reintegration of C (b)*

As an example we consider the situation shown in Fig. 7.2. The nodes of the directed neighborhood graph represent the network nodes while the edges represent the capability of receiving a beacon packet and hence the possibility of sending data to the beacon sender. At a given time the node C is no more able to receive the beacons sent by A (a). C removes A from the set of its neighbors and updates its distance estimation h. B receives a beacon from C, which carries the new distance estimation, and might answers[1] with an update frame because it is now closer to the sink than C. As a result, C selects B as new neighbor and finds its new position in the routing structure (b).

For more complex failure scenarios, the basic behavior of the algorithm is almost the same in all cases. As soon as a node (or a subtree) becomes unable to route data towards the sink, the distance estimation of the isolated element increases and eventually becomes larger than the distance estimation of a neighbor node. After the reception of an update frame, the isolated element or subtree finds its new place in the routing structure.

The continuous search for new neighbor nodes allows – on a longer time scale – the reintegration of nodes which can not take advantage of the previously described failover mechanism based on redundant *non routing*[2] links and at the same time provides a continuous optimization of the routing structure

---

[1] With probability $p < 1$.

[2] Links pointing to nodes which are not nearer to the sink.

The active search also permits complete network initialization by using only a limited amount of energy, noticeably lower than the option of a continuous channel observation during a whole beacon period. However, it requires much more time and hence may not be appropriate during the initialization in many cases.

7.1.4 Simulation Study

The behavior of the self-healing mechanism has been investigated by using the simulation framework described in chapter 5

The node model has been extended by including the synchronization protocol described in chapter 8 along with clock error models described there and the capability of scanning the beacon interval. The parameters m and p have been set to the values found to be reasonable in the previous chapter 6

Three different node placements have been considered. In the simplest case, all nodes, which have been placed in a single room, were able to directly communicate with each other. This placement has been used in particular to analyze the dependency of the setup procedure from the number of nodes.

As a second option, we considered the placement found in section 6.2.4 to represent a worst case: the nodes have been distributed over k aligned rooms and the concentrator (*sink*) has been located in the room k+1. The propagation parameters have been set such that the nodes in room i were able to communicate almost certainly with the nodes in the rooms $i \pm 1$ but there was no direct communication path to the elements in the rooms $i \pm j$ with $j \geq 3$.

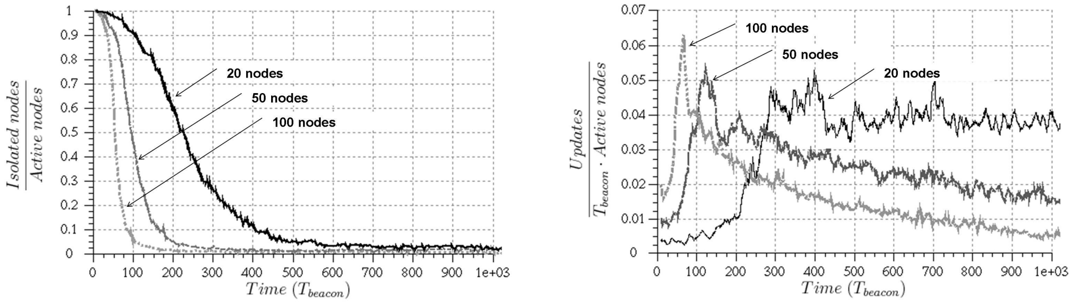
A third, more realistic placement has been obtained by distributing the nodes in a cubic structure consisting of R_z floors, each one with $R_x \cdot R_y$ rooms. The number of nodes in each room is selected randomly in the interval $[0 \ldots Max_{nodes}]$. The attenuation between rooms belonging to the same floor has been set like in the previous case, while the direct communication between different floors is possible only if the floors are adjacent. The sinks have been put alternatively in the middle or in a corner of the structure.

It should be mentioned here that direct comparison of the results collected using the three models is not meaningful because of the different propagation parameters.

Network setup

The results in chapter 6 assume that each node observes the radio channel for a complete beacon interval after its activation in order to collect the information concerning all the active neighbors. After this initial beacon period, it switches to the low duty cycle scheduling. This solution allows a fast network configuration at the expense of the energy required for the extended channel observation.

Although it is not likely, it should not be excluded that during the long network life a jamming signal disturbs the communication so massively that afterwards all nodes are isolated. In this case it is interesting to know whether the nodes are able to rebuild the network by themselves. Therefore, it has been evaluated first how much time is required to build the network by relying on the slow beacon period scanning mechanism only without increasing the duty cycle. Clearly the set-up time is a function of the total number of nodes in the communication range in this case. Simple computations suggest that it is inverse proportional to the number of neighboring nodes and observation windows.



(a) Fraction of isolated nodes vs. time for different number of nodes

(b) Average number of updates per node

Figure 7.3: *Network setup for different number of directly communicating nodes*

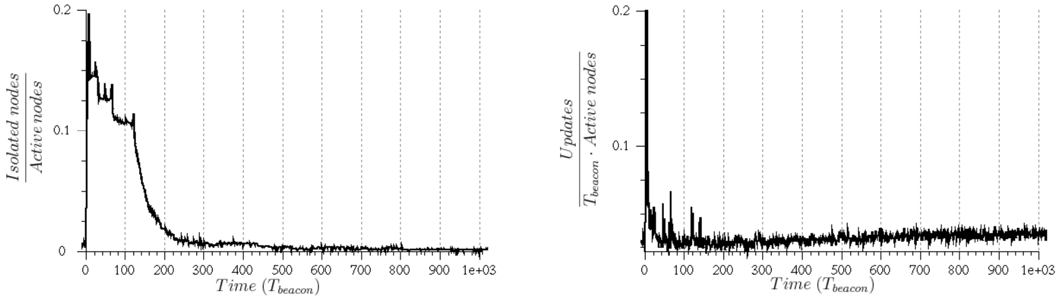
A simulation study has been done by putting a different number of nodes (20, 50 and 100 nodes) into a single room. The nodes have been activated in a random order during the first beacon interval. The curves (one for each network size) in Fig. 7.3 and in the subsequent figures have been obtained by calculating the mean value observed over 50 simulation runs for each measurement interval.

The time required to integrate 90% of the nodes into the routing structure is almost inverse proportional to the number of active nodes (Fig. 7.3(a)) as expected.

It is interesting to observe that the mean number of update packets generated per node during one beacon period shows a maximum corresponding to the maximum of the integration rate (Fig. 7.3(b)). This effect can be observed also during a failover phase and gives a hint on the basic behavior of the proposed algorithm. At the beginning the nodes join together to build some isolated clusters. As the nodes continue to discover new neighbors, the clusters grow and merge together until eventually all of the nodes belong to a big cluster. When the sink, – which is activated last – becomes part of that big cluster, the nodes increase the update generation rate in order to reconfigure the routing structure towards the sink and it is possible to observe the above mentioned effect. The nodes belonging to smaller isolated clusters have to wait longer before they discover some already integrated neighbors.

It can be estimated that by the time 90% of the nodes have been integrated into the routing structure, each node has discovered on average less than three neighbors by shifting the observation window, the rest of the neighbors has been discovered based on the update messages.

The total time required for the complete network setup spans a few days. Therefore, this setup strategy does not fulfill the requirements in section 4.2, which state that a verification of the network functionality has to be possible immediately after node deployment while the service technician is still on the premises. However, the self configuration properties observed in this experiment guarantee that the network is able to re-establish the full functionality even after a massive failure (e.g. due to the presence of massive jamming signals).



(a) Fraction of isolated nodes vs. time

(b) Average number of updates per node

Figure 7.4: *Network behavior if the 10 most important relay nodes fail. (100 nodes distributed over 5 aligned rooms)*

Self-Healing

It can not be ruled out that during the network life some nodes get isolated because of environment changes or because of the temporary presence of interference signals. As long as the service interruption is short, the nodes will be able to find the previous neighbors again once the cause of the disturbance is removed. However, after a long service interruption the nodes will lose the synchronization with their neighbors completely due to clock drift.

To check the self-healing capabilities of the network, we observed the network behavior in presence of severe outages. The quality metric introduced in section 6.2.4 allows to identify the most important nodes at each level of the routing structure, i.e. the nodes which have to transport the highest portion of the information flow. During a first simulation series 100 nodes evenly distributed over five aligned rooms have been considered. The propagation parameters have been chosen according to the worst case previously described. After the initialization phase the ten most important nodes directly connected to the sink (based on the metric mentioned above) were disabled resulting in many other nodes becoming isolated. After 120 beacon periods[3] the disabled nodes were activated again and the nodes started to reintegrate themselves into the routing structure. The results plotted in Fig. 7.4 show the superposition of the observations done during 50 runs.

Soon after the beginning of the disturbance ($t = 0$), a significant fraction of the isolated nodes was reintegrated into the routing structure by the mechanism described in Section 7.1.3. It is worth noting that in the graph of Fig. 7.4(a) the *number of isolated nodes* includes also the failed relay nodes which can not participate in the communication because of the presence of disturbing signals. The initial restructuring phase is characterized by an increased update generation rate (Fig. 7.4(b)). After that initial restructuring phase, other nodes were able to reintegrate themselves into the routing structure by finding new neighbors. After the disturbance had been removed, the reactivated nodes tried to find their position in the routing structure and the network recovered fully. Table 7.1 contains the measurements done during this test series.

---

[3] This interval has been chosen in order to break the node synchronization and to force the node search.

Table 7.1: *Duration of the failover process during and after the disturbances*

	During the presence of the disturbances			After the end of the disturbances		
	Time since the beginning of the disturbances (beacon intervals)		Success rate	Time since the end of the disturbances (beacon intervals)		Success rate
	mean	stddev		mean	stddev	
Aligned rooms	25.7	30.3	45/50	118.4	59.3	50/50
3D building (sink in the middle)	17.2	25.1	50/50	92.9	42.9	50/50
3D building (sink in a corner)	18.9	27.7	47/50	134.4	69.3	50/50

The graph in Fig. 7.4(b) shows the mean number of updates per node and beacon interval. This value increases as long as the failure recovery proceeds and then (as noted in 6.2.5) saturates to a constant value, which is upper bounded by 0.03 for the chosen parameters.

The same test has been repeated using the 3D node placement described previously. The playground consists of 60 rooms grouped in 5 levels with 3x4 rooms each. The number of nodes in each room was randomly chosen in the interval (0,8) at the beginning of each simulation run. The sink has been placed either in the center room or in a corner room. The propagation parameters have been chosen such that the nodes in a room were able to communicate almost certainly with the nodes in the adjacent rooms but the nodes could not communicate to nodes farther than two rooms away if the rooms belonged to the same level.

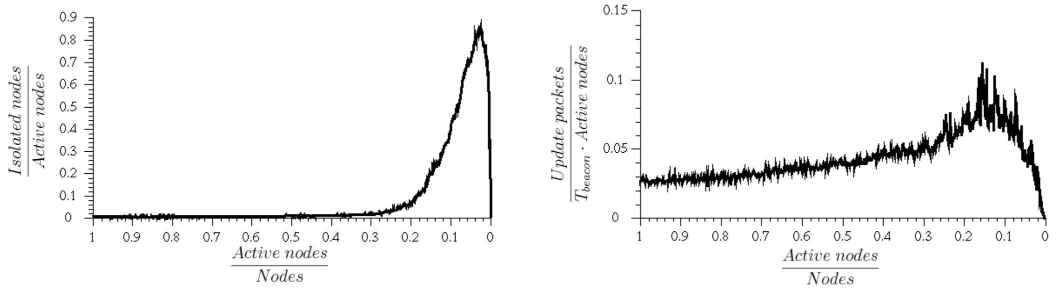
Each test run followed the scheme described previously. The observations show almost the same behavior as for the scenario with aligned rooms.

With respect to the data in Table 7.1, the failover period is defined as the time which is required to reintegrate all nodes able to exchange data into the routing structure. That means all nodes not affected by the disturbance during the first part of the test and all nodes afterwards. The intervals are measured starting with the activation (respectively deactivation) of the disturbance. Observing the field *success rate*, we note that in some cases the network structure did not integrate all active nodes during the disturbance period. This can be explained by observing that sometimes the damage is such that for some nodes there is actually no path remaining to the rest of the network.

By comparing the value for the mean distance from the sink metric[4] at the beginning and at the end of this test, even a slight reduction of this metric can be observed. This means that the nodes are forced to search for new neighbors in a failure case and on average they select some new neighbors which are nearer to the sink.

---

[4] This metric gives an indication of the energy consumption for the data transmission in absence of data aggregation (see section 6.2.4).



(a) Isolated nodes

(b) Updates per active node

Figure 7.5: *Network behavior in presence of a progressive erosion of the number of active nodes (3D building structure with the sink placed in a corner)*

Resiliency

A further failure scenario takes into account the progressive erosion of the number of active nodes due to energy depletion or malfunctioning.

The same networks used for the previous test have been stressed by progressively turning off all nodes during a five day interval. The node deactivation sequence has been randomized by selecting the deactivation time of each node in the given interval according to a uniform distribution. The curves plotted in Fig. 7.5 show the mean collected during the 50 runs which have been performed using the previously described 3D node placements with the sink placed in a corner. At a macroscopic level the network behavior resembles the theoretical results in the work of Srisankat et al. [59], but the different propagation model does not allow a direct comparison between the observations and the theoretical results in [59].

In particular, each time a node is failing the remaining nodes cooperate to reconstruct a connected network. However, at a given point this is no more possible and the network of the surviving nodes remains definitively partitioned as indicated in Fig. 7.5(a). This point has been defined as *breakdown point* and the fraction of surviving nodes at this partitioning time, has been measured. It is interesting to note that the breakdown point does not vary too much for the considered node placements (Table 7.2).

The graphic in Fig. 7.5(b) shows an increase of the update generation near to the breakdown point, and it is reasonable to assume that during this phase the node reintegration happens mainly due to the presence of redundant links. The active search for still active neighbors plays only a minor role because of the frequent node failures and the relatively low node density.

A further increment of the update generation rate can be observed, when the fraction of damaged nodes goes beyond the previously identified breakdown point. During this phase, many nodes use the updates to find their position into the isolated group they belong to. After having reached the maximum, the update generation rate decreases quickly in correspondence to a further node disappearing. During this last phase almost all nodes are isolated and there is no need of generating updates.

Since for the considered application the maximum acceptable network damage is

Table 7.2: *Connectivity breakdown in presence of a progressive network damage*

	Connectivity breakdown for the *network of survisor* ($Active\ nodes/Nodes$)	
	mean	stddev
Aligned rooms	0.18	0.08
3D building (sink in the middle)	0.25	0.12
3D building (sink in a corner)	0.23	0.08

in a range of about 10% of the total number of nodes, the results can be considered satisfactory.

7.1.5 Further considerations

The bootstrap algorithm analyzed in the previous chapter is particularly suitable for the implementation on WMNs due to its compactness and simplicity.

After the deployment and power-on phases, the network has to work unattended for many years. Its nodes have to be able to cope with modifications of the propagation characteristics, the presence of interfering signals and maybe failures of nodes. An extended version of the bootstrap algorithm has been analyzed in this chapter. It provides continuous optimization and failure recovery capabilities at reasonable complexity and energy costs. Moreover, the utilization of the same algorithm during the whole network life avoids the necessity of explicitly identifying the transition point between power-on, normal operation and failure recovery.

A simple mechanism for removing the stale neighborhood relationships and a search procedure based on the observation of the channel for a short interval each beacon period have been the only additions to the original algorithm required to fulfill the new requirements.

Simulation studies done with different node placements highlighted the self-configuration, self-healing and resiliency properties of a network running the modified algorithm.

The node capabilities with respect to processing, memory and energy constraints as well as the node placements and the propagation parameters were chosen based on a real field test scenario. Therefore, the author is confident that the results are representative for real installation conditions and that the proposed approach will also perform well in a real network.

Chapter 8

Time synchronization

The time synchronization is a typical problem in distributed systems. The nodes measure the time by using a local oscillator driven clock. Because of the non ideal behavior of the oscillator, the local time readings, done by different nodes, differ – they lose synchronization without correction.

The implementation of time synchronization algorithms on wireless sensor nodes requites to reconsider the well known approaches in order to accommodate new constraints like energy efficiency of the algorithm and the possibly huge number of nodes.

This chapter is aimed at introducing the synchronization problem, at analyzing the existing solutions and at showing an alternative approach proposed by the author [60]. This is based on the transformation of the original problem into an estimation problem which is then solved by using a Kalman Filter (KF). This algorithm is tailored to the implementation in Wireless Metering Networks (WMNs) which are characterized by an extremely low duty cycle and which adopt a beacon enabled MAC protocol.

8.1 The time synchronization problem – an introduction

Time plays an important role in the operation of distributed systems. On one side applications might rely on the presence of accurate time information in order to carry out their tasks, as example because an event consists of the combination of the measurements taken by different sensors [61]. On the other side the network protocols themselves might require the existence of a common time base, like in the case of Time Division Multiple Access (TDMA) based Medium Access Control (MAC) protocols or in presence of coordinated wake up.

When speaking of time synchronization, it is important to note that the problem can be observed from different points of view, depending on whether the time measured by the nodes adheres loosely or tightly to the physical time. In the former case two nodes have to agree on a common idea about the duration of a given interval (*Internal Synchronization*), while in the latter case there is the additional constraint that the agreed interval measurement has to adhere to the measurement of the same interval done in the global time reference[1] (*External Synchronization*). The existence of a node

---

[1] The duration of a second is precisely defined in terms of the number of transitions between two energetic levels of an atom of Cesium-133. A network of atomic clocks spread around the world provides a frame of reference for the time measurements. The operators of the different atomic clocks

able to provide the UTC time is a precondition for the external synchronization.

All the clock devices implemented in the different sensor nodes share an almost identical structure. There is a *hardware clock* consisting of an oscillator (clock source), working at a specific frequency, and a counter register which is incremented in hardware after a certain number of oscillator transitions.

The relationship between the global clock t and the local HW-clock can be represented by a function $H(t)$.

An affine transformation is computed on $H(t)$ in order to obtain the local *software clock*

$$S(t) = \theta \cdot H(t) + \psi \tag{8.1}$$

where θ and ψ are called *drift rate* and *phase shift*, respectively. Most of the clock synchronization algorithms act on the tuple (θ, ψ) in order to compensate the discrepancy between the local clocks and the reference clock.

The parameter ψ is chosen in order to correct the difference between the local clock and a reference clock at the synchronization time (time offset), while θ is chosen in order to compensate the first derivative of the synchronization error (frequency offset).

8.1.1 Origin of the frequency offset

It is easy to identify the origin of the frequency offset in the analog HW components. The clock sources used in a sensor node, which actually can be implemented using different kinds of circuits, can be divided in two classes: those based on mechanical resonant devices, such as crystals and ceramic resonators, and those based on electrical phase-shift circuits such as RC (resistor, capacitor) oscillators.

The members of the second class are cheap, have low energy consumption, fast startup and can be entirely built in an integrated circuit. Unfortunely, they suffer from poor accuracy over temperature and supply voltage, and show huge variations from the nominal output frequency.

Crystal and ceramic resonator based oscillators offer a much higher accuracy at the price of higher costs, higher energy consumption and longer startup times. However, even the behavior of these components is not ideal. The accuracy of the generated signal is influenced, among other things, by temperature, humidity and aging process of the resonator.

The behavior of crystal and ceramic resonators is similar even though the crystals offer higher accuracy. Crystal based frequency sources have been widely analyzed in the past. A brief but clear treatment of the principle characteristics can be found in [62].

The core of a crystal resonator is the *crystal unit*. It is a thin quartz[2] slice to which electrodes have been applied. It builds a mechanically vibrating system that is linked via the piezoelectric effect to the electrical world.

Because quartz is piezoelectric, a voltage applied to the electrodes causes the quartz plate to deform slightly. The amount of deformation due to an alternating voltage depends on how close the frequency of the applied voltage is to a natural mechanical resonance of the crystal.

---

work together in order to provide the international Coordinated Universal Time (UTC) time scale. The UTC time is derived from the indication of the atomic clocks, by occasionally adding some small corrections in order to keep the time indication synchronous with the astronomical time. The UTC can be considered a representation of the real time.

[2] Quartz is a single-crystal form of S_iO_2.

The resonance frequency of a quartz unit varies with the temperature. A commonly used quartz shows a frequency variation of $\pm 25ppm$[3] for a temperature range of $-55°C$ to $+85°C$.

The word aging is used to identify "the systematic change in frequency with time due to internal changes in the oscillator" [62]. The primary causes of this process are the mechanical stress in the mounting structure, mass transfer from or to the quartz element and changes of the quartz material. The resonance frequency changing rate can be estimated to about 5 *ppm* to 10 *ppm* per year for a commonly adopted quartz.

Moreover, the tolerances inherent in the component manufacturing process are responsible for a deviation of the oscillator frequency from its nominal value, which in many cases might be estimated to be almost uniformly distributed in the interval ±20 *ppm*.

There are other causes of clock inaccuracy, but their contribution is negligible in comparison to the three described effects.

8.2 An overview of existing synchronization mechanisms

As stated before, time synchronization is a standard problem for distributed systems. It has been addressed with many different approaches. The Network Time Protocol (NTP) [63, 64] represents a milestone in this research area. It provides an external synchronization mechanism for all Internet clients and, in such a way, implicitly defines a stable time reference for the whole Internet.

NTP and similar protocols can be also used to provide synchronization between sensor nodes, but the behavior is not optimal because of the peculiarities of the Wireless Sensor Networks (WSNs).

New protocols have been proposed especially for the WSNs. These explicitly address the low energy requirements, the large number of nodes, the topology dynamic, the presence of unidirectional channels and other aspects which make the implementation on WSNs so challenging.

8.2.1 The Network Time Protocol (NTP)

The first version of this protocol has been adopted as recommended internet standard in the year 1989.

The standard considers the existence of a layered overlay network where:

- The uppermost layer (*stratum 1*) consists of a limited number of reference time server having access to a reference clock providing the UTC time.
- A second layer (*stratum 2*) consits of a much bigger number of secondary servers directly synchronized with some elements of the stratum 1.
- Most of the nodes occupy the following *stratum 3*. It contains more than 50% of all nodes.
- The nodes occupying the levels deeper than 3 are no more than 20% of all nodes.

The behavior of the protocol is sketched in Fig. 8.1. A node **A** sends an NTP packet including a timestamp (T_{i-3}) to an NTP server **B**. **B** timestamps the packet upon

---

[3] This is the frequency error normalized to the frequency ($\frac{\Delta f}{f}$). A frequency error of 25ppm corresponds to a clock error of about 2.2s per day

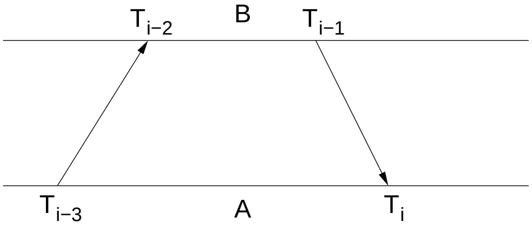


Figure 8.1: *NTP data exchange and timestamps [63]*

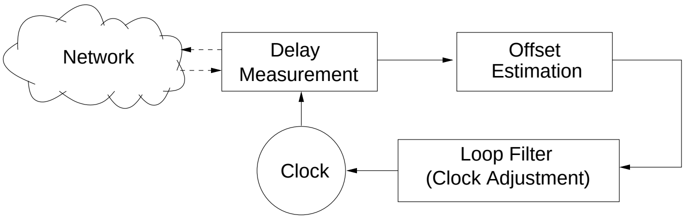


Figure 8.2: *Logical structure of NTP adapted from [63]*

reception (T_{i-2}), copies the two timestamps into a reply packet, adds a new timestamp (T_{i-1}) and sends the packet back. The node **A** records the arrival timestamp (T_i).

T_{i-3} and T_i are generated by the clock in **A** while T_{i-2} and T_{i-1} refer to the clock in **B**. If the propagation and transmission delay are constant in the two communication directions, it is possible to estimate the offset (δ_i) between A and B and the round trip propagation and transmission delay (ρ_i).

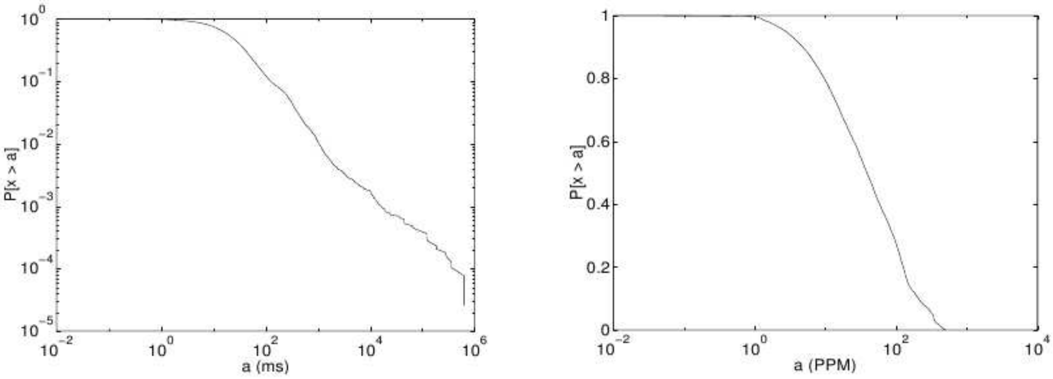
$$\delta_i = \frac{T_{i-2} - T_{i-3} + T_{i-1} - T_i}{2} \quad \text{and} \quad \rho_i = T_{i-2} - T_{i-3} - (T_{i-1} - T_i) \qquad (8.2)$$

The true offset between A and B lies in an interval of size ρ_i centered at δ_i. The offset estimation quality is improved by combining the measurements done in successive data exchange phases. The resulting offset estimation is then processed by a Phase-Locked Loop (PLL) based filter in order to obtain a new tuple (θ_i, ψ_i) which minimize the clock error. The values are provided to the clock used to generate the timestamps.

As noted in [65] the accuracy depends on the network latency. It is typically in the order of few milliseconds in a LAN, varies between 10ms and 100ms in WAN and reaches a big unpredictable value when the nodes are spread over the global Internet. The implemented data filter is characterized by a quite long transient phase, which in some cases can last for many hours.

In the 1997 Mills [66] analyzed the accuracy of NTP using a large set of time servers spread over the whole Internet. The median of the time offset was about 21 ms and the median of the frequency offset was about 38 ppm. The cumulative distribution function (CDF) of synchronization errors (time offset) and frequency offset observed by Mills are plotted in Fig. 8.3.

It is clear that periodic resynchronization is needed in order to keep the error bounded. The work of Mills considers a synchronization period of 50s.



(a) Time offset

(b) Frequency Offset

Figure 8.3: *NTP synchronization errors – cumulative distribution functions (CDFs) [66]*

8.2.2 WSN synchronization algorithms

Even though many of the previously developed synchronization algorithms can be adapted to work in a WSN, these show only suboptimal performance. Recently huge efforts have been made in order to develop new algorithms which address explicitly the low energy consumption requirements and the high network dynamic.

In particular, WSNs show some peculiarities which have to be considered for the development of a synchronization strategy.

- The network might scale to a large number of nodes.
- The nodes are severely energy-constrained and unreliable.
- The MAC frame delivery delay might show huge variations (e.g. because of ARQ error correction).
- The propagation delay is negligible[4].
- The error associated with the timestamp generation (time difference between the reception of the last bit and the generation of the timestamp) is not negligible. Moreover, it depends on the protocol layer which is responsible for the timestamp generation.

All proposed algorithms can be divided in two classes according to the main goal. The first class includes those solutions aimed at promoting an agreement on a common time base only upon the appearance of a significant event. The *Post Facto Synchronization* is achieved with algorithms working on-demand and hence particularly energy efficient. In this case, the nodes work unsynchronized most of the time and synchronize sporadically only just after the detection of an important event. An example of these protocols can be found in [67].

The second class consists of all the algorithms aimed at providing continuous synchronization between the nodes. Since the considered low duty cycle MAC (see section 4.3.2) requires the implementation of a continuous time synchronization, only those protocols providing continuous synchronization will be analyzed.

---

[4] If $30\,m$ is an upper bound for the distance between neighboring nodes, the mean propagation delay is upperbounded by $0.1\,\mu s$. It is much shorter than the bit duration

Sender/Receiver synchronization

Some protocols base their functioning on the conventional approach used by NTP. They assume all links to be bidirectional and rely on a data exchange similar to the one used by NTP.

Lightweight Time Synchronization protocol (LTS) [68] and Timing-sync protocol for sensor networks (TPSN) [69] are two examples of this class of protocols. Both protocols establish a pair-wise synchronization between neighboring nodes and then a network-wide synchronization by propagating the synchronization information along a spanning tree rooted at the reference node. The two protocols differ mainly in the strategy adopted for the construction of the spanning tree and in the computations done on the collected timestamps.

Receiver/Receiver synchronization

Other protocols leverage the intrinsic broadcast characteristics of the propagation medium to achieve the synchronization among all those nodes which receive the same packet. They do, however, not care about synchronizing the receiver nodes with the sender of the same packet.

The Reference Broadcast Synchronization (RBS) protocol [70] provides a mechanism for the synchronization of the nodes belonging to the same broadcast domain and a strategy to propagate the data over several broadcast domains towards the remote nodes.

The basic idea consists of having a sender which periodically sends a (not necessarily timestamped packet) into a broadcast channel. All receivers timestamp the packet arrival. These timestamps are then exchanged between all receiver nodes which use the information to compute phase offset and drift rate for each neighbor.

Instead of adapting the own clock parameters, each node stores the computed values in a table and uses these to convert between the own and the neighbor's clock.

This approach agrees with the observations of Romer and Elson in [71] which highlight the importance of creating an operational clock on top of the local free running clock instead of trying to modify the clock parameters.

It is interesting to note that in this case the timing error due to the packet generation process is irrelevant and the timestamping error at the receiver is small. Elson et al. [72] did an extensive analysis of the reception time error by using real nodes operating at 19.2 Kbps. They found out that the error is almost normal distributed with zero mean and a standard deviation of about $11\,\mu s$ (much smaller than the symbol duration).

A common agreement on one timestamp is then obtained by successively converting the timestamp at each node into the next hop node's timescale.

An alternative approach

Parallel to the previously described mechanisms Hu and Servetto [43] developed a new approach to the synchronization problem which approaches the optimum for increasing node density.

The authors start from the consideration that the process of providing a common time scale to all nodes by periodic data exchange generates a non negligible overhead for increasing network density[5]. Moreover, the synchronization error increases in a multihop network with the square root of the distance from the reference node.

Instead of relying on a pair-wise synchronization, the proposed method tries to obtain a common agreement on a "virtual clock", by observing the result of the superimposition of the signals generated by all other nodes

They assume the existence of a reference node (*node 1*) having a clock c_1 and an operational counter $S_1(t)$, which is incremented at integer values of c_1. The goal of the algorithm consists in providing each node (i) with a local copy of the operational clock $S_i(t)$ which increments at the same time as $S_1(t)$[6] independently from the local clock c_i.

The synchronization happens periodically. During each synchronization period, the *node 1* starts the generation of a square wave signal having the rising flanks (from -1 to +1) in correspondence with the integer values of c_1.

Each node observes a convenient number of transitions between -1 and +1 of received signal and estimates the position of the zero crossing point by using a static Bayesian estimator. Afterwards, the node starts the generation of a square wave signal raising from -1 to +1 in correspondence to the estimated zero crossing point (Fig. 8.4).

The raising flank or the signal resulting from the superimposition of all the square waves crosses zero almost exactly in correspondence to the integer values of c_1.

In order to understand the behavior of this synchronization algorithm, it has to be noted that the estimation errors are almost normal distributed with zero mean and bounded variance. Moreover, the detection of the zero crossing point of the sum of all waves is equivalent to computing the arithmetic mean of the estimations done by the single nodes. If the errors are independent, the difference between the zero crossing point of the combined wave and the increment point of the reference counter $S_1(t)$ will converge to zero if the number of superimposed signals approaches infinity because of the *Law of Large Numbers.*

This approach does not rely on the exchange of additional traffic, it uses the broadcast characteristics of the physical medium and it relies on local estimations for the emission time of synchronization pulses. In this case, the synchronization takes the form of an estimation problem. The authors show the optimality of the proposed approach if the number of nodes approaches infinity. An application to a network with a finite number of nodes is described in [73]. The algorithm used seems to be instable in networks of finite size and requires the introduction of a feedback from the reference node in order not to diverge.

A critical aspect of the whole concept is the assumption of the wide sense stationarity of the process describing the measurements. In a real scenario, modifications of the environmental parameters have influence on the parameters of the stochastic process which describes the measurements. The process itself will, therefore, be no more stationary. A continuous adjustment of the parameters of the used estimator may solve the problem but it would make the algorithm particularly inefficient (and maybe infeasible) on the low-resource nodes used in the considered WMN (see 4.2), e.g., because an update of the used estimator requires the computation of a matrix inversion.

---

[5] E.g. RBS requires the exchange of a number of synchronization messages proportional to the second power of the number of nodes.

[6] In the random case the algorithm tries to minimize the square synchronization error.

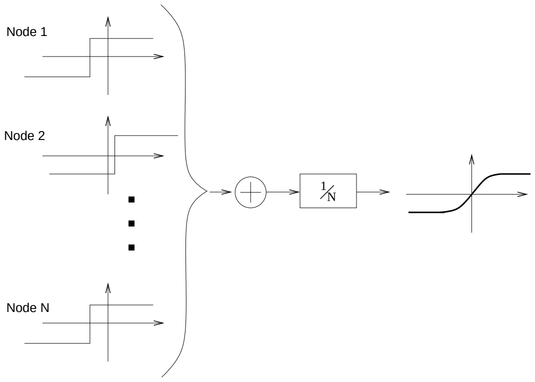


Figure 8.4: *The effect of wave superimposition in [73]: The zero crossing point of the resulting signal provides the mean of the estimations carried out by the single nodes.*

8.3 Opportunistic synchronization in a beacon enabled networks

This section contains the description of a synchronization algorithm which has been developed by the author specifically for the implementation on WMNs.

The following considerations apply to a network characterized by the constraints described in section 4.2, whose nodes communicate by using the MAC protocol described in section 4.3.2. However, since the existence of a beacon enabled MAC protocol, the presence of a hierarchic routing structure and the adoption of a low power / low duty cycle activation strategy represent quite general assumptions common to many WSN projects, the author feels confident about the applicability of the proposed solution to a wide class of problems.

The network initialization and optimization procedure defines implicitly which nodes are able to receive the beacon signal a specific node generates, and hence the direction of the communication between neighbor nodes (see chapter 6).

The assumptions of a beacon based MAC protocol and of a tree-like routing structure with unidirectional links apply to a large class of WSNs. Therefore, the proposed synchronization method is generally applicable to many WSN scenarios - irrespective of the use of the specific MAC layer and the specific bootstrap algorithm.

Each node is equipped with a free running clock, which is characterized by a precision ϵ_{clock} due to the manufactory process. Environmental changes, e.g. temperature changes, may impose additional clock errors.

In order to accommodate the synchronization errors, each node activates the receiver prior to the expected beacon emission time to provide a guard period.

8.3.1 Problem description

The following considerations apply to a sensor network consisting of a finite number of nodes M. Each node is equipped with a *free running* clock source according to the recommendations in [71]. Instead of adjusting the parameters of the local clock, an *operational clock* is built on top of the free running local clock, like in [43].

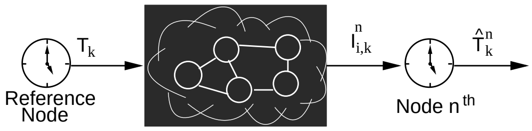


Figure 8.5: *Schematic representation of the blackbox approach*

A simplified clock error model indicates that at time t, measured according to a hypothetic absolute time reference, the clock of the node i indicates the time

$$c_{i,t} = (1 + \delta_i) \cdot (t - t_0) + \Delta_{i,0} + \Psi(t) \tag{8.3}$$

where $\Delta_{i,0}$ is the initial clock offset at the time t_0, $\delta_i \in [-\epsilon_{clock}, \epsilon_{clock}]$ is a constant frequency offset and $\Psi(t)$ is an additional error which takes into account the dependency of the clock on the environmental conditions.

The primary goal of the synchronization algorithm is to force all nodes to have the same beacon interval in order to minimize the length of the guard period. From this perspective, global clock synchronization is only a side effect of the suggested procedure. If a node is chosen as reference, all other nodes have to adjust the own beacon interval to the one of the reference node.

Each node generates periodically (every T_{beacon}) a beacon signal in order to indicate the beginning of a MAC frame (see section 4.3.2). The beacon interval is measured using the internal clock and hence it differs from the nominal value.

A node is able to measure the beacon generation interval for each one of its neighbor nodes. The error due to signal propagation is much lower than the measurement error, which is upper bounded by the duration of a symbol on the physical layer. The measurement error is negligible in comparison to the duration of a beacon interval.

The sink provides the gateway to a public network (e.g. GSM or UMTS) for remote storage and processing of the measurement data by the WSN operator. Therefore, it will be connected to the regular power supply and also be more powerful. Therefore, we assume that the sink is equipped with a better clock and hence we want that all nodes generate the beacon signal at the same rate the sink does.

8.3.2 Proposed WSN synchronization algorithm

The synchronization problem can be viewed as an estimation problem. A node n tries to estimate the beacon interval of the reference node by using the measurements of the beacon interval of all observed neighbor nodes ($I^n_{i,k}$ denotes the measurement of the k-th beacon interval of the node i, taken by the node n).

The reference node generates its beacon signal periodically, exactly each $T_k = T$ seconds. The measurements done by the node n are affected by additive errors $\xi^n_{i,k}$.

At the time k, the node n computes the estimation of the reference interval $\hat{T}^n_k$ at the time k by using the previous observations and uses this value as length of the next beacon interval.

If the network between the reference node and the node n is viewed as a blackbox (Fig. 8.5), it is possible to write the following relation (8.4) between the reference interval and the measurement taken by the node n[7].

---

[7] In order to improve the readability of the formulas, the indexes n and i are omitted from the notation.

First the equations are derived for a node n having only one neighbor, then a simple extension is introduced if the node observes the behavior of more than one neighbor.

$$\begin{cases} T_{k+1} = T_k \\ I_k = T_k + \xi_k \end{cases} \tag{8.4}$$

The error ξ_k is the difference between the measured beacon interval and the beacon interval of the reference node. If the quantity ξ_k can be modeled as Additive White Gaussian Noise (AWGN), the optimal estimation of T_k can be easily carried out with a Kalman Filter (KF - see A). In general, however, ξ_k is not AWGN.

A similar problem had been addressed in [74] for estimating the generation rate of ATM cells on the basis of the observation of their arrival time.

However, it has to be noted that the origin of the measurement errors, and hence the characteristics of the error process, differs in the two cases. In the cited paper, ATM cells arrive with variable delays, which depend mainly on the queuing process in the intermediate nodes, while, in the considered case, the measurement error ξ_k is mainly due to the adjustment process of the beacon generation time in all nodes between n and the concentrator and might experience sudden variations. In spite of these differences the author argues that the prefiltering approach is useful in order to smooth the measurement error and make the problem tractable.

If ξ_k is AWGN the following approach is equivalent to the one proposed in [43] where the estimation of impulse generation time is carried out using a *maximum likelihood estimator*.

As first step the signal ξ_k is smoothed and transformed it into a low pass signal. Then the low pass signal is modeled with an autoregressive process of the first order (AR(1)),which is one of the simplest models with low pass characteristics.

The smoothing filter uses the moving average of the last N observed intervals to smooth the error components.

$$\bar{I}_k \;=\; \frac{1}{N}\sum_{i=0}^{N-1} I_{k-i} \;=\; T_k + \frac{1}{N}\sum_{i=0}^{N-1}\xi_{k-i} \;=\; T_k + \bar{\xi}_k \tag{8.5}$$

Now the eq. 8.4 can be rewritten by using eq. 8.5.

$$\begin{cases} T_{k+1} = T_k \\ \bar{I}_k = T_k + \bar{\xi}_k \end{cases} \tag{8.6}$$

As stated before it has been postulated that $\bar{\xi}_k$ can be modelled as AR(1) system as follows:

$$\bar{\xi}_k \;=\; a\,\bar{\xi}_{k-1} + r\,w_k \quad ; \quad w_k \in \mathcal{N}\,(0,1)\ \ i.i.d \tag{8.7}$$

The following step consists of writing the equations of the Kalman Filter (KF) for the given model with the colored noise described by eq. 8.7

By using the approach described in Appendix A.2, the eq. 8.6 can be rewritten in the following form

$$\begin{cases} T_{k+1} = T_k \\ \bar{D}_k = (1-a)\,T_k \,+\, r\,w_k \;=\; H\,T_k \,+\, r\,w_k \end{cases} \tag{8.8}$$

whith $\bar{D}_k \equiv \bar{I}_k \; - \; a\bar{I}_{k-1}$ and $H \equiv 1 - a$.

For this system the equations of the KF take the following form

$k \geq 1$

$$\begin{cases} G_k = \frac{H\, P_{k-1}}{H^2\, P_{k-1} + r} \\ P_k = (1 - H\, G_k)\; P_{k-1} \\ \hat{T}_k = \hat{T}_{k-1} + G_k\, \left(\bar{D}_k - H\, \hat{T}_{k-1}\right) \end{cases} \tag{8.9}$$

with the initial conditions $(k = 0)$[8]

$$\begin{cases} \hat{T}_0 = m_{T_0} - \frac{\sigma^2_{T_0}\, (m_{T_0} - \bar{I}_0)}{\sigma^2_{T_0} + r} \\ P_0 = \left(\left(\sigma^2_{T_0}\right)^{-1} \; + \; r^{-1} \right)^{-1} \end{cases} \tag{8.10}$$

Where m_{T_0} and $\sigma^2_{T_0}$ are the expectation and the variance of T_0, respectively, $\hat{T}_k$ is the estimation of T_k, $P_k$is the variance of the estimation error $\hat{T}_k - T_k$ and G_k is the gain of the KF.

Now, in order to compute the KF, the values of m_{T_0} and $\sigma^2_{T_0}$ have to be determined for each node[9], along with the parameters of the autoregressive model $(a,\, r)$.

The following description suggests how the first and second order moments can be derived from the a priori knowledge about the clock and how the tuple $(a,\, r)$ can be estimated by using a small number of measurement collected before starting the synchronization procedure.

Mean and variance - Assuming that for each node n, including the reference node (*node* 0), the frequency f_n of the oscillator driving the clock is independent and identically uniformly distributed in the interval $[(1 - \epsilon_{clock})\; f_{nom}\;,\; (1 + \epsilon_{clock})\; f_{nom}]$, where the precision ϵ_{clock} depends on the manufacturing process, the beacon duration at the node 0 is $T_0 = T_{beacon} f_{nom}/f_0$. Moreover, the measurements of this interval taken by using the clock of the node n can be written following the same reasoning as in [74] as

$$T_0 = T_{beacon} \frac{f_n}{f_0} \tag{8.11}$$

Its expectation is

$$\begin{aligned} m_{T_0} &= E\,[T_0] \;=\; E\left[T_{beacon} \tfrac{f_n}{f_0}\right] \;=\; T_{beacon} E\,[f_n]\, E\left[f_0^{-1}\right] \\ &\approx T_{beacon} \tfrac{1 + 2\epsilon_{clock}}{1 - \epsilon^2_{clock}} \\ &\approx T_{beacon} \end{aligned} \tag{8.12}$$

The last approximation is possible because $\epsilon^2_{clock} \ll \epsilon_{clock} \ll 1$.

---

[8] The estimation $\hat{T}_0$ is the one which minimizes the uncertainty on T and P_0 is the corresponding variance of the estimation error.

[9] These values refer to the measurements taken with the local clock.

The same reasoning can be applied to the computation of the variance

$$\begin{aligned}\sigma^2_{T_0} &= E\left[T_0^2 - E\left[T_0\right]^2\right] \;=\; T^2_{beacon}\left\{E\left[f_n^2\right]E\left[f_0^{-2}\right] - E\left[f_n\right]^2 E\left[f_0^{-1}\right]^2\right\}\\ &\approx T^2_{beacon}\left\{\frac{\epsilon^2_{clock}+3}{3(1-\epsilon^2_{clock})} - \frac{1}{(1-\epsilon^2_{clock})^2}\right\}\\ &\approx 2/3\left(T_{beacon}\,\epsilon_{clock}\right)^2\end{aligned} \tag{8.13}$$

AR(1) parameters - The stored measurements are used in order to compute an estimation of the auto covariance coefficients c_0 and c_1. By applying the covariance method we obtain $a = c_1/c_0$ and $r^2 = (1-a^2)c_0$

If a node observes the beacon signals generated by different neighbors, it uses the mean value of the measurements collected during the last beacon interval. This approach fits with the idea of considering the network like a blackbox. Ignoring the quality of the different measurements, the errors affecting each sample can be considered identically distributed and with the same variance. In that case, the estimator which minimizes the square error coincides with the mean of the samples.

The continuous adjustment of the beacon generation interval modifies the model parameters an observer perceives. The system has to adapt itself to these changes and in particular it may be necessary to reinitialize the KF, as noted in [74], where the author suggested a continuous observation of the buffer state in an ATM switch in order to determine when the network condition changed and KF reinitialization was needed.

The proposed synchronization strategy tries to follow the modification of the system parameters by continuously adjusting the parameters of the AR(1) model. At the same time the mean absolute synchronization error (absolute value of the difference between the arrival time of a beacon and the corresponding expected value) is evaluated continuously and the KF is reset if this value starts growing[10].

An alternative approach based on using a second KF for the estimation of the model parameters seems to converge slower.

8.3.3 Simulation

The behavior of the proposed synchronization algorithm and its sensitivity to the parameter variations have been studied using simulation. A model of the proposed algorithm has been implemented using the Omnet++ [48] simulation library, along with different models of the free running clock. Each node measures the time intervals using the own clock which is characterized by a precision in the order of the duration of a symbol (10 μs for the analyzed network).

Clock models

If the frequency offset is not constant, the time perceived by the i^{th} node can be expressed by the following equation

$$c_{i,t} = \int_{t_0}^{t}(1+\delta_i(x))dx + \Delta_{i,0} + \Psi(t) \tag{8.14}$$

---

[10] As long as the error is bigger than a given threshold.

where t is the time measured by a hypothetical reference clock, $\Delta_{i,0}$ is the difference between the local and the global clock at time t_0 and $\delta_i(t)$ is the first derivative of the difference between the frequency of the local and the ideal oscillator. $\Psi(t)$ takes into account all other error components.

In order to identify the behavior of the proposed algorithm in response to different error conditions, the author implemented the following four clock models:

Frequency offset (*FreqOff*) - The model considers only the existence of a constant frequency offset which is, however, individual for each node: $\delta_i(t) = \delta_i \in [-\epsilon_{clock}, \epsilon_{clock}]$ where the clock tolerance ϵ_{clock} depends on the manufacturing process ($\epsilon = 40\,ppm$ for the simulation study). The additional error component $\Psi(t)$ consists only of the quantization error and is upperbounded by the symbol duration ($10\ \mu s$).

Aging - Since the expected lifetime of the considered network spans over many years, the effects on the clock stability due to the aging process (see 8.1.1) have been modeled assuming the frequency offset to be a linear function of the time ($\delta_i(t) = \delta_{i,0} + \rho_i \cdot t$). For a commercial quartz ρ may range between ±5 and ±10 ppm/year [75]. ($\rho_i \in [-10^{-4}\,ppm/s, 10^{-4}\,ppm/s]$ for the simulation study).

Environmental changes (*EnvChange*) - Environmental parameters and in particular the temperature influence the frequency of a quartz driven oscillator. The nodes of an indoor network may experience significant temperature variations over a relatively short time interval, especially during the cold season if the nodes are placed close to a radiator typical for e.g. a heat meter application. The author tries to reproduce this effect by means of a simple Markov model with two states (H,L). The transitions between the two states happen with rate λ_{HL} and λ_{LH}, respectively. the frequency offset is constant as long as a node does not change the state $\delta_i(t) = \delta_{i_{base}} + \delta_{i_{state}}$, where $\delta_{i_{state}}$ is specific for each state. It has to be noted that, from the point of view of the synchronization mechanism, this behavior is worse than the real case because it introduces a discontinuity in the frequency offset. ($\lambda_{HL}^{-1} \in [500s, 1000s]$, $\lambda_{LH}^{-1} \in [1000s, 2000s]$, $\delta_{i_H} \in [-40ppm, 0]$, $\delta_{i_L} \in [0, 40ppm]$, $\delta_{i_{base}} \in [-40ppm, 40ppm]$ are the parameters used for the simulation runs).

Jitter - In order to stress the synchronization algorithm the presence of a random clock jitter superimposed to a constant frequency offset has been considered. The additional jitter error component ($\Psi(t)$) is modeled as a Poisson process with intensity λ_{jitter} and normal distributed amplitude $A_j \in \mathcal{N}(0, \sigma_{A_j}^2)$ (for the following simulation runs $\lambda^{-1} \in [500s, 2000s]$ and $\sigma_{A_j} = 0.01s$).

All four clock error models have been used in the simulation study to assess the behavior of the proposed algorithm. The error model assuming random clock jitter systematically provided the worst case results. Therefore, the following description is mainly focused on this error model. The observations done with the other error models are reported only if specific effects have to be described.

Metrics used for the evaluation

As stated before, the aim of this study was finding a way to reduce the guard period and hence to increment efficiency of the MAC Layer protocol. In order to get a metric

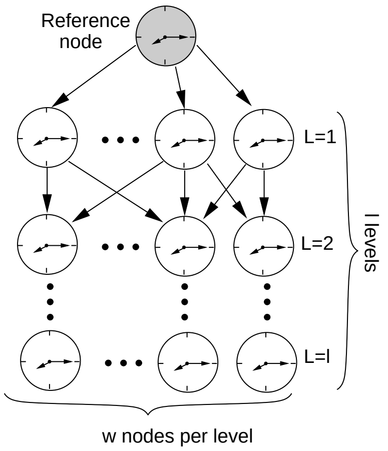


Figure 8.6: *Network structure used for the simulation study*

for the quality of the local synchronization, the Absolute Synchronization Error (ASE) for each node[11] has been considered. As consequence of the local error reduction, the length of the beacon intervals (BI) selected by each node converges to a common value, achieving a global synchronization. The standard deviation of the beacon intervals measured at a given time is an indicator of the quality of the synchronization achieved. The measurements were smoothed using a moving average computed over the last ten samples.

Simulation setup and methodology

Taking in account the typical routing structure generated by the bootstrap algorithm (see chapter 6), the author decided to consider a network with only unidirectional links where the time information propagates from the concentrator to the periphery.

The M nodes were organized in a grid consisting of l levels (the level number is the distance from the reference node) with w nodes each (Fig. 8.6). Each node at level i received the beacon generated by all the neighbors at level $i-1$. There was no communication path between nodes belonging to the same level.

An experiment consisted of at least fifty runs, each one with different seeds for the random number generators. The nodes drew the parameters for the given clock error model at the beginning of each run. A run spanned 10 days of simulated time.

Results

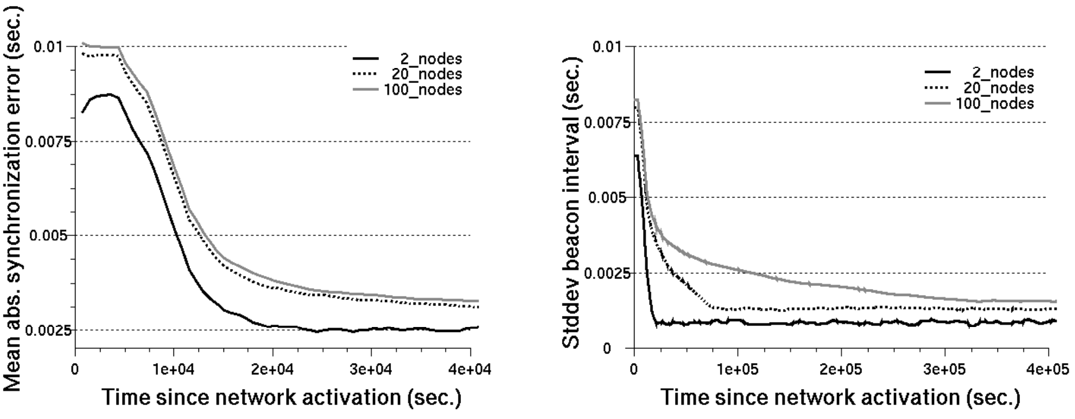
Preliminary results showed that the synchronization error increases for increasing l and decreasing w (assuming the total number of nodes M to be constant). Therefore, the simulations had been first based on the worst case where the network structure degenerates to a line ($l = M$, $w = 1$).

Behavior observed using the jitter clock error model The two graphs in Fig. 8.7 have been collected at the end of a simulation study consisting of 50 runs for each value of the network length $l = M$. The plotted lines indicate the mean of

---

[11] Absolute value of the difference between the expected and the real beacon arrival time.

the traces collected during all the runs sharing the same network configuration. The proposed approach is effective in reducing the effect of jitter, in particular the mean absolute synchronization error between neighboring nodes quickly reaches its minimum and saturates after few tens of beacon intervals (Fig. 8.7(a)) even for relatively large networks. While the convergence time (time required to reduce the error to 10% above the final value) of the ASE metric is almost independent of the network length l, the convergence time of the beacon interval estimation depends on l. For a network with 100 aligned nodes the convergence requires no more than 900 beacon intervals (Fig. 8.7(b)).

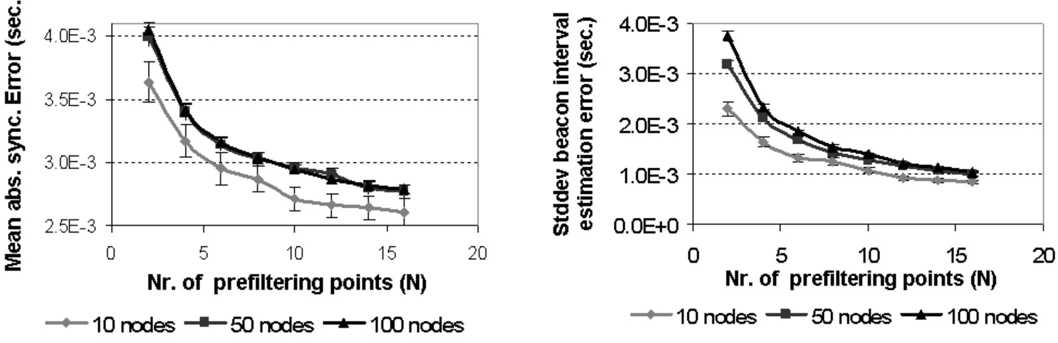


(a) ASE metric

(b) Standard deviation of the estimated beacon interval

Figure 8.7: *Behavior for jitter error model (beacon interval* 360s*)*

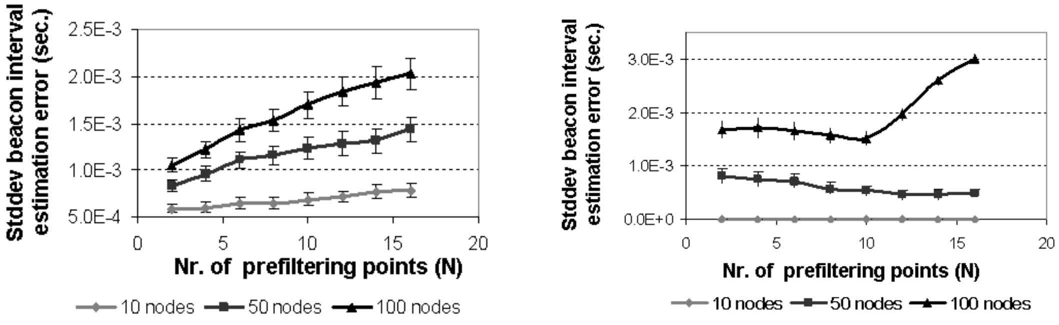
The impact of prefiltering As stated before, the collected data is smoothed using a moving average filter of length N. If using few prefiltering points, the quality of the synchronization increases significantly with N and then saturates (Fig. 8.8) if jitter modeled as a Gaussian process is assumed. However, if the measurement errors do not reflect a normal distribution an increase of N may cause a degradation of the estimation by reducing the systems ability to follow the changes of the model parameters (Fig. 8.9).



(a) ASE metric

(b) Standard deviation of the estimated beacon interval

Figure 8.8: *Influence of prefiltering in presence of jitter*



(a) Aging (b) Environmental changes

Figure 8.9: *Influence of prefiltering on the standard deviation of the estimated beacon interval, in presence of different clock error models*

If the clock error consists merely of frequency offset, the estimation error is almost independent from the length of the smoothing prefilter, at least for $N <= 20$.

It has to be noted that memory constraints may pose an additional limitation which has to be considered when choosing the length of the smoothing filter.

The estimation error increases with the maximum distance l from the reference node at a rate which seems to be lower than $\sqrt{l}$ for larger values of l (Fig. 8.10). Similar curves have been observed for the synchronization error between adjacent nodes.

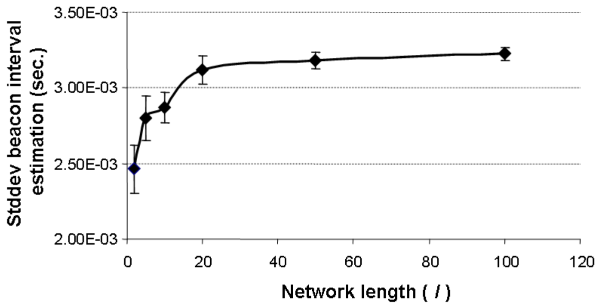
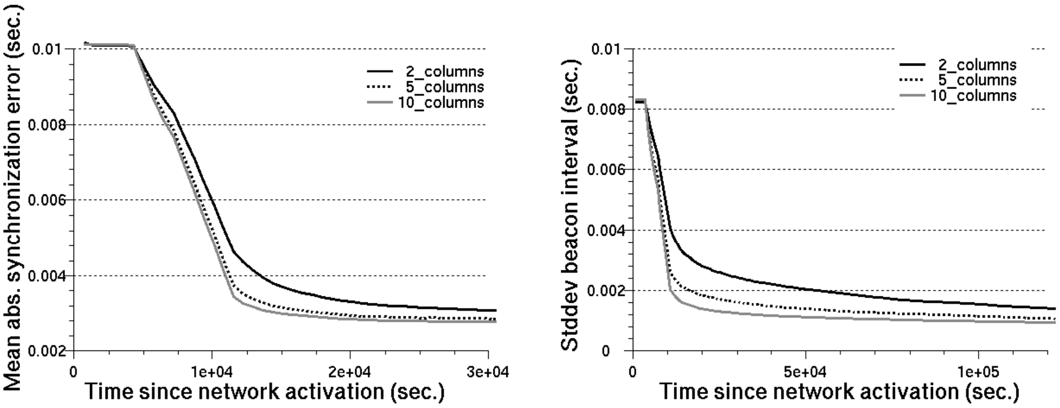


Figure 8.10: *Dependence of the standard deviation of the beacon interval estimation on the network length l*

Dependence on the number of nodes per level As stated before, a network structure consisting only of aligned nodes is a worst case which is not really representative for the typical network topology. The behavior for a more typical network structure has been analyzed locating the nodes in a grid structure consisting of 50 levels (rows, $l = 50$) and increasing the nodes per level w (columns, $w = 2, 5, 10$).

As shown in Fig. 8.11 a reduction of the convergence time along with a slight reduction of the synchronization error can be observed in such a structure. However, the basic behavior is the same as in the worst case network.

Improvement due to the synchronization algorithm Table 8.1 summarizes the results for a network consisting of hundred aligned nodes.



(a) ASE metric

(b) Standard deviation of the estimated beacon interval

Figure 8.11: *Behavior in presence of jitter for increasing number of nodes at each level (columns)*

Table 8.1: *Comparison of the observations done using different clock error models, in a network with 100 aligned nodes (N=8)*

	No Sync		Sync	
Clock error model	ASE-mean (sec.)	BI-stddev (sec.)	ASE-mean (sec.)	BI-stddev (sec.)
FreqOff	9.5e-3	8.2e-3	5.7e-5	8.5e-4
Aging	2.3e-2	2.0e-2	4.4e-4	1.5e-3
EnvChange	4.9e-3	4.2e-3	4.7e-5	1.6e-3
Jitter	1.0e-2	8.2e-3	3.0e-3	1.5e-3

The runs without synchronization algorithm have been done assuming that – upon reception of a beacon signal generated by a parent – a node simply updates the estimation of the arrival time of the next signal and leaves its own beacon interval unchanged. The length of the smoothing filter used by the synchronization algorithm has been set to N=8 points.

The synchronization algorithm causes a significant improvement of the observed metrics for all error models. As stated before the jitter error model represents a worst case for the synchronization algorithm. For the also critical EnvChange error model the synchronization algorithm is able to reduce the absolute synchronization error to 1% of the original one. However, the estimation of the beacon interval is only reduced to $\frac{1}{3}$. This is, however, not a shortcoming of the algorithm but is a consequence of the stochastic properties of the error process.

Comparison with the algorithm of Servetto and Hu As stated before the work of Servetto and Hu [73] has many similarities with the proposed approach. Its results are therefore considered as points of reference for benchmarking the proposed algorithm. The graph in in Fig. 8.12 has been obtained averaging the observations done in an experiment consisting of 50 runs in a network with 300 nodes, which have been organized in a mesh structure with 15 columns and 20 rows. The standard deviation of the jitter amplitude has been set to 10% of the beacon interval like in [73]. The results have been normalized to the duration of a beacon interval in order to permit a direct comparison with the results in the cited paper.

The observed error parameter is – after convergence – less than 10 percent of the value reported in the reference paper. This indicates that the proposed algorithm performs better than the other one. However, differences in the clock error model do not allow to exactly quantify the improvement.

Contrary to the algorithm proposed in [73], the proposed solution does also not require any additional traffic in order to make the synchronization stable.

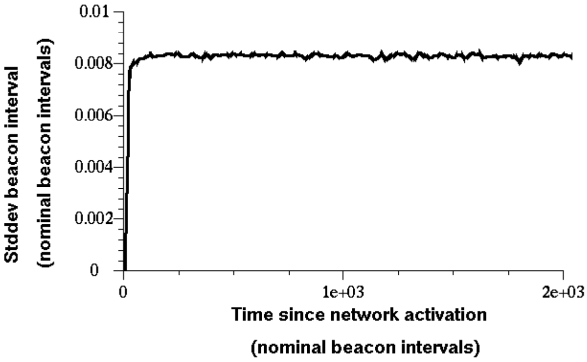


Figure 8.12: *Standard deviation of the estimated beacon interval – 20x15 nodes grid network,* $\sigma_{jitter} = 0.1T_{beacon}$.

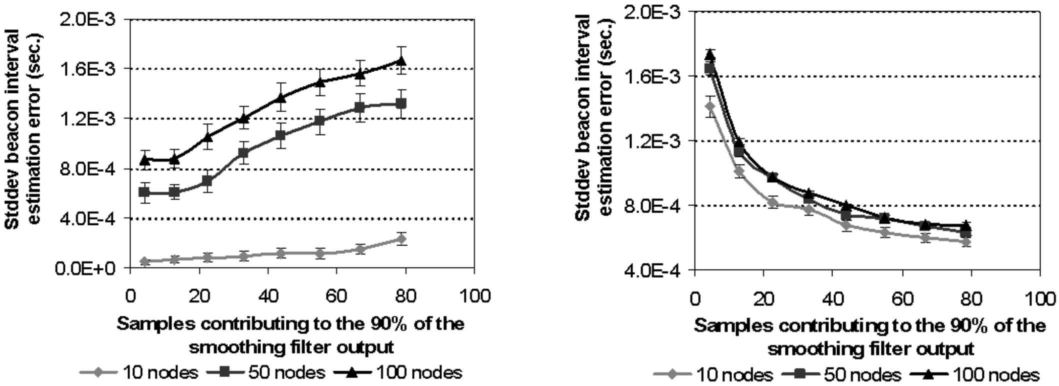
8.3.4 On the implementation on low-resource devices

A straightforward implementation of the proposed algorithm on a sensor node requires the availability of enough memory to store the data for the computation of the smoothing filter and of the coefficients of the AR(1) model. The number of data items which have to be stored is proportional to $N + P + 1$, were N is the number of points of the prefilter and P is the number of points considered for the computation of the autocorrelation values used for the computation of the coefficients of the AR(1) model.

If memory constraints do not allow a direct implementation of the synchronization algorithm it is possible to reduce the memory usage by substituting the moving average with the Exponential Moving Average (EMA). The EMA requires only the storage of the last value taken by the computed quantity.

Using this approach the smoothing filter takes the form $\bar{I}_k = \alpha \cdot I_k + (1 - \alpha) \cdot \bar{I}_{k-1}$ where $0 < \alpha < 1$ (similar equations may be written for all parameters which are computed by averaging some quantity). The length of the impulsive response of this filter is infinite. In order to permit a comparison with the previous results, the X-axis of the graphs in Fig. 8.13 reports the number of previous samples contributing to 90% of the filter output ($N_{90\%} = log_{(1-\alpha)}(0.9 \cdot \alpha)$).

The different behavior of the EMA influences the quality of the synchronization. Fig. 8.13 shows an example of the effects which may be observed. If the clock error is mainly due to jitter, the behavior of the synchronization algorithm (Fig. 8.13(b)) does not differ much from the previous observations (Fig. 8.8(b)). In presence of non-Gaussian errors, the EMS approach causes a noticeable degradation of the performance (Fig. 8.13(a)). The beacon interval estimation error also becomes sensitive to the parameters of the smoothing filter.



(a) FreqOff

(b) Jitter

Figure 8.13: *Influence of EMA prefiltering on the standard deviation of the estimated beacon interval*

An intuitive explanation is that now a larger number of previous samples contribute to the computation of a new value and, therefore, the system adapts itself slower to the modifications of the error process.

8.3.5 On the reinitialization of the KF

As stated before (section 8.3.2), the continuous beacon interval adaptation is responsible for the modification of the characteristics of the error process. Therefore the KF might become unable to follow the evolution of the beacon generation process. In this case a reinitialization of the KF becomes necessary.

The evolution of the Mean Absolute Synchronization Error (MASE) has been considered to define a trigger event for the filter reinitialization. The MASE is computed by using an exponential moving average filter. It has to be noted that, as long as the filter is able to follow the evolution of the observed quantity, the MASE is a non increasing function of the time, while, when the filter loses the ability of following the evolution of the observed quantity, the MASE starts increasing. Therefore, the system takes traces of the minimum values taken by the MASE since the last filter reset and resets the filter each time actual value exceeds the minimum by at least a given tolerance (20% seems to be a good choice). This happens only as long as the error is bigger than a given target value which has been set to two times the measurement accuracy in this case.

The number of filter resets depends on the error model and its parameters. In case of a simple frequency offset the updates are concentrated at the beginning of the network operation. The values in Table 8.2 have been observed in a node at a distance of 20 hops from the reference node[12].

---

[12] This particular node has been chosen, because, during the test, it has been characterized by the

Table 8.2: *Evolution of the KF reinitialization rate in absence of jitter, observed on the 20th node in a line*

Interval (beacons)	[0-30)	[30-60)	[60-120)	[120-240)
Frequency of the KF reinitialization (1/Beacons)	0.124	0.081	0.043	0.002

Sudden variations of the error characteristics trigger the reinitialization of the KF filter. In case of jitter the estimator is reset at an almost constant rate during the whole network operation time. The reset rate is an increasing function of the error standard deviation and of the error rate.

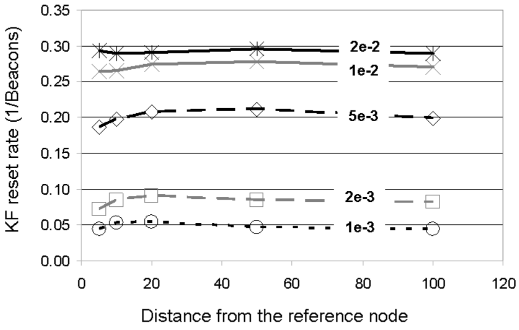


Figure 8.14: *Average KF reinitialization rate in presence of jitter plotted in dependence of the distance from the reference node and for different values of the jitter standard deviation*

The graph plotted in Fig. 8.14 shows the average KF reset rate observed in different nodes for different values of the error standard deviation.

This value depends on the distance from the reference node and has a maximum when the nodes are at about 20 hops from the reference. The same behavior can be observed in a network with more than one node per layer. The reason for this phenomenon is not clear and more accurate investigations would be required to allow a correct interpretation of this data.

It is worth noting that in the case of the *EnvChange* error model the average rate of KF reinitialization reflect the observations done with the *FreqOff* model.

8.3.6 How to generate a stable time base by mutual synchronization of the clock references

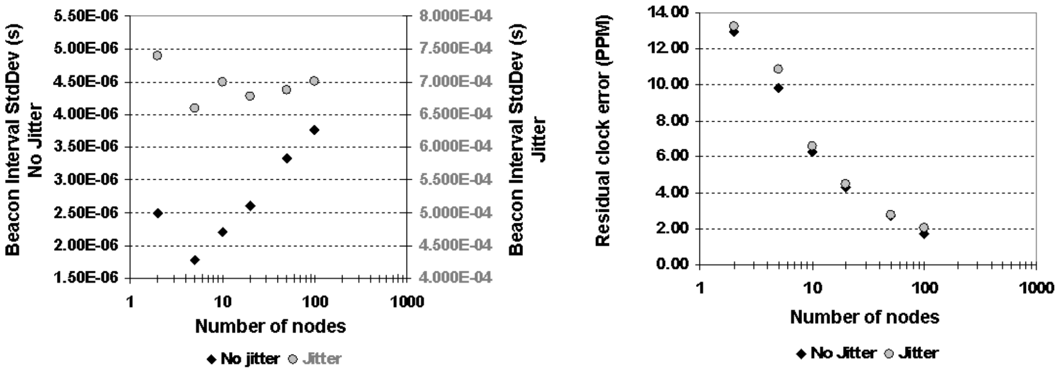
The analyzed synchronization mechanism provides clock synchronization by propagating the time information from the reference node to the periphery.

It is interesting to analyze the behavior of a group of reference nodes which are involved in a continuous mutual synchronization process. In this case, the respective

---

highest rate of filter reinitializations.

beacon interval estimations converge to the mean of the beacon intervals of the unsynchronized nodes.



(a) Standard deviation of the synchronization error (b) Mean absolute clock error

Figure 8.15: *Mutual synchronization*

The standard deviation of the beacon estimation error (Fig. 8.15(a)) is, after the convergence, in presence of jitter about a tenth of the value observed in the open loop and about the half in absence of jitter, while the mean absolute clock error decreases rapidely with the number of nodes (Fig. 8.15(b)).

The previous observations suggest that a mutual synchronization of the reference nodes might be a viable solution to obtain a virtual time base which is more stable than each single reference clock.

8.3.7 Further considerations

The proposed local synchronization mechanism is suitable for a general class of WSNs using a beacon enabled MAC and unidirectional communication links. After convergence, all nodes are sending the beacon signals at a constant beacon rate. This also suggests a viable way to generate a global time base in a WSN.

The proposed algorithm relies on the a priori knowledge of the MAC layer behavior and, in particular, it makes an opportunistic usage of the beacon frames which are necessary for the functioning of the MAC layer.

The synchronization problem has been reformulated as an estimation problem, which has been solved by using a Kalman Filter (which is optimal in presence of AWGN). In order to allow the operation of the Kalman Filter on data affected by errors which are not AWGN, a smoothing prefilter has been introduced.

The proposed method, which relies only on local computation and which uses the a priori knowledge about the implemented MAC layer, is effective in reducing the impact of different clock errors and behaves better than the similar solution proposed in the literature [73].

The algorithm may also be implemented in devices with severe RAM limitations by modifying the smoothing filter.

Chapter 9

An analysis of the network behavior with all mechanisms activated

The previous analysis steps have been mainly focused on the functionalities of the single algorithm in the worst case. All the simulation parameters, like the building structure, the propagation parameters and the activation strategy, had been chosen with the purpose of stressing the analyzed component. These unfavorable combinations of events will hopefully happen only seldom in a real network. It is also natural to ask which is the overall behavior of a network under more realistic boundary conditions.

The following section 9.1 describes the idealized model of a residential building which is the basis of the analyses carried out in this chapter. The rest of this chapter shows the behavior of the network and in particular the sensitivity to the variation of the different parameters.

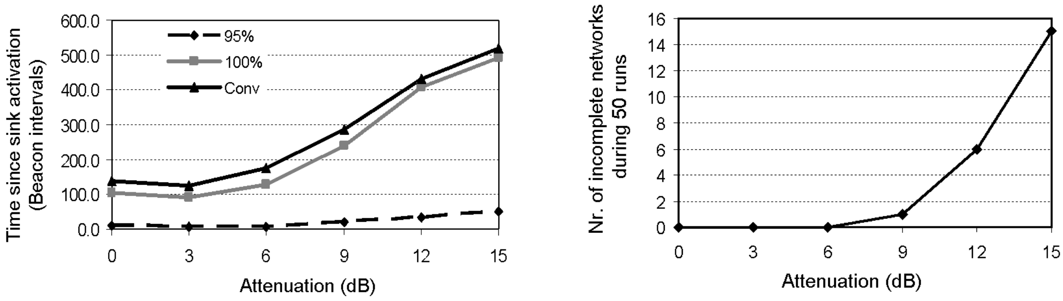
9.1 A 3-D node placement

The following analyses have been done by using a node placement which seems to better reproduce an hypothetical real setup.

The considered playground resembles the structure of a residential building having the form of a cuboid consisting of five floors, each one having twelve rooms grouped in three equal sized adjacent rows. The propagation parameters are set according to the measurement reported in section 5.3.2. In particular, the attenuation between adjacent floors has been set to 12 dB while the attenuation between adjacent rooms of the same floor has been set to 9 dB. The mean attenuation between nodes in the same room is drawn during the initialization phase according to a uniform distribution in the interval $[0\ dB, 10\ dB]$. The stochastic portion of the attenuation is Gaussian distributed with standard deviation $\sigma = 6dB$ (for the whole propagation model see section 5.3). The fixed part of the attenuation L_f has been set to 35 dB while 10 dBm is the level of the signal generated by the nodes.

At the beginning of the simulation, the number of nodes in each room is drawn uniformly between 1 and 5. The sink is the only elements in its room which is located in a corner of the building.

The activation strategy tries to approximate the real case, where many installers



(a) Integration and convergence time

(b) Number of incomplete networks

Figure 9.1: *Dependence on the attenuation between adjacent rooms*

are active at the same time in different parts of the building. The nodes are activated in a random order during an interval of 100 beacons, and the sink is activated last just after 100 beacon periods from the simulation start.

At the activation time, each node collects the information about the already active nodes by observing the channel for a whole beacon period and making use of the algorithm described in chapter 6.

The parameters of the neighbor selection algorithm have been set according to the values suggested in the previous chapter.

From the activation on each node tries to synchronize with the neighbor nodes and dynamically adjusts the guard period by setting its guard period to four times the average absolute synchronization error. By doing so the nodes are able to achieve a better utilization of the communication resources.

The failure recovery mechanism described in chapter 7 enables the reintegration of the nodes which lose the synchronization with the neighbors and provides continuous adaptation of the network structure.

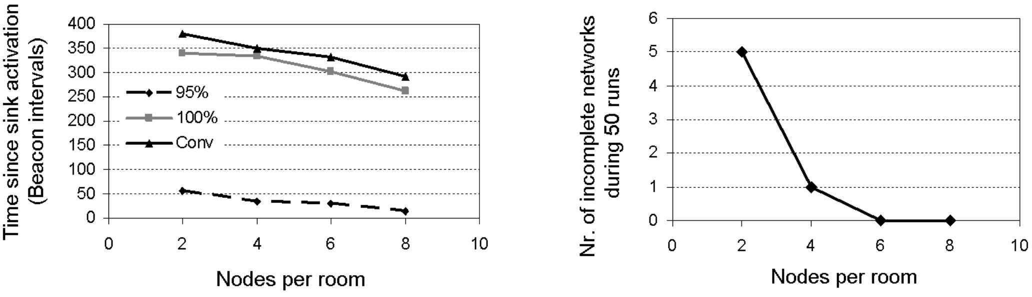
9.2 Sensitivity to the variation of the propagation conditions

During the previous tests the attenuation values between the rooms have been kept constant in all runs, in such a way it has been possible to observe the dependence of the algorithm performance on the parameter selection.

Clearly the propagation conditions influence the ability of obtaining a connected network. With respect to the attenuation between adjacent rooms it is easy to figure out the results in the two extreme cases of no attenuation and unbounded attenuation. The former case will end almost certainly with a connected network, while in the latter case it is impossible to obtain a network which spans more than a single room.

A short test aimed a showing the transition between the two cases has been done, by keeping the algorithm parameters constant and varying the attenuation between neighboring rooms.

The results in Fig. 9.1(b) show the existence of a transition for the integration above which the probability of obtaining a fully connected network decreases rapidly.



(a) Integration and convergence time (b) Number of incomplete networks

Figure 9.2: *Dependence on the number of nodes in each room*

The curves in Fig. 9.1(a) represent the dependence of integration and convergence time on the attenuation. It is interesting to note that the shortest integration and convergence times do not correspond to the absence of attenuation. It is even worth noting that even though the majority of the nodes are integrate quickly into the routing structure, the network completion requires a much longer time interval. This result, which appears at a first glance in contrast with the results in chapter 6, depends on additional constraints concerning the minimal distance between active periods, which do not permit to select some combination of neighbors.

Moreover, the deletion of stale neighborhood relationships, which has been introduced in chapter 7 removes also some intermittent links which are characterized by a marginal signal quality. The combination of the two effects delays the network integration (and convergence) by the time required to discover the better neighbors. This worsening of the convergence speed is, however, compensated by the improved reliability of the links belonging to the routing structure after the convergence.

9.3 Sensitivity to the variation of the node density

The tests carried out in chapter 7 highlighted the dependence of the failure recovery behavior, and in particular of the network setup time, on the node density. Therefore, it is not surprising that a similar behavior can be observed in a network running the whole protocol suite, as a test carried out with an increasing number of nodes highlighted (Fig. 9.2).

The results have been obtained by putting in each room, except the one containing the sink, the same number of nodes.

The results plotted in 9.2(a) show, as expected, an improvement of the integration time for increasing node density. In particular, it has to be remembered that because of the mechanism introduced in chapter 7 some nodes rely for the completion of the network setup on the active search process. Consequently the setup time is influenced by the node density.

Moreover, a low node density may result in the impossibility of carrying out the network initialization (see Fig.9.2(b)), e.g., because the impossibility of determining a set of neighbor nodes which satisfy the constraints on the minimal distance between active periods.

9.4 On the guard period selection

As stated before the main goal of the time synchronization is to achieve a local synchronization which permits the reduction of the guard period and hence a better utilization of the scarce communication resources.

The simplest solution consisting of the adoption of a fixed guard period is clearly suboptimal. In fact, with respect to the results in Fig. 8.11(a), it appears clear that the interval length, which is optimal at the beginning of the network activities, results in the waste of network capacity (energy) during the steady state operation.

Much more interesting is the possibility of adapting the length of the guard period to the actual synchronization error, which is measured by the Absolute Synchronization Error (ASE)[1]averaged among all uplink neighbors.

The algorithm which determines the evolution of the guard period has to respond to the competing demands of providing a prompt reaction in case of synchronization loss, observing a "conservative" behavior and permitting an effective reduction of the guard period.

Many different empiric solutions have been tested. The best results have been observed with the following algorithm 2.

Algorithm 2: Adaptation of the guard period

$maxGP$ ($minGP$)	is the maximum (minimum) length of the guard Period.
S_{ASE}	is a scale factor for the ASE
GP_i	is the guard period at the time i
ASE_i	is the average ASE at the time i

```
// Start with the largest guard period
$GP_0 = maxGP$
// Update the length of the guard period
foreach $i$ do
    if At least a beacon has been received then
        // Reduce gently the guard period
        $GP_i = MIN\,(maxGP\,,\,MAX\,(minGp\,,\,0.6 \cdot GP_{i-1} + 0.4 \cdot S_{ASE} \cdot ASE_i))$
    else
        // Respond quickly to a synchronization error
        $GP_i = maxGP$
end
```

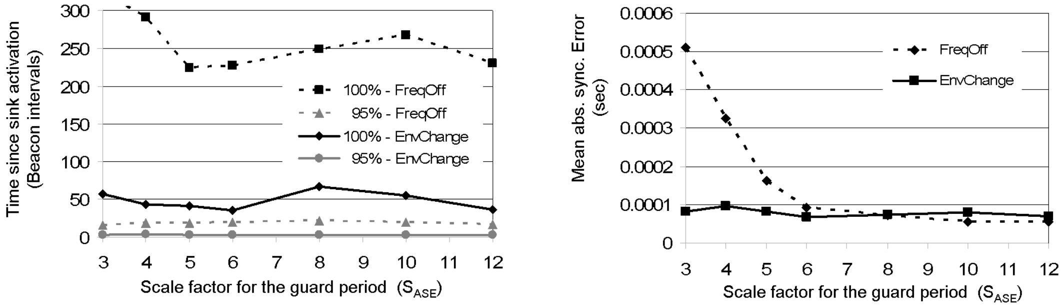
The length of the guard period converges to $MAX\,(minGp\,,\,S_{ASE} \cdot ASE_i)$ where the last term is the asymptotic value taken by the ASE (if any). The initial condition is set to the half of the activity period, which maximizes the probability of observing the beacon signals the neighbors generate.

The adoption of the Exponential Moving Average (EMA) permits to follow the variations of the ASE and at the same time to smooth the series of the guard period values.

The reduction of the guard period influences the behavior of the time synchronization algorithm by modifying the distribution of the beacon arrival measurements by

---

[1] The ASE has been selected instead of other functions since it is easier to compute on weak processors.



(a) Integration time

(b) Mean Absolute Synchronization Error at the end of the simulation

Figure 9.3: *Sensitivity to the variation scale factor S_{ASE}*

preventing the reception of those beacon signals which advance the expected arrival time by more the actual guard period. As consequence, the estimator reacts quicker if the local beacon estimation is shorter than the beacon interval of the neighbors and slower in the opposite case. Moreover, the beacon loss might cause the removal of the corresponding node from the neighbor list and at the same time an underestimation of the ASE. The content of the *"else"* clause is aimed at permitting the observation of the lost node during the next observation period and so at permitting a partial compensation of the additional error.

Interestingly these effects, which are particularly noticeable in presence only a simple frequency offset (See. 8.3.3 - *FreqOff*), appear mitigated if the clock error varies during the time like in the clock error model *EnvChange.*

An increment of the scale factor (S_{ASE}) mitigates this effect by forcing the adoption of wider guard period and hence by permitting the accommodation of bigger estimation errors (Fig. 9.3(a)). The optimum value seems to be about 6, in correspondence to a (probably local) minimum for the network integration time.

As stated before, the adaptation of the guard period influences the behavior of the time synchronization algorithm. The curves in Fig. 9.3(b) show the dependence of the mean absolute synchronization error on the scale factor S_{ASE} obtained with two different clock error models. The results observed with the *EnvChange* clock error model are somehow surprising. The absolute synchronization error is invariably smaller than in the case of the simple frequency offset.

A possible explanation lies in the observation that because of the presence of an unstable clock the reduction of the guard period happens slower than in the case of simple clock drift.

Chapter 10

Conclusions

The need for replacing a preexisting hardware system for indoor metering applications offered the opportunity to study the feasibility of a Wireless Sensor Network (WSN) based approach in order to implement an indoor Wireless Metering Network (WMN). In particular, it allowed analyzing the problems related to the strict energy constraints and the application in an indoor environment.

Preliminary analysis highlighted the inappropriateness of the available solutions. Therefore, new energy efficient solutions have been investigated for the most critical aspects of the network operation.

In order to keep the complexity low and hence to make the problem tractable, the single problems (network bootstrap, self-healing and time synchronization) have been treated separately. The analyses have been mainly focused on the worst case behavior of the single solutions. At the end, the behavior of the complete system has been analyzed under milder assumptions which better reproduce a hypothetical real setup. These last results confirmed the validity of the developed solution.

10.1 The propagation model

The unavailability of an adequate number of hardware modules to tests the different solutions imposed the adoption of an alternative, simulation based solution for the validation of the proposed algorithms and network protocols.

An new propagation model has been developed with the explicit purpose of reproducing the peculiarities of the indoor radio signal propagation. The validity of the proposed approach has been checked by comparing the simulation results with measurements done in a real building. It has to be noted that the proposed model is characterized by a limited computational complexity which makes it suitable for the simulation of large networks, i.e. up to 1000 nodes.

A simulation framework implementing this propagation model has been developed as extension of preexisting work done at the TU-Berlin. It is characterized by a modular approach which allows an incremental approach to the development of the protocols suite and, at the same time, a simple path to the test of alternative solutions.

10.2 Network bootstrap

The bootstrap process has been addressed first. It has been developed by considering only a subset of the existing boundary constraints[1] in order to obtain a generic solution.

The resulting greedy heuristic algorithm starts with the knowledge of a limited number of already active neighbor nodes and produces a tree like routing structure rooted at the concentrator. It can be used in all cases in which a low power sensor network uses a beacon enabled MAC layer.

Deliberately, no assumption has been made about the strategy exploited to provide the initial information to the nodes. According to the mainstream ideas of the WSN research, during the bootstrap phase each node might operate according to a higher duty cycle. This enables fully autonomous operations at the cost of higher energy expenses.

In the considered case, since the node installation is a supervised process, some external device (e.g. a handheld computer) may be used which communicates a list of the already active nodes to each new element at the power-on time. In such a way it would be possible to obtain a further reduction of the energy expense during the bootstrap.

10.3 Self-healing

The long expected operation network life time together with the indoor deployment make the implementation of self-healing and network maintenance / optimization mechanisms mandatory.

Instead of developing an autonomous set of mechanisms to cope with these issues, the author preferred to integrate the required functionalities into the previously developed bootstrap algorithm which then will be active the whole network life. This approach allows, on one side, to keep the overall code complexity low and, on the other side, a seamless transition between the different phases of the network life. It has to be noted that the boundary between the different life phases are fuzzy and the clear identification of the transition points between them is a difficult task.

10.4 Time synchronization

The time synchronization is a typical problem of all distributed systems. Every solution is based on a continuously running mechanism which needs constant measurements of the clock difference in order to compensate the non-ideal behavior of the local clock system. Because of these mechanisms, the generation of dedicated data traffic might be source of non negligible energy expenses.

The formulation of the original synchronization problem as an estimation problem and the exploitation of the a priori knowledge about the MAC protocol behavior allowed to obtain a particularly robust and energy efficient synchronization strategy. The developed algorithm fulfills the main objective of reducing the synchronization error between directly communicating nodes and at the same time provides a common time

---

[1] Only the low duty cycle, the memory and computational constraints and the used Medium Access Control (MAC) protocol.

base to the whole network structure. Because of its simplicity the synchronization algorithm can be implemented on any system characterized by the periodical emission of beacon (or data) packets.

10.5 Final consideration

The original question was the feasibility of an indoor WMN able of long lasting unattended operation even in presence of strict energy and HW constraints.

This work provides the description of a set of enabling mechanisms for the considered network. Some additional protocols and algorithms are needed to transform the studied network into a commercial product. In particular, a transport protocol featuring the data aggregation, some security mechanisms and a firmware update protocol have to be added to the proposed protocol set. However, since the first field trial of the new network nodes started soon after the end of this project, the author is confident that the first commercial products which implement the described solutions will be deployed soon.

Appendix A

The Kalman Filter

With his famous publication in the year 1960 [76], Rudolf Kalman proposed a new approach to the state estimation problem. He based the construction of his recursive filter on the properties of conditional Gaussian random variables. The proposed algorithm was based on the idea of minimizing the state vector covariance norm by adding, at each iteration, a correction term proportional to the prediction error.

A.1 The equations of the Kalman Filter (KF)

The following considerations apply to the discrete time system described in Fig.A.1 where $x(t)$ is the state vector we intend to estimate, $A(t)$ is the known square state transition matrix, $y(t)$ is the measured vector, $H(t)$ is the rectangular measurement matrix and $b(t)$ is a given control signal. $w(t)$ and $v(t)$ are two zero mean white Gaussian noise processes which are mutually independent and independent from $x(t)$ and $y(t)$. They represent the process noise and the measurement noise, respectively. The former has the same dimension as $x(t)$ while the latter has the same size as $y(t)$. $r^w(t)$ and $r^v(t)$ are the respective covariance matrices.

The system evolution is described by the following equation:

$$\begin{cases} x(t+1) = A(t)x(t) \; + \; b(t) \; + \; w(t) \\ \quad\; y(t) = H(t)x(t) \; + \; v(t) \end{cases} \tag{A.1}$$

We want to compute the state estimate $\hat{x}(t)$ by adding a correction component, which takes in account the last observation $y(t)$, to the *a priori* state estimate $\hat{x}^-(t)$ which has been computed based on the previous observations (A.2).

$$\hat{x}(t) = \hat{x}^-(t) + K(t)\left(y(t) - H(t)\hat{x}^-(t)\right) \tag{A.2}$$

The matrix $K(t)$ is a *gain*, which has been chosen in order to minimize the estimation error. The factor $\left(y(t) - H(t)\hat{x}^-(t)\right)$ is called *innovation* or *residual*. The innovation represents the discrepancy between the a priori model and the observation. In order to make the equation more readable, we indicate from now on the time dependency in the subscript and we omit it when it is not relevant.

In order to compute the optimal gain, we define the *a priori* estimate error $e_t^- \equiv x_t - \hat{x}_t^-$ and the *a posteriori* estimate error $e_t \equiv x_t - \hat{x}_t$. The covariance of the

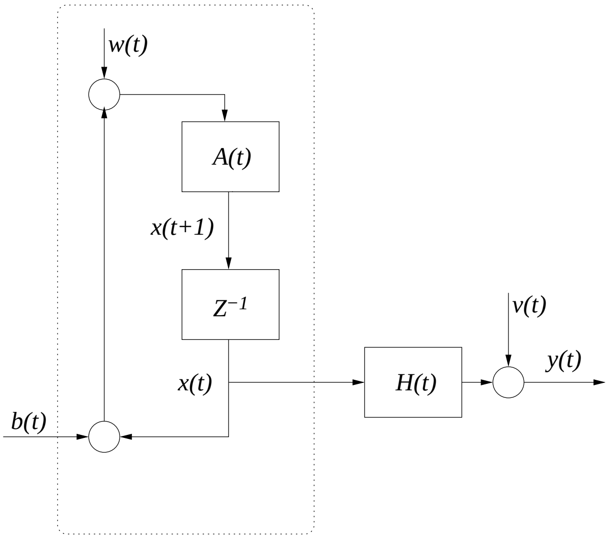


Figure A.1: *The system model used for the derivation of the KF equations*

two estimate errors are

$$P_t^- = E\left[e_t^- e_t^{-T}\right] \tag{A.3}$$

and

$$P_t = E\left[e_t e_t^T\right] \tag{A.4}$$

Afterwards, the optimal gain is computed by minimizing the *a posteriori* estimate error.

One of the possible expression of K_t is given by

$$K_t = P_t^- H^T \left(H P_t^- H^T + r_t^v\right)^{-1} \tag{A.5}$$

When the covariance of the measurement decreases, the system weighs the residual more heavily. On the other hand a reduction of the a priori estimate error causes a decrease of the importance of the residual.

The computation of the KF can be split in two stages which follow one other:

Prediction (time update) An a priori estimation is computed on the basis of the past knowledge (A.6) along with a projection of the error covariance (A.7)

$$\begin{aligned} \hat{x}_t^- &= A\,\hat{x}_{t-1} + b_t & \text{(A.6)} \\ P_t^- &= A\,P_{t-1}\,A^T + r_{t-1}^w & \text{(A.7)} \end{aligned}$$

Correction (measurement update) The new measurements are incorporated into the model in order to improve the estimate. This step consists of the computation of the filter gain (A.8), the correction of the estimate (A.9) and the computation of the estimation error covariance (A.10)

$$\begin{aligned} K_t &= P_t^- H^T \left(H P_t^- H^T + r_t^v\right)^{-1} & \text{(A.8)} \\ \hat{x}_t &= \hat{x}_t^- + K_t \left(y_t - H \hat{x}_t^-\right) & \text{(A.9)} \\ P_t &= \left(I - K_t H\right) P_t^- & \text{(A.10)} \end{aligned}$$

Because of its recursive structure the KF is more flexible than other estimators (e.g. the Wiener filter) and its simple structure makes it interesting for the implementation on computing devices.

The whole derivation of the filter equations can be found in the work of Kalman [76] or in each book on optimal control. An alternative derivation of the filter based on a deterministic approach, which emphasizes the error reduction aspect, can be found in [77].

A.2 The KF equations in presence of colored noise

The previous equations are valid when the measurement and input noise processes are white. It is natural to ask if these can be extended in order to deal with colored noise processes. The optimal handling of this new problem is generally possible but at the expense of an increased complexity of the filter. The way to address the problem can be found in each book on optimal filtering (like [78]). It consists in modeling the two colored processes $w(t)$ and $v(t)$ with two finite-dimensional systems driven by two independent white noise processes. This is possible if w and v are independent and stationary. The main problem consists now in having to work with a system much larger than the original one.

The particular case of output noise $v(t)$ having Markov properties needs mentioning, because it can be treated without increasing the dimension of the optimal filter. The derivation of the filter follows the path described in [78].

Starting point are the equations describing the system evolution for $t \geq 0$:

$$\begin{cases} x(t+1) = A(t)x(t) \; + \; b(t) \; + \; w(t) \\ \quad\;\; y(t) = H(t)x(t) \; + \; v(t) \\ v(t+1) = N(t) \; + \; \eta(t) \end{cases} \tag{A.11}$$

In equation A.11 $\eta(t)$ and $w(t)$ are independent Gaussian white processes. The initial conditions $x(0)$ and $v(0)$ are mutually independent and independent of $\eta(t)$ and $w(t)$. Moreover, $x(0)$ and $v(0)$ are Gaussian vectors with means $m_{x(0)}$ and 0 and covariance matrices $P(0)$ and $R(0)$, respectively.

It is possible to consider the quantity

$$\begin{aligned} \bar{y}(t+1) &= y(t+1) - N(t)y(t) \\ &= H(t+1)x(t+1) \; - \; N(t)H(t)x(t) \; + \; v(t+1) \; - \; N(t)v(t) \\ &= [H(t+1)A(t) \; - \; N(t)H(t)]\, x(t) \; + \; \eta(t) \; + \; H(t+1)w(t) \\ &= \bar{H}(t)x(t) \; + \; \eta(t) \; + \; \bar{G}(t)w(t) \end{aligned} \tag{A.12}$$

defined for each $t \geq 0$. It is then possible to set

$$\bar{y}(0) \; = \; H(0)x(0) + v(0) \; = \; y(0) \tag{A.13}$$

At this point the KF theory permits to carry out the estimation of x based on the observations of $\bar{y}$.

It is interesting to note that the size of the new model does not differ from the size of the original system.

Bibliography

[1] G. Pottie and W. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, pp. 51–58, 2000.

[2] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless micro-sensor network models," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 6, no. 2, pp. 28–36, 2002.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, 2002. [Online]. Available: citeseer.ist.psu.edu/akyildiz02survey.html

[4] H. Karl and A. Willig, *Protocols and architectures for Wireless Sensor Networks*, Wiley, Ed., 2005.

[5] C.-Y. Chong and S. P. Kumar, "Sensor networks: Evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, August 2003.

[6] A. Ha, *Wireless Sensor Network Designs*, Wiley, Ed., 2003.

[7] C. Sharp, S. Schaffert, A. Woo, N. Sastry, C. Karlof, S. Sastry, and D. Culler, "Design and implementation of a sensor network system for vehicle tracking and autonomous interception," in *2nd European Workshop on Wireless Sensor Networks (EWSN)*, 2005, pp. 93–107. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1462002

[8] K. Römer and F. Mattern, "The design space of wireless sensor networks," *Wireless Communications, IEEE [see also IEEE Personal Communications]*, vol. 11, no. 6, pp. 54–61, 2004. [Online]. Available: http://dx.doi.org/10.1109/MWC.2004.1368897

[9] J. A. Paradiso, "Energy scavenging for mobile and wireless electronics," *IEEE pervasive computing*, vol. Jan 2005, pp. 17–27, 2005.

[10] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," in *Proceedings of the ACM Mobicom*, 2000.

[11] C. E. Perkins, *Ad Hoc Networking*, Addison-Wesley, Ed. Pearson Education Corporate, 2001.

[12] E. H. Callaway, *Wireless Sensor Networks Architectures and Protocols*, Auerbach, Ed., 2004.

[13] J. M. Rabaey, M. J. Ammer, J. L. da Silva, D. Patel, and S. Roundy, “Picoradio supports ad hoc ultra-low power wireless networking,” *Computer*, vol. 33, no. 7, pp. 42–48, 2000.

[14] C. Lynch and F. O. Reilly, “Processor choice for wireless sensor networks,” in *Workshop on Real-World Wireless Sensor Networks*, 2005.

[15] “Datasheet of the mica2 wireless measurement system,” Crossbow Technologies, inc., 2007. [Online]. Available: http://www.xbow.com

[16] “Datasheet of the transceiver chipcon cc1100,” 2007. [Online]. Available: http://focus.ti.com/docs/prod/folders/print/cc1100.html

[17] S. Khan, K. Iniewsky, Y. Wang, and W. Myint, “Emerging short reach wireless technologies: From 802.11n to 60+ ghz mm-wave radios - a silicon perspective,” in *IWCMC’06*, 2006.

[18] P. Santi, *Topology Control in Wireless Ad Hoc and Sensor Networks*, Wiley, Ed., 2005.

[19] M. Sugano, T. Kawazoe, Y. Ohta, and M. Murata, “Indoor localization system using rssi measurement of wireless sensor network based on zigbee standard,” in *The IASTED International Conference on Wireless Sensor Networks*, 2006.

[20] S. Roundy, D. Steingart, L. Frechette, P. Wright, and J. Rabaey, “Power sources for wireless sensor networks,” *Lecture Notes in Computer Science*, vol. Volume 2920/2004, pp. 1–17, 2004.

[21] J. Blumenthal, M. Handy, F. Golatowski, M. Haase, and D. Timmermann, “Wireless sensor networks - new challenges in software engineering,” *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA’03. IEEE Conference*, vol. 1, 2003. [Online]. Available: http://citeseer.ist.psu.edu/718537.html

[22] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, “System architecture directions for networked sensors,” in *Architectural Support for Programming Languages and Operating Systems*, 2000, pp. 93–104. [Online]. Available: citeseer.ist.psu.edu/382595.html

[23] A. Dunkels, B. Grnvall, and T. Voigt, “Contiki - a lightweight and flexible operating system for tiny networked sensors,” in *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Tampa, Florida, USA, Nov. 2004. [Online]. Available: http://www.sics.se/ adam/dunkels04contiki.pdf

[24] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, “The nesc language: A holistic approach to networked embedded systems,” in *ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2003. [Online]. Available: citeseer.ist.psu.edu/gay03nesc.html

[25] R. Kumar, S. P. Chaudhuriz, and U. Ramachandran, “System support for cross-layering in sensor network stack,” in *International Conference on Mobile Ad Hoc and Sensor Networks (MSN)*, Dec. 2006.

[26] A. Kpke, V. Handziski, J.-H. Hauer, and H. Karl, “Structuring the information flow in component-based protocol implementations for wireless sensor nodes,” Technical University Berlin, Telecommunication Group, Tech. Rep., 2004.

[27] “Zigbee specification,” January 2008.

[28] “IEEE standard for information technology, telecommunications and information exchange between systems, local and metropolitan area networks - specific requirements part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs),” January 2006.

[29] F. Cuomo, S. D. Luna, and T. Melodia, “Routing in zigbee: benefits from exploiting the ieee 802.15.4 association tree,” in *ICC ’07. IEEE International Conference on Communications*, 2007.

[30] N. Kushanagar, G. Montenegro, and C. Schumacher, “Ipv6 over low-power personal area networks (6lowpans): Overview, assumptions, problem statements and goals,” August 2007, RFC 4919.

[31] N. Kushanagar, G. Montenegro, J. Hui, and D. Culler, “Transmission of ipv6 packets over ieee 802.15.4 networks,” September 2007, RFC 4944.

[32] N. Abramson, “Development of alohanet,” *IEEE Transaction on Information Theory*, vol. 31(2), pp. 119–123, 1985.

[33] R. Jurdak, C. V. Lopes, and P. Baldi, “A survey, classification and comparative analysis of medium access control protocols for ad hoc networks,” *IEEE Communications Surveys & Tutorials*, vol. 6, 2004.

[34] L. Gu and J. A. Stankovic, “Radio-triggered wake-up capability for sensor networks,” in *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS04)*, 2004, pp. 1080–1812.

[35] A. Koubaa, M. Alves, and E. Tovar, “A comprehensive simulation study of slotted csma/ca for ieee 802.15.4 wireless sensor networks,” in *Proc. IEEE International Workshop on Factory Communication Systems (WFCS 06), Torino, Italy*, June 2006, p. 183192.

[36] J. Ha, W. H. Kwon, J. J. Kim, and Y. H. Shin, “Feasibility analysis and implementation of the IEEE 802.15.4 multi-hop beacon enabled network,” in *15th Joint Conference on communications and Information (JCCI)*, 2005.

[37] R. Wattenhofer, “Algorithm for wireless sensor networks,” 2006, eWSN 2006 Tutorial.

[38] J. M. Kahn, R. H. Katz, and K. S. J. Pister, “Mobile networking for smart dust,” in *ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99),Seattle, WA*, 1999.

[39] B. N. Clark, C. J. Colbourn, and D. S. Johnson, “Unit disk graphs,” *Discrete Mathematics*, vol. 86, p. 165.177, 1990.

[40] E. Miluzzo, X. Zheng, K. Fodor, and A. T. Campbell, “Radio characterization of 802.15.4 and its impact on the design of mobile sensor networks,” in *5th European Conference on Wireless Sensor Networks, EWSN*, R. Verdone, Ed., January / February 2008.

[41] F. Kuhn, R. Wattenhofer, and A. Zollinger, “Ad-hoc networks beyond unit disk graphs,” in *Joint workshop on Foundations of mobile computing, DIALM-POMC*, 2003.

[42] F. Xue and P. R. Kumar, “The number of neighbors needed for connectivity of wireless networks,” *Wireless Networks*, vol. 10, pp. 169–181, 2004.

[43] A.-S. Hu and S. D. Servetto, “Asymptotically optimal time synchronization in dense sensor networks,” in *WSNA ’03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications.* New York, NY, USA: ACM Press, 2003, pp. 1–10.

[44] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*, C. U. Press, Ed., 2005.

[45] D. Lymberopoulos, Q. Lindsey, and A. Savvides, “An empirical characterization of radio signal strength variability in 3-d ieee 802.15.4 networks using monopole antennas.” in *EWSN*, 2006, pp. 326–341.

[46] M. Günes, M. Wenig, and A. Zimmermann, “Improving manet simulation results - deploying realistic mobility and radio wave propagation models,” in *ISCC*, 2007, pp. 39–44.

[47] A. L. Cavilla, G. Baron, T. E. Hart, L. Litty, and E. de Lara, “Simplified simulation models for indoor manet evaluation are not robust,” in *IEEE SECON*, 2004.

[48] A. Varga, “The omnet++ discrete event simulation system,” in *European Simulation Multiconference (ESM’2001)*, 2001.

[49] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, M. Press, Ed., 1989.

[50] V. A. K. W. Drytkiewicz, S.Sroka and H.Karl, “A mobility framework for omnet++,” in *3rd International OMNeT++ Workshop, at Budapest University of Technology and Economics, Department of Telecommunications Budapest, Hungary,*, 2003.

[51] D. Son, B. Krishnamachari, and J. Heidemann, “Experimental study of concurrent transmission in wireless sensor networks,” in *SenSys ’06: Proceedings of the 4th international conference on Embedded networked sensor systems.* New York, NY, USA: ACM, 2006, pp. 237–250.

[52] A. Kochut, A. Vasan, A. U. Shankar, and A. Agrawala, “Sniffing out the correct physical layer capture model in 802.11b,” in *ICNP ’04: Proceedings of the Network Protocols, 12th IEEE International Conference.* Washington, DC, USA: IEEE Computer Society, 2004, pp. 252–261.

[53] N. Altan and M. Gronauer, "Verfahren zur installation eines hierarchischen netzwerkes," EP1638262 (10.01.2008) and DE102006024336A1 (06.12.2007).

[54] L. Clare, G. Pottie, and J. Agre, "Self-organizing distributed sensor networks," in *SPIE Conf. Unattended Ground Sensor Technologies and Applications*, vol. 3713, 1999, pp. 229–237.

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

[56] S. Chakrabarti and E. Nordmark, "Lowpan neighbor discovery extensions," August 2007, internet-Draft.

[57] R. W. T.Moscibroda, P. von Rickenbach, "Analyzing the energy-latency trade-off during the deployment of sensor networks," in *INFOCOM '06*, 2006.

[58] F. Kammer and H. Tubig, "Graph connectivity," Institut fr Informatik, TU Mnchen, Tech. Rep. TUM-I0422, Dec 2004.

[59] S. S. Kunniyur and S. S. Venkatesh, "Network devolution and the growth of sensory lacunae in sensor networks," in *ISIT 2004 (International Symposium on Information Theory, 2004)*, 2004.

[60] N. Altan and E. Rathgeb, "Opportunistic clock synchronization in a beacon enabled wireless sensor network," in *7th International Conference on AD-HOC Networks & Wireless, Sophia Antipolis, France*, 2008, accepted Paper.

[61] X. R. Wang, J. T. Lizier, O. Obst, M. Prokopenko, and P. Wang, "Spatiotemporal anomaly detection in gas monitoring sensor networks." in *EWSN*, ser. Lecture Notes in Computer Science, R. Verdone, Ed., vol. 4913. Springer, 2008, pp. 90–105.

[62] J. R. Vig, "Introduction to quartz frequency standards, revision 1," *NASA STI/Recon Technical Report N*, vol. 93, pp. 15 330–+, Oct. 1992.

[63] D. L. Mills, "Internet time synchronization: the network time protocol." *IEEE Trans. Communications*, vol. COM-39, pp. 1482–1493, 1991.

[64] ——, "Network time protocol (version 3) specification, implementation and analysis," March 1992, RFC 1305.

[65] B. D. Deeths and G. Brunette, "Using ntp to control and synchronize system clocks - part i: Introduction to ntp," Sun microsystems inc., Tech. Rep., July 2001, sun BluePrints OnLine.

[66] D. L. Mills, A. Thyagarajan, and B. Huffman, "Internet timekeeping around the globe," in *Precision Time and Time Interval (PTTI) Applications and Planning Meeting*, December 1997, pp. 365–371.

[67] J. Elson and D. Estrin, "Time synchronization for wireless sensor networks," in *IPDPS '01: Proceedings of the 15th International Parallel & Distributed Processing Symposium.* Washington, DC, USA: IEEE Computer Society, 2001, p. 186.

[68] J. van Greunen and J. Rabaey, “Lightweight time synchronization for sensor networks,” in *WSNA ’03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications.* New York, NY, USA: ACM, 2003, pp. 11–19.

[69] S. Ganeriwal, R. Kumar, and M. B. Srivastava, “Timing-sync protocol for sensor networks,” in *SenSys ’03: Proceedings of the 1st international conference on Embedded networked sensor systems.* New York, NY, USA: ACM Press, 2003, pp. 138–149.

[70] R. Karp, J. Elson, D. Estrin, and S. Shenker, “Optimal and global time synchronization in sensornets,” Tech. Rep., 2003. [Online]. Available: citeseer.ist.psu.edu/article/karp03optimal.html

[71] J. Elson and K. Römer, “Wireless sensor networks: A new regime for time synchronization,” UCLA, Tech. Rep., July 2002. [Online]. Available: citeseer.ist.psu.edu/elson02wireless.html

[72] J. Elson, L. Girod, and D. Estrin, “Fine-grained network time synchronization using reference broadcasts,” *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, 2002.

[73] A.-S. Hu and S. D. Servetto, “Algorithmic aspects of the time synchronization problem in large-scale sensor networks,” *ACM/Kluwer Journal on Mobile Networks and Applications*, 2004. [Online]. Available: citeseer.ist.psu.edu/hu03algorithmic.html

[74] K. S. Kim and B. G. Lee, “Kalp: A kalman filter-based adaptive clock method with low-pass prefiltering for packet networks use,” *IEEE Transaction on communication*, vol. 48, pp. 1217–1225, 2000. [Online]. Available: citeseer.ist.psu.edu/kim00kalp.html

[75] J. Vig, “Introduction to quarz frequency standard,” in *International Frequency Control Symposium and Exposition*, 2006. [Online]. Available: http://www.ieee-uffc.org/freqcontrol/quartz/vig/vigtoc.htm

[76] R. Kalman, “A new approach to linear filtering and prediction problems,” *Transaction of the ASME-Journal of basic Engineering*, vol. 82 (series D), pp. 35–45, 1960.

[77] J. L. Roux, “An introduction to kalman filter: probabilistic and deterministic approach,” University of Nice, Tech. Rep., 2003.

[78] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, T. Kaliatt, Ed. Prentice-Hall Inc., 1979.