Sensors on speaking terms

Schedule-based medium access control protocols for wireless sensor networks

Lodewijk F.W. van Hoesel

Composition of the Graduation Committee:

prof.dr.ir	C.H.	Slump	Universiteit Twente (promotor)
dr.	P.J.M.	Havinga	Universiteit Twente (assistant promotor)
prof.dr.ir	G.J.M.	Smit	Universiteit Twente
prof.dr.ir	W.C.	van Etten	Universiteit Twente
dr.	K.G.	Langendoen	Technische Universiteit Delft
prof.dr.ir	L.	van der Perre	IMEC
prof.dr.ir	A.B.	Smolders	Technische Universiteit Eindhoven/NXP

This research was conducted within the EU project EYES (IST-2001-34734) and the EU project Embedded WiSeNts (FP6-004400).

Printed by Wöhrmann Print Service
ISBN 978-90-365-2497-1

SENSORS ON SPEAKING TERMS

SCHEDULE-BASED MEDIUM ACCESS CONTROL PROTOCOLS FOR WIRELESS SENSOR NETWORKS

DISSERTATION

to obtain
the doctor's degree at the University of Twente,
on the authority of the rector magnificus,
prof.dr. W.H.M. Zijm,
on the account of the decision of the graduation commitee,
to be publicly defended
on Thursday, June 21st, 2007 at 15:00

by

Lodewijk Frans Willem van Hoesel

born on March 11th, 1978,
in Eindhoven, The Netherlands

This dissertation is approved by:

prof.dr.ir	C.H.	Slump	Promotor
dr.	P.J.M.	Havinga	Assistant promotor

Abstract

Wireless sensor networks make the previously unobservable, observable. The basic idea behind these networks is straightforward: all wires are cut in traditional sensing systems and the sensors are equipped with batteries and radio's to virtually restore the cut wires. The resulting sensors can be placed closely to the phenomenon that needs monitoring, without imposing high wiring costs and careful engineering of the position of the devices. Sensors can even be attached to mobile objects. As a result, monitoring can be done more efficiently and at higher sensing resolution compared to traditional sensor systems. Yet, these are not the only advantages of wireless sensor networks.

We —as user of the wireless sensor network– are not interested in constant streams of sensor readings, but we are more interested in the interpretation of the sensor readings. This is exactly the big potential of wireless sensor networks. Due to intelligence, added locally to the wireless sensors, and collaboration between the individual sensors, the network by itself can carry out complex tasks related to observing. To achieve these goals, wireless sensors must be "on speaking terms" i.e. the nodes must be able to exchange information.

The aim of this thesis is to provide a set of communication rules —commonly known as *medium access control* (MAC) protocol– that organizes efficient communication through a shared wireless medium and is well suited for the inherent characteristics of wireless sensor networks. The MAC protocol determines when a wireless sensor transmits its sensor information and is thereby in control of one of the most energy consuming components in the wireless sensor hardware architecture. The lifetime of the battery operated wireless sensors is thus heavily dependant on the efficiency of the MAC protocol.

This thesis provides a self-organizing, schedule-based medium access approach. In general, this class of medium access is recognized for its energy-efficiency and robustness against high peak loads. These aspects are required by wireless sensor networks. Its drawbacks are message delay, over-provisioning and required time synchronization between wireless sensors.

In our approach, wireless sensors analyse local medium usage and autonomously choose when to access the medium i.e. transmit. In the analysis, medium usage of second order neighbours is also taken into account. Therefore, our approach does not suffer from the well-known hidden terminal problem, and additionally, the wireless medium is spatially reused without energy-wasting conflicts. The medium access schedule of wireless sensors is adjusted when —e.g. due to mobility of nodes– conflicting schedules exist. The correctness of the general medium access control protocol is verified with a formal analysis tool, the model checker Uppaal.

Based upon the general schedule-based medium access approach, EMACs and LMAC are designed. Both protocols have been implemented on prototype wireless sensors designed in this thesis. The prototype hardware platforms are used to demonstrate that time synchronization is sufficiently attainable between wireless sensors.

The EMACs protocol includes clustering techniques to determine the role of wireless sensors concerning wireless communication. To maintain a connected communication structure, potentially not all sensors are required to actively participate in multi-hop communication. Therefore, two roles are introduced. Active sensors create a connected backbone, which is used by passive sensors. Passive sensors can conserve energy and the lifetime of the wireless sensor network is extended. With the introduction of these roles, we actively target one of the drawbacks of schedule-based access: over-provisioning.

The LMAC protocol is a simplified version of the EMACs protocol. In wireless sensor networks, sensor readings are most often forwarded to so-called gateway nodes. These nodes present the findings of the network to an exterior user. The LMAC protocol includes the necessary functionality to transport messages to gateways without requiring additional routing protocols. Using the LMAC protocol, we demonstrate that message delays can be significantly reduced. Our methods are applicable on our general schedule-based medium access approach, thereby addressing one of the drawbacks of schedule-based medium access: latency.

The LMAC protocol can be easily compromised when attackers have full knowledge of the protocol. Encrypting messages, applying source authentication and authenticating message content, makes the task of undermining a wireless sensor network more difficult, yet security keys are easily compromised. Wireless sensor networks are even more easily attacked with jamming. We present such (energy-efficient) jamming attacks for our schedule-based medium access approach and in particular LMAC. For this class of attacks little knowledge is required about the MAC protocol, which makes it a realistic threat. We show its effectiveness on LMAC and present countermeasures. Additionally, consequences for other schedule-based MAC protocols are discussed.

Acknowledgement

Working towards a dissertation is like making a jigsaw puzzle. Some pieces just fit at the first try, while others look alike and need careful matching. I am very grateful to the many people I could rely on to get even the last jigsaw piece fitted and I would like to seize this opportunity to thank them.

First and foremost I would like to thank Kees Slump and Thijs Krol, who gave me the opportunity to begin the puzzle. Kees gave me the freedom to conduct research at a befriended research group, although I am quite sure he kept a close eye on my progress. Thijs welcomed me into his CADTES group and is the embodiment of the pleasant atmosphere in the research group.

I would also like to thank Paul Havinga. He supervised me on a day-to-day basis. His enthusiasm for wireless sensor networks and smart surroundings is inspiring. We enjoyed many fruitful discussions, during the famous EYES meetings on Tuesday's and during several trips to exotic countries. The discussion I remember best is the one in the jungle of Australia where we (almost) stepped on a (probably poisonous) snake, or was it the one just before the fishy speeding ticket? Thanks for the instructive and pleasant years!

Stefan Dulman, Tim Nieberg, Wu Jian and Supriyo Chatterjea have been great roommates and colleagues. Although everyone from a different angle, together we pursued to get wireless sensor networks a few steps closer to reality. It was a pleasant work ambience in the office and I enjoyed our activities outside the office!

I am much obliged to Yee Wei Law and Pieter Hartel for putting medium access security aspects on the research agenda. Not the "standard" Alice, Bob and Eve kind of secrecy protocols etc, but jamming. During the jamming experiments, I tried my best Chinese —probably all words I know are curse words– on Law, but was not half as successful as he in Dutch.

It was pleasant to collaborate with Angelika Mader and Ansgar Fehnker. If I am not mistaken, it is due to me that Angelika's computer kind of crashed during the evaluation of topology (29). Topology (29) is apparently not the easiest one, but it is comforting to know that Angelika's monstrous fast replacement computer verified the topology in a wink –I am quite sure that she is happy with this little "incident".

Pierre Jansen is always ready for a chat, either on important research topics or just ordinary chitchat. We discussed many times wireless sensor networks and medium access control. Additionally, I enjoyed our occasional relaxing "Limburg-style" cycling trips, although I am still not convinced that a GPS route planner is necessary bicycling equipment.

My best supporters are the ones I do not see every day and certainly not at work:

my family. I am my parents, René and Louise van Hoesel very thankful for their constant support and encouragement. The one I unfortunately see least is my sister Anneke. She is currently pursuing her PhD in the U.S.A. Nevertheless, she is always ready for a cheerful optimistic chat and I enjoy the visits she makes. I would like to thank Huub and Thea van Dieren for their compassionate support.

Mildred van Dieren, thank you for your loving and unconditional compassion. You make this work worthwhile.

Contents

Chapter 1

Introduction

Ongoing miniaturization and optimization of low-power electronics, especially of microprocessors, memory and transceivers have contributed greatly to the growth of interest for wireless applications. Interesting challenges can be found in wireless applications where this low-power technology is massively utilized: wireless sensor networks. The vision of *a veiled network of cheap, tiny embedded devices, collecting sensorial data and communicating wirelessly to collaborate in reaching a contextual description of their environment* [22, 52] —a so called wireless sensor network– has become realizable due to these recent advances in technology.

The deeply embedded devices (i.e. the wireless sensor nodes) can be placed closely to the phenomenon that needs monitoring, without imposing high wiring costs and careful engineering of the position of the devices [4]. Their small size and low cost allow for dense and large-scale deployment. Consequently, monitoring can be done more efficiently and at higher sensing resolution compared to traditional sensor systems. *The previously unobservable becomes observable.*

Yet, the power of wireless sensor networks lies not only in observing. Abstracting simple sensor readings to interrelated events, allows for the guarding of complex processes. Due to intelligence and collaboration between the individual nodes, the network by itself can carry out complex tasks related to observing.

For example, Madden [68] presents an architecture for organizing the wireless sensor network as a distributed database, which is able to answer complex queries. Another application illustrating the in-network intelligence capabilities of wireless sensor networks, is the proactive assistance for assemble-it-yourself furniture [8].

Functionality along the same line is applicable in warehouse monitoring, where wireless sensors check temperatures of fresh food and at the same time guard the complex logistics process in the warehouse, making sure every truck receives the correct goods [23].

Wireless sensor networks provide insight into complex processes by close observation and collaboration between the individual building blocks of the network. Ultimately, wireless sensor networks help us complete our daily tasks or even make our lives easier and safer. However, before any of the above functionalities can be realized, the wireless sensors must be on speaking terms i.e. the nodes must be able to exchange information.

The aim of this thesis is to provide a set of communication rules —commonly known as *medium access control* (MAC) protocol– that organizes efficient communication through a shared medium and is well suited for the inherent characteristics of wireless sensor networks.

The outline of this chapter is as follows. First, we describe the context of wireless sensor networks. Secondly, we describe general objectives of the data link layer. In this layer, which is defined by Zimmermanns OSI model [115], the set of rules for communication resides. The influence of the MAC protocol on the performance of the wireless sensor network is discussed. Next, we present the contributions of this thesis and conclude this chapter with an outline of this thesis.

1.1 Context of wireless sensor networks

Wireless sensor nodes have two primary functions: (1) sensing and (2) communicating. The sensing function is used to obtain a contextual description of the physical environment covered by the network. For this purpose, the nodes are equipped with one or more sensors. The communication function is used to transport sensor information to other sensors and to the user(s) of the wireless sensor network.

We have stated that the goal of this thesis is to provide a set of communication rules, which is well suited for the inherent characteristics of wireless sensor networks (WSNs). What are these characteristics? Römer et al. [92] argue that it is becoming increasingly difficult to describe typical characteristics —both regarding hardware and software– of wireless sensor networks (WSNs) due to the wide variety of applications. Consequently, the design space of WSNs is very broad. In this thesis, we limit ourselves to the following (according to the design space defined in [92]):

- **Communication modality** — Nodes communicate with each other by sending and receiving small (packetised) messages. There are several ways imaginable in which wireless sensors can communicate with each other, e.g. light, (ultrasonic) sound and radio. The most common communication modality in WSN literature is by means of radio. In this work, we assume such communication modality (Appendix A).

  Our medium sharing methods focus themselves on the characteristics of low-cost and low-power radios. Consequently, the communication range between nodes is limited and might be prone to errors. In addition, transmissions are naturally not directed to one receiver in particular, but can be received by all nodes within radio coverage of the transmitter, i.e. neighbour nodes. The wireless channel is a truly shared channel. Nodes cannot transmit and receive at the same time, i.e. the communication modality is assumed to be half duplex [95]. Additionally, nodes cannot transmit or receive multiple packets in parallel.

- **Network topology** — While network topology can take many forms (e.g. single hop, star, networked star, tree and arbitrary graph [92]), our focus is on multi-hop networks in which nodes can potentially communicate with all their neighbour nodes.

  In this thesis, we assume that nodes collaborate in delivering sensor information and that multi-hop forwarding is applied when necessary (Figure 1.1). In other words, the area covered by the wireless sensor network is expected to be larger than the radio coverage of single nodes. This allows for *spatial reuse* of the wireless medium, i.e. multiple nodes can transmit simultaneously if not interfering. Our medium sharing methods intend for spatial reuse of the wireless medium.

- **Patternless and iterative deployment** — We presume that nodes are deployed in their physical environment without careful engineering of their position. Additionally, nodes may be added or removed after initial deployment. Hence, we expect that node density may vary over time and that the network topology is dynamic. This assumption requires the medium sharing protocols to be scalable. Further, dense and large-scale node deployment is most likely.

- **Mobility** — The majority of the WSN work (e.g. environmental monitoring or military applications) assumes that nodes are fixed and do not change their position after initial deployment. Interesting applications, e.g. the warehouse monitoring application [23], arise when (part of the) nodes are attached to mobile objects.

  We assume mobility and a dynamic network topology. From a MAC protocol perspective, the set of neighbouring nodes, to which nodes can communicate, is not fixed; nodes may travel into or move out of communication range. However, we assume that node speeds are relatively slow compared to transmission ranges and message durations. We do not evaluate the performance of MAC protocols in mobile settings, but we show that our protocols adapt to changing neighbourhoods and dynamic topologies.

- **Data sink nodes** — We presume that there is a heterogeneous interest in sensor readings. For this purpose, special devices will be placed in the WSN to collect sensor readings to present them to the outside world. From a data link layer perspective, we assume that nodes are identical, although the nodes themselves can be of different types i.e. with different amounts of memory, computational power or different sensors.

- **Ad hoc networking** — WSNs are in general classified as ad hoc networks, i.e. nodes directly communicate with each other without needing a supporting infrastructure. Consequently, communication links can have a finite life span and communication is required to be self-organizing.

In Chapter 2, we present two wireless sensor network application domains which correspond to the above discussed context: (1) environmental monitoring, and (2) transportation and logistics. The application overview in Chapter 2 is by far not complete, however, the sketched applications provide sufficient view on application requirements regarding the MAC protocol.

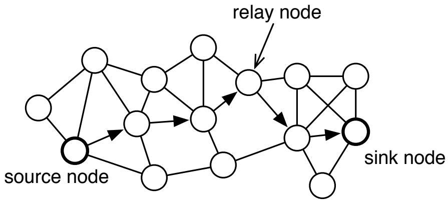


Figure 1.1 — Nodes (represented as circles) can only communicate with nodes in radio coverage (represented as lines). The source is unable to directly reach the sink node. Intermediate nodes assist in forwarding messages

After describing the vision and the design space of wireless sensor networks, now let us look at the challenges.

Cheap and tiny sensor nodes allow for fine-grained observation. As a result, sensors might be deployed densely and in large numbers, as we concluded above. The large scale of wireless sensor networks makes it virtually impossible to replace or refresh power sources on the individual nodes. It is most likely that the nodes are placed unobtrusively in the environment (i.e. built into walls etc), complicating the task of energy replenishment even more. Often, terms like *ubiquitous computing*, *pervasive computing* and *ambient intelligence* are used in relation to wireless sensor networks, all indicating the expected unobtrusiveness of such networks.

On the other hand, the user of the network expects the network to be cost-effective and thus to have a lifetime of many years, depending on the application type. Therefore, *energy* is assumed to be a very scarce resource and its limited availability is one of the most challenging problems in WSNs.

In general, wireless sensors have severely limited resources to carry out sensing and communication tasks [5]. The reason for this is again cost-effectiveness. In our work, we assume that the devices constructing the wireless sensor network are deeply embedded and resource constrained. This is in contrast to the sometimes in literature described wireless sensors build of PC's, laptops or PDA's. In general, the latter devices have plenty of memory and processing power available, yet their high cost per device results in an expensive wireless sensor network. Obviously, this makes dense and large-scale deployment unattractive. It is crucial to make efficient wireless sensor systems, both hardware and software.

In the next section, we describe the objective of the data link layer and —in particular– of the medium access control protocol.

1.2 Objectives of the data link layer

The objective of the so called *data link layer* is to provide communication services between interconnected terminals [115]. In other words, the data link layer organizes reliable communication between nodes within radio coverage. An important part of the data link layer is the medium access control (MAC) protocol. This protocol defines when nodes are allowed to transmit packets and when they are allowed to receive.

The wireless channel is a shared medium. This means that any transmitted (packetised) message can be received by any node within radio coverage of the transmitter and the reception of packets can also be disturbed by interference or concurrent transmissions. In other words, a node can easily disturb a transmission of another node when transmitting simultaneously, as we show in Appendix C.

The *hidden terminal problem* [55] is a well known example in which two or more nodes access the medium simultaneously and cause collision at receiving node(s). In general, conflicts in medium access cause transmissions to get lost and have a negative influence on the performance of the data link layer. Conflict-free organization of the *access* of the medium (i.e. when to transmit) is a key objective of the data link layer.

Additionally, transmissions can get lost due to external causes, e.g. due to reflections in the wireless channel. For this reason, reliability issues are often included in the data link layer e.g. nodes respond to received packets by transmitting dedicated acknowledgement packets. These packets confirm the successful arrival of data, and, when an acknowledgement fails to arrive, the data link layer typically decides to resend the data. Note that for any acknowledgement scheme, mutual communication is required between nodes.

In this work, we pursue these two objectives: (1) efficient, conflict-free communication and (2) reliable (multi-hop) communication. The data link layer itself is —strictly speaking– not involved in the multi-hop forwarding of messages. The aim of this thesis is therefore not to study routing protocols or networking layers, although these have an influence on efficiency of communication as well. In Section 2.6, we discuss data link layer requirements in light of WSNs.

The performance of a data link layer is captured in the following measures:

- **Message delay** — Message delay or latency is the time it takes from the generation of information until its delivery in its final destination. A node might need to wait until its turn to access the medium. This time is source of latency. It is often preferred that latency is as short as possible, however, we argue that in most WSN applications some latency is tolerated (Chapter 2).

- **Delivery ratio** — The delivery ratio measures how many of the total generated packets are delivered to their final destination. Packets can —for example– be dropped when message buffers in the nodes are full or when the channel capacity is insufficient to transport all generated data.

- **Channel utilization** — The channel utilization gives an indication of how much of the channels capacity can be used to transfer information. In fact, this measures the overhead of the data link layer.

- **Fairness** — The MAC protocol virtually shares the total channel capacity between nodes. Fairness measures the (average) share of channel capacity each nodes gets. Preferably, each node gets a fair share of the capacity. In WSNs, this requirement is often relaxed, since nodes collaborate to establish a context description of their physical environment. Selfish usage of the wireless medium is therefore not likely to occur.

Next, we discuss the relationship between the data link layer and the most prominent challenge of wireless sensor networks: the constricted energy reserves. The MAC protocol directly controls the communication modality of the wireless sensor node and is in charge of regulating one of the most energy consuming components in the sensor node architecture: the radio (Appendix B).

1.3 Relationship between MAC protocol and node lifetime

We argued that wireless sensors lack resources, and especially energy to carry out their tasks. How can energy wastage, for benefit of the node's lifetime, be minimized by the data link layer?

In the recent state of technology, the RF transceiver consumes most energy in the sensor node architecture (Appendix B). Cheap, low-power transceivers, suitable for this architecture, consume typically in the order of 30 mW in receiving or transmitting state, and in standby state less than 30μW. Thus, a reasonable battery of 1 Ah allows for 100 hours of continuous transmitting or receiving (excluding any other component of the wireless sensor node) and roughly 11 years of being in standby mode. During the lifetime of the battery, a sensor node is able to transmit or receive roughly 2145 (raw) Mbytes. Every hour in standby mode reduces this by 21.5 kbytes, but increases the lifetime of the node by 59.6 minutes. It is clear that a node should switch its transceiver to the standby state whenever possible, in order to prolong its lifetime.

Because communication is one of the most energy consuming operations in wireless sensor networks, communication optimization is an interesting research topic. After all, this would significantly increase the lifetime of nodes, or would allow for downsizing power sources, which would reduce hardware costs of the wireless sensors. The energy problem and communication challenges congregate in the data link layer, making it a relevant research topic.

When looking at the above (simplified) energy consumption model, we conclude that the sleep state is the preferred state regarding node lifetime (Figure 1.2(1)). In this state, the energy consumption is roughly a factor $1,000$ below the other states, meaning that —neglecting energy consumption of other components– the lifetime of a node would increase by a factor $1,000$. The standby state is clearly the preferred state concerning lifetime of the node's power source.

However, nodes are required to collaborate in establishing a representative view of their physical environment and consequently communication is required. Nodes must selflessly be available for forwarding data from other nodes. In Figure 1.2(1), nodes clearly do not interact and message delays are optimally worse.

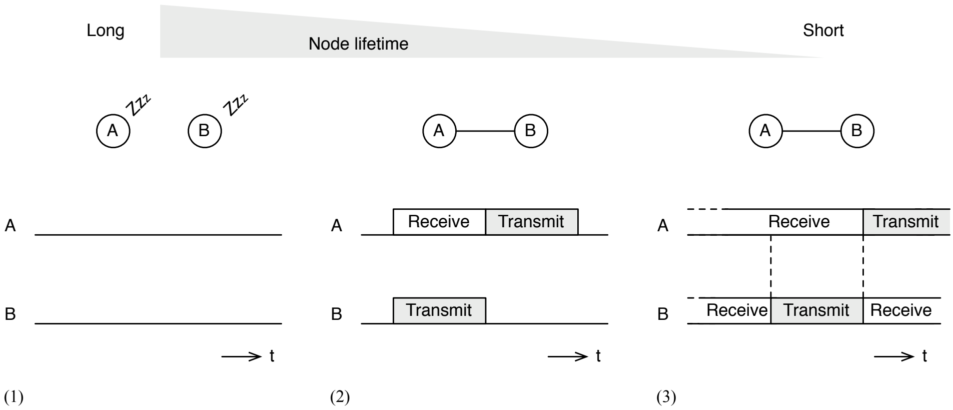


Figure 1.2 — How is the node's lifetime affected by the data link layer? (1) A transceiver always in sleep state results in longest node lifetime, but no communication takes place. Highly reactive nodes (3) result in short lifetimes. Practical MAC protocols trade-off energy consumption with reactiveness (2)

In Figure 1.2(3), we see the exact opposite: nodes are always in receive state to accept incoming packets, even when no packets are transmitted. As a result, the network is very reactive and message delays are short. From an energy consumption perspective, the situation in Figure 1.2(3) would drain the node's power source quickest –an invidious situation.

The data link layer should apply some sort of *sleeping pattern* in which it is still able to serve its neighbouring nodes in relaying data, and is able to conserve energy by putting the transceiver into sleep state. Clearly, to communicate efficiently, a trade-off should be made between energy conservation and readiness for communication.

A sleeping pattern by itself does not completely solve the energy-wastage problems at the data link layer. To get a useful sleeping pattern, nodes must synchronize their communication activities and maintain synchronization. The better the synchronization, the more energy can be conserved at the data link layer and the more energy-efficient the MAC protocol will be, as we argue in Chapter 3.

There is in general a trade-off between performance measures of MAC protocols [96]; to achieve low latency, nodes are required to be able to receive messages often, process them and forward them to other nodes. This prevents long periods of inactivity in which energy is conserved. Obviously, data throughput is limited by long sleep intervals.

Above, we sketched this trade-off with a very simple energy consumption model. Efficient use of the transceiver has a large impact on battery lifetime, but has little impact on the volume of data that can be transmitted or received. Therefore, we require efficient networking protocols, which are especially tailored for the class of devices in WSNs.

Not all trade-offs can be solved by the MAC protocol itself; they are given by the requirements of the complete wireless sensor system. At data link level, the *overhead* and energy-waste —due to, for example, *idle listening*– must be reduced to a minimum and parameters —like length of sleep interval and duty cycle– must be tune-able to meet application requirements.

1.4 Contributions

In this thesis, we look at efficient data link layers for wireless sensor networks.

Important for designing a data link layer is that it should be well-suited for typical wireless sensor networks. We establish a clear view on what WSN applications and routing protocols require from a communication perspective.

We provide taxonomy of medium sharing mechanisms and discuss their value for use in wireless sensor networks. The mechanism of choice is *schedule-based* medium access, because it has good prospect in being energy-efficient.

We present a novel schedule-based medium access approach and analyse its performance from a theoretical perspective. Additionally, we verify the proposed algorithm by both simulation and model checking tools. Our contribution includes self-organization and self-healing of schedule-based medium access, making this type of medium access protocols suitable for dynamic, mobile environments and making it scalable.

We discuss two concrete MAC protocols based upon our method: EMACs and LMAC. EMACs characterizes itself by its cross-layer optimization capabilities and backbone creation based on clustering techniques. We also introduce the notion of active and passive nodes. Active nodes are part of the autonomously created backbone, while passive nodes save energy by not participating in maintaining communication structures.

The LMAC protocol is a simplified version of EMACs. The protocol is approached from a practical perspective. We use results of [59] to show that it performs excellently on energy-efficiency and delivery ratio. We show that latency can be reduced and that the protocol can be made adaptive to the expected amount of data that has to be transported.

We present an energy-efficient methodology on how our medium scheduling approach —in particular LMAC– can be attacked. We discuss consequences for other WSN MAC protocols. We propose countermeasures against the attacks.

1.5 Thesis overview

The outline of this thesis is as follows. Chapter 2 is, in a manner, the second part of the introduction. In the chapter, characteristics of wireless sensor networks are discussed that are relevant for the design of WSN data link layers. The application layer ultimately determines how much data nodes generate and imposes requirements like message travel times etc. In addition, representative WSN routing protocols and their requirements are discussed. The chapter provides an overview of data link layer requirements, such as, for example, energy-efficiency, scalability and self-organization capabilities.

Chapter 3 gives a taxonomy of channel sharing methods and provides an overview of the state of the art of MAC protocols for WSNs. In the chapter, we make a selection of MAC protocol classes that fulfil our requirements best.

Next, we present a general medium sharing method in Chapter 4. The method is based upon scheduled access (often called TDMA), which is well-suited for WSNs because of its energy-efficiency and robustness against high peak loads. We present a simple, yet effective algorithm, which allows nodes to be self-organizing on a data link layer level and provides conflict-free communication in a multi-hop network setting. Verification efforts have been taken to increase the trust in the correctness of the algorithm in Chapter 5.

Chapter 6 discusses EMACs, a medium access control protocol designed for wireless sensor networks. The protocol is a concrete appearance of the work presented in Chapter 4. Its key feature is an incorporated clustering mechanism, which creates a connected backbone in the wireless sensor network. Nodes that are not part of this backbone use other nodes to carry out energy-consuming communication tasks. Consequently, the lifetime of the wireless sensor network is prolonged. In addition, we present cross-layer optimization for the EMACs protocol and a multi-hop routing protocol.

In Chapter 7, we sketch LMAC. This schedule-based MAC protocol is tailored to the special communication needs of wireless sensor networks. The protocol includes routing towards designated gateway nodes in the network. In schedule-based MAC protocols, latency is often a major concern. In Chapter 7, we present mechanisms to reduce latency and show their effectiveness by simulation.

In a network where sensors communicate wirelessly, it is particularly easy to eavesdrop or inject false messages. When attackers have full knowledge of the MAC protocol in use, the WSN can easily be attacked by misusing link layer functions of well-behaved nodes. Chapter 8 discusses attacks and countermeasures. Our focus is on the LMAC protocol, however, SMAC and BMAC —two representative MAC protocols for wireless sensor networks– are also examined. Our simulations are validated for LMAC by real-live jamming experiments.

Chapter 9 provides conclusions.

Chapter 2

Characteristics of wireless sensor networks

Resource-constrained, deeply embedded sensors communicate wirelessly and collaborate in establishing a representative description of their environment. This chapter provides insights into relevant characteristics of such wireless sensor networks*.*

Wireless sensor networks are often application specific networks. By this, we mean that depending on the inherent behaviour of the application, optimizations can (and should) be carried out. Consequently, wireless sensor networks do not exactly fit into the view of traditional communication systems, known as the OSI model. Special protocols need to be designed to accomplish network lifetime requirements.

We sketch a few challenging application scenarios for wireless sensor networks and use them to derive common characteristics, augmented by literature. These are invaluable for the design of networking layers and —especially to our interest– the MAC protocol.

2.1 Introduction

In this chapter, an overview is provided of the characteristics of wireless sensor networks. The main focus is on sensor data that must be transported by the network (size, frequency, transport patterns, etc.) however, attention will also be paid to communication requirements of in-network reconfiguring and collaboration between nodes. These are important aspects to provide a set of requirements for the medium access control protocol.

The outline of this chapter is as follows. Firstly, some example applications of wireless sensor networks are described. Secondly, we focus on the requirements these applications impose. Next, we describe representative WSN routing protocols and discuss what communication is required to establish routes in multi-hop networks. In Section 2.6, we provide requirements for WSN MAC protocols and conclusions.

2.2 WSN applications

In this section, a few application examples are given of wireless sensor networks. We present here only a selection of the numerous applications envisioned for wireless sensor networks. We silently leave out military (sniper locating, battlefield surveillance), health care (tracking and monitoring patients in hospitals, drug administration), home/office automation (climate control, lighting control, asset monitoring, machine-to-machine applications), smart kindergartens [99] and many other applications. Our selection of applications is motivated by our assumptions in Section 1.1.

The focus lays on the (common) functional properties of the applications. How often is data required from the wireless sensors? How does the data flow through the network? Based upon the typical WSN applications described in the following sections, we build a set of general application requirements (Section 2.3). Underlying layers and protocols should fulfil or enable this set of requirements.

2.2.1 Environmental monitoring

We give here three examples of applications in environmental monitoring: *habitat monitoring*, *precision agriculture* and *irrigation advice*. These three applications will be discussed shortly, but many more applications in this area can be imagined: early forest fire detection, dike integrity monitoring, volcano monitoring, etc. In contrast to the other applications we will discuss in this section, the network is not involved in the interpretation of the data. The main interpretation and use of the data is left to biologist and ecological experts or centralized control models.

The applications in the area of environmental monitoring all have in common that their main task is to collect and store sensor readings for future reference. Therefore, it is not required that sensor readings are reported directly upon acquiring. Although short latency is an important requirement in many communication systems, in most WSN applications relatively long intervals between the generation of messages and their delivery can be tolerated.

2.2.1.i Habitat monitoring

In the recent past, environmental monitoring trials have been carried out with wireless sensor network technology [100, 101]. A habitat of seabirds was closely monitored on an island during a four month period. Inside the underground burrows of the birds, infrared sensors were placed, monitoring the occupancy of the burrows. On the island itself, climate sensors (like temperature, humidity and rainfall sensors) were placed, to get a good impression of the climate on the island and its effect on the behaviour of the seabirds. All the sensor data was collected remotely and was off-line processed into graphical overviews.

To meet the requirements of biologists, the sensor nodes in this application sampled their sensors once every 70 seconds. The resulting data packet consisted of 36 bytes, including timestamp. The TinyOS radio stack (see `http://www.tinyos.net`) was used to transmit the packet. All generated data was forwarded to a central point in the small multi-hop network.

The sensor nodes also provided the hardware status, the network status and the quality of the radio links. Besides several interesting biological aspects, the authors found some unexpected failing of their node hardware. For example, the logs showed a strong relation between rain and fog, and the quality of the wireless communication. Moisture in the air reduced the quality of the radio links significantly.

In addition, in the harsh environment of the island, the sensor hardware rapidly wore down and several sensors seized operation after only a few days. Again, moisture seems to have a negative influence on the packaging of the electronics. This points out that in rollout of a wireless sensor network, the packaging is also an important factor in the node lifetime.

2.2.1.ii Precision agriculture

Precise treatment of crop diseases has many benefits. It limits the use of —often– environment-unfriendly substances like pesticides, and it reduces farming time and costs.

In [60], an example of precision farming is described. In this case, the microclimate around potato crops is monitored to estimate the risk of phytophthora, an easily-spread fungal disease, which can destroy a harvest. Spores develop on the leaves of the potato plant when the temperature is above 10 ^{o}C and humidity is over 75% for 2 days or more (see `http://en.wikipedia.org/wiki/Phytophthora_infestans`). These two parameters —temperature and humidity– are therefore closely watched by a dense deployed WSN.

The above simple rule to assess the risk of infection is a good example of the behaviour of WSNs in general. Sensor readings, as such, are often not interesting to know in detail, however their interpretation is. In this case, the farmer wants insight into areas that are at risk of infection, so action can be taken with preventive measures. The wireless nodes could locally record temperature and humidity and assess the infection risk. After checking the validity of their results with their neighbouring nodes, the assessment is transmitted to the point where the farmer can access the readings.

In [60], the nodes collected sensor samples once every minute and ten temperature and humidity samples were combined into a short data packet. The data packets are forwarded to a central server, where they are further processed and presented to the farmer in graphs.

2.2.1.iii Irrigation advice

To keep the grass on sports fields in optimal shape, they have to be irrigated regularly in dry periods of the year. Especially golf clubs are keen on keeping their playing fields in perfect shape and often have full time green keepers in service. Irrigation is a difficult and costly task. In the first place, water becomes extremely expensive when regulatory quotas are exceeded. Thus, irrigation should be well planned, so that the quota is not exceeded. This is naturally very difficult since the weather is —(un)fortunately– unpredictable. Secondly, the colour of grass is often the only feedback on the irrigation policy, while the actual water content of the soil is neglected.

Research is done in the area of irrigation and drainage (see for example [6]). The goals of this research are to provide a model that provides a fitting irrigation strategy based on soil structure, the current moisture in the soil, salinity of the soil, vaporization, weather forecasts and the moisture needs of the grass. Note that these parameters are slowly changing over time. Thus, high sample rates are unnecessary and the wireless sensor network can be inactive for long periods.

A wireless sensor network can provide valuable input to the model, while providing ease of installation. Soil moisture sensors can be put underground, to give actual numbers on the water content of the soil. Climate sensors can be spread throughout the greens to measure e.g. sunlight intensity to predict the amount of water that is vaporizing. These sensor readings can be fed into the model and eventually the system plans the irrigation or gives feedback on the irrigation in progress, in the end saving on water and costs.

How much data is generated by soil moisture sensors? The soil moisture sensor ECHO-TE [26] —for example– measures volumetric water content (4 bytes), electrical conductivity (4 bytes) and temperature (3 bytes) sequentially, resulting in 13 bytes per measurement including a timestamp. Taking measurements once every quarter of an hour would already give a good view on the trend of the soil moisture.

2.2.2 Transportation and logistics

2.2.2.i Car park monitoring

It is often difficult to find an empty car parking space in an unfamiliar city. In addition, local government has no reliable indication of the occupancy of parking spaces and, therefore, the construction of new spaces may not meet the actual requirements. These two issues can be solved in an application based on a wireless sensor network.

Assume that a wireless sensor is built in the concrete of the road [57]. The state of a parking lot can be monitored using —for example– a magnetometer. When a car is parked on top of the sensor, a change occurs in the magnetic field, indicating that the space is taken. This example sketches that the information that is generated by a wireless sensor can truly be very small; in this case even a binary state. The wireless sensor transmits the state of the parking space above it only when it changes. The data of all wireless sensors is collected at a central point, where it can give accurate insight into the occupancy of parking spaces and can provide correlation with the received parking money. Additionally, the same wireless sensor network can be used in conjunction with road signs to indicate cars arriving in the city centre where the chances are best to find a free parking space.

2.2.2.ii Warehouse monitoring

Warehouses are usually large buildings, where many goods are stored for longer or shorter term. When goods are delivered, they have to be registered, including the place where they will be stored in order to be able to locate the goods if they have to be transported elsewhere.

Warehouse monitoring is especially interesting in warehouses where is dealt with fresh goods, like vegetables or flowers [23]. By law, the temperature has to be recorded of these goods and customers must be able to access these records. Transport and logistic companies suffer the consequences for spoiled fresh foods, if they cannot show that the goods remained within temperature bounds. To prevent these costly claims, transport and logistic companies closely want to monitor temperature of the goods and act upon any out-of-bound temperatures. This monitoring is an excellent task for a wireless sensor network.

Temperature sensors cannot be attached to the fresh food itself (hygiene) and handling costs of placing temperature sensors close to the products is too expensive. A logical and sufficient solution is to equip the trolley —in which the products are transported– with several temperature sensors that can monitor the temperature of the products carried. The carts in the warehouse form a wireless network and assist each other in forwarding temperature readings to a central database. For most applications it would be sufficient to obtain a temperature reading once every few minutes.

The wireless network can carry out even more complex tasks than product monitoring. As an example, we give here the checking of the logistics process. When trolleys are made aware of what products they are carrying, the network can assist in localizing goods (Section 2.3.4) and the carts can detect whether they are placed in the right trailer for transportation. In this process, the *presence* of a sensor node is used as additional information, comparable to (active) RFIDs.

2.2.2.iii Chemical drums

An interesting application scenario is sketched in the CoBIs project (EU IST-2003-4270) and is situated around chemical plants. For safety reasons, chemical storage facilities have a certain maximum capacity for hazardous materials and certain compounds are not allowed to be stored together. To detect potentially hazardous situations, the chemical drums are equipped with wireless sensor nodes. These nodes monitor storage conditions of individual drums, but also raise an alarm when chemical reactants are in proximity, or when storage regulations are violated. In this case, the wireless sensors use their communication modality also as a sensing modality.

2.2.3 Control tasks

Besides monitoring the physical environment, nodes can also perform (simple) control tasks. In this case, a wireless sensor network is comparable to the human central nerve system. Simple responses can be solved locally (for example turning on heating if a room is below a specified temperature or closing the window blinds if the light intensity is above a threshold) i.e. the reflexes of the network, while more complicated tasks are processed by a central controlling system (like minimizing the total energy consumption of the heating system [84]).

Timing critical control tasks are more difficult to execute. Since the network is energy constrained, the nodes will be often in a low-power mode, conserving as much energy as possible. This dramatically reduces the response time of the wireless connected sensor(s) and actuators. We have to think here in the order of seconds of reaction time, ruling out quick feedback loops. The energy constraint on wireless sensor nodes makes these networks unsuitable for reactive controlling.

The approach chosen in ZigBee ([114], Section 3.4) is to connect part of the network to the mains to enable continuous transceiver operation (see also Section 3.4). In, for example, a wireless light switch application, the node connected to the lamp (and thus no energy constraint) is actually continuously receiving, while the wireless switch only wakes up when the user presses the button. In wireless sensor networks, we expect the energy constraint to be more homogeneous among the devices, because sensing and communicating are combined.

2.3 Requirements from applications

In the previous sections, we describe several WSN applications. In this section, we provide insight into the requirements these applications impose.

2.3.1 Network lifetime

Cheap and tiny sensor nodes allow for fine-grained observation. As a result, sensors might be deployed densely and in large numbers. The large scale of those pervasive networks makes it virtually impossible to replace or refresh power sources on the individual nodes. It is most likely that the nodes are placed unobtrusively in the environment (i.e. built into walls etc), complicating the task of energy replenishment even more.

On the other hand, the user of the network expects the network to be cost effective and thus to have a lifetime of many years, depending on the application type. Therefore, *energy* is assumed to be a very scarce resource and its limited availability is the most challenging problem in WSNs. Note that WSNs are always *on*. This is one of its unique characteristics.

Power sources can be roughly divided into two classes (see also Section A.2.3): (1) batteries and (2) energy scavenging i.e. energy extracted from the environment by solar cells, piezo elements, heat converters etc.

Batteries —depending on the type– deplete even when not used and make sensor nodes truly throwaways. Energy scavenging is mostly the preferred solution to the energy problem, since light, vibrations etc can be assumed to be available, at least during some parts of the day. However, the efficiency of most methods is still doubtful. Energy scavenging requires more attention in the research community.

How can a network cope with lack of energy? In general, redundancy is assumed in WSNs to deal with failures of any kind, like (temporary) energy depletion of the nodes. However, this solves only part of the problem. Obviously, hardware and protocols must be designed together to be energy-efficient. But also globally energy can be saved using functional clustering and data aggregation. These topics are discussed in the next sections.

2.3.1.i Functional clustering

The purpose of clustering is to create a hierarchical structure in the network and to reduce complexity and thereby energy consumption. This is —for example– used to allocate certain tasks in the network to some of the nodes; other nodes can then conserve energy. In fact, clustering makes the network less *homogeneous*. In some applications, heterogeneity is even introduced by deploying a mix of capable (i.e. mains powered, computational powerful etc.) and resource constrained devices.

In Chapter 6 [42], we describe an algorithm based on the clustering principle, that enables the network to create —autonomously– a connected *backbone* in the network of *active* nodes, while the other nodes reduce their activity in wireless communication to a large extent. Those, so called, *passive* nodes can still transmit messages using the created backbone.

To create these hierarchical structures in the network, nodes elect (local) head nodes [10]. For this we need local communication between sensor nodes. From time to time, cluster heads release their state as cluster head to ensure that they do not run out of energy too quickly. This release triggers (local) re-execution of the clustering mechanism.

2.3.1.ii Data aggregation

Since there might be redundancy in the data that sensors generate, packets can be aggregated [13]. By aggregation we mean the reduction of the number of packets and the overall size of the transmissions by the following mechanisms: (1) removing duplicates and (2) calculating summarizing statistics like average values. Nodes apply these mechanisms on their own data and data they forward on behalf of other nodes.

The benefits of aggregation are evident. By reducing the amount of data that is transmitted, the lifetime of the network is increased. In Section 2.3.7, we argue that wireless nodes transmit most data to a so called gateway node (see for example the precision agriculture application in Section 2.2.1.ii). Obviously, when we are dealing with a large network, the volume of data forwarded to this node is tremendous. Therefore, in-network data aggregation is a useful mechanism. We tacitly assume here that computation required for aggregation is less energy-consuming than communication. This is of course dependant on the complexity of the aggregation operation.

2.3.2 Event detection and streaming data

It is important to distinguish between two ways of looking at the physical environment in the above-described applications:

- **Streaming data** — Sensors can be sampled continuously. This stream of data can be processed locally (i.e. noise reduction, compression etc.) and then sent into the network. In this scenario, a node generates data at a constant rate and this type of measurement is useful to study, for example, acid concentration in rivers over time or camera surveillance.

- **Event detection** — This measurement type is especially useful in alarm systems. For example, when the temperature rises above a defined threshold or when two chemicals are coming close (detected by sensors in their barrels), nodes report this. This measurement type generates, in principle, less data to be transported by the wireless sensor network, but node activity might be very high when multiple nodes detect the same out of bound physical phenomena.

  Many short data packets are generated around the same time, describing the same event in the network. A node that receives multiples of the event detection packets might combine them into one larger packet, which reduces the overhead in the network.

  Inherently, the network can be inactive for long periods of time when no events occur.

Note that the main difference between streaming data and event detection does not lay in the *sample frequency* to obtain the sensor readings, but in the way the measurements are handled. In the case of event detection, the sensor node must be provided with information on what are *normal* sensor values and what is out of bound. Thus, responsibility of interpreting sensor information is moved to the sensors themselves and an important benefit of this is that the volume of data that has to be transported by the network is naturally small.

An obvious drawback of event detection is that we as users of the wireless sensor network are not able to distinguish between the case that a sensor is operating correctly (and the sensor values are within the specified range) and the case that the sensor seized operation. For this purpose, the sensors should notify their presence from time to time. In Section 2.3.6, we discuss how the operation of the sensor nodes can be controlled from outside. Wireless sensor networks must be able to cope with both types of measurements, and even a combination of the two.

We note here that the size of the transmitted information generated by a node is not large. Typically, a (pre-processed) sensor reading fits in a few bytes, however some sensors —like digital cameras– are an exception. Nevertheless, we argue that this type of sensor only would send its captured image when something important is detected. Although the packets are small, many different sensors might react to the same event, thus many packets might be generated at the same moment.

2.3.3 Time stamping

Without knowing *when* a sensor sample was taken, a measurement is —of course– less valuable. For most applications, however, it is not important for the samples to arrive instantly at their destination. In other words, applications usually do not have very strong requirements on the latency induced in the network (Section 2.3.7.ii).

In the environmental monitoring examples described in Section 2.2.1 the sample time granularity is in the order of minutes or hours. It would therefore be sufficient if the wireless sensors collect time stamped samples, store them internally and report them —say– once every day. However, the fear for failure in the network or the unexpected reset of sensor node hardware (as in [101]) makes it so that we are reluctant to store readings inside the network.

The time stamping of samples requires the wireless sensors to be either synchronized with each other (and preferably with a good clock source), or to cooperate in updating the age of the message in the network. The later scheme is very simple to implement. Upon receiving the message, a node has to record the arrival time using its own —potentially unsynchronized– clock. The node calculates how long the message has been in its buffer and adds this time to the age that is carried with the message just before sending out the message. When the message finally arrives at its destination, the time of creation can simply be calculated. Either scheme —time stamping or delay tracking of messages– will suffice for wireless sensor networks.

2.3.4 Node position

To interpret the sensor value, the location of *where* a sensor reading was obtained must also be known. Depending on the kind of application, the unique identification number of the node that reported the sensor reading might be sufficient location information. However, in literature many *localization algorithms* are described that estimate (relative) coordinates of nodes. We classify the many approaches in two classes:

- **Range based [37]** — Distance between nodes can be estimated using various techniques, including received signal strength, time difference of arrival, etc. Note that some of these techniques require specialized, potential energy-consuming hardware to carry out the estimations. When a node has estimated its distance to enough nodes around it with known positions, it can estimate its own coordinates.

- **Range-free [78]** — Range-free localization schemes use topology information to estimate their coordinates. Combining information on nodes that can be heard and information on nodes that cannot be received along with their coordinates allows a node to create inclusion areas (i.e. the area in which the node needs to be). This technique does not need special hardware to estimate the node's coordinates.

Most localization algorithms consist of two phases: (1) an initial phase in which nodes discover nodes in range and make a first estimate of its coordinates, and (2) a refinement phase in which nodes use estimated positions of other nodes to improve their own estimate.

Both phases have in common that nodes need to exchange messages with (all) nodes in radio range. Localization might be considered communication intensive, certainly when the application requires the nodes in the network to update their estimate frequently, for example when the mobility in the network is high.

2.3.5 Authentication, data integrity and security key management

It is a key issue that any data generated by the wireless sensor network can be trusted [97].

When can data be trusted? We recognize two aspects: (1) source authentication (is the data originating from a trustworthy source?), and (2) data authentication (is the data not altered during transport?). Standard cryptographic techniques can provide solutions for these aspects. Often they are based upon establishing shared security keys between source and destination nodes. Unfortunately, these methods are often computational intensive [97].

Karlof et al. propose TinySec [54], a link layer encryption mechanism that provides the above mentioned aspects and also message secrecy. The core of TinySec is an efficient block cipher and keying mechanism that is tightly coupled with the Berkeley TinyOS radio stack. TinySec makes use of a single, symmetric key that is shared among the wireless sensors. Before transmitting a packet, each node first encrypts the data and applies a *message authentication code* (MA code), a cryptographically strong, not forgeable hash to protect data integrity. The receiver verifies that the packet was not modified in transit using the MA code and then deciphers the message.

For the MAC protocol the requirement of authentication and secrecy involves (1) increase of message size, e.g. added MA code to the message, (2) additional traffic due to security key management and (3) authentication of MAC control messages. In Chapter 8, attack and defence of WSNs at data link level is further discussed.

2.3.6 Configuring the WSN from outside

During the lifetime of a wireless sensor network, the interest in collected data may change. For example, this is likely in environmental monitoring applications ([15], Section 2.2.1). Therefore, it is useful to be able to *specify* to the network, which measurement type should be used, from which sensors and at what rate data should be collected. The nodes then adjust their operation accordingly. This subject is of importance, because the data link layer has to transport this configuration data reliably to relevant sensor nodes.

2.3.6.i Reprogramming

The easiest way of reconfiguring a sensor node is to reprogram the nodes with updated firmware code. This is useful especially during the debugging of a sensor network. However, in most cases reprogramming is too expensive in terms of energy consumption. Note that programs for wireless sensors are around a few tens of kilobytes in size (Appendix A). This relatively huge amount of data has to be forwarded by intermediate nodes, draining their energy supplies as well. At its destination, the updated program must be written to (flash) memory, which requires a high current.

Reijers et al. [86] propose a reprogramming mechanism based upon efficiently transferring code changes. The program code on the node is basically patched by a *change script*, executed on the nodes. Levis et al. [63] let nodes transmit the version number of their firmware at a slow pace. When they (locally) detect that a neighbouring node has old firmware, neighbours with more recent firmware automatically start update procedures (a similar scheme, Deluge, is presented in [47] and is included in the TinyOS distribution). Both schemes make use of underlying networking protocols to transfer program code. Therefore, it remains a complex process to update the networking part significantly.

2.3.6.ii Querying the network

Instead of thinking of the network as executing pre-programmed tasks, one might also consider the network as a distributed database, which can be queried for sensor readings. In this setting, the nodes are pre-programmed with a query interpreter, which parses incoming queries and changes the node's behaviour accordingly. We give here two examples of queries: one of streaming data and one of event detection. The examples are taken from [68].

When we are interested in light and temperature readings, sampled once per second for a time interval of 10 seconds, the following query is given to the network:

```
SELECT nodeid, light, temp
FROM sensors
SAMPLE PERIOD 1s FOR 10s
```

In this query, nodes are asked to return their *identification number* (ID) as well. Note that this is an example of *streaming data*. An example of event detection query is:

```
SELECT nodeid, temp
WHERE temp > 25ºC
OUTPUT ACTION alarm()
SAMPLE PERIOD 60s
```

In this example, nodes in the wireless sensor network are asked to monitor temperature once every minute and when the temperature is above 25^oC, they need to report their ID and temperature and they execute the action `alarm()`. Note that the above queries do not specify the destination of the data, but assume knowledge in the network where the interest in the sensor readings is.

Considering the queries above, we can predict the amount of data that will be generated by nodes in the network. Namely, every node with a temperature and a light sensor —in the first query– will generate messages which contain node ID, and recent sensor readings. Ten messages will be generated in total per node at a rate of one per second. Suppose there are 8 of such nodes in the network and a message has a length of 6 bytes —if we assume a 2 byte ID and 2 byte sensor readings–, we expect the network to deliver 48 bytes per second during 10 seconds. The wireless sensor network can adapt its operation to this expected rate and conserve energy by, for example, adjusting its active/sleep duty cycle [113].

It is more difficult to predict the data volume that will be generated by the second query, because in this query we need knowledge of the actual temperature. Nevertheless, if we know from previous queries what temperatures roughly are at nodes, we can still do some estimation. If a node detects a temperature higher than 25^oC, it will generate a message of 4 bytes every minute.

Being able to estimate the amount of data that has to be transported by the network can be of importance in designing networking protocol for wireless sensor networks. The above described applications can be described using complex sets of queries. All nodes in the network should be able to understand the queries that are sent to them. This can be done by a *query interpreter*. The network can also benefit if the network is able to aggregate the results of query results. In general, this reduces the data volume that has to be transported by the network. In [68], a framework for query based sensor networks is presented: TinyDB.

2.3.7 Conclusions on application requirements

In the following sections, we briefly discuss our conclusions on general wireless sensor network application aspects.

2.3.7.i Typical communication patterns

We distinguish three typical communication patterns in WSN applications (Figure 2.1):

1. **From central point(s) to sensors** — We assume that there are one or more gateways in a wireless sensor network. These gateways allow users to specify or change the operation mode of the network (i.e. sample rates etc). This is typically done by creating and injecting *queries* [68] or *business rules* [70] into the WSN. Such a query or rule should be propagated to every node in the network. A gateway also presents the results of the WSN to the user. Thus, gateway nodes create a bridge between the outside world and the WSN.

2. **From sensor to central point(s)** — This path is to be taken by messages containing sensor readings or detected events, which are of interest for the outside world. These messages have to be forwarded until they reach a gateway, where they will be presented to this outside world. This type of communication is used far more than the other two types.

   Due to local pre-processing of raw sensor samples in the nodes, in network processing and aggregation, the size of the generated messages is small (in the order of a few tens of bytes). This small message size is typical for WSNs, but so are high peak loads in the network, due to a physical event, which is detected by multiple sensors.

3. **Local communication from sensors to sensors** — This type of communication is required for creating functionality beyond acquiring sensor information. Examples are functional clustering, security key management and authentication, localization, consensus algorithms, routing, etc. Often, this communication pattern makes use of local broadcast i.e. a node transmits data to all its neighbouring nodes.

2.3.7.ii Latency

Wireless sensor network applications can be event based and/or have a streaming nature. In the case of streaming data —like in the habitat monitoring and irrigation advice applications– (multi-hop) latency i.e. time between generation of the sensor sample and the delivery of it at its final destination, is of little importance. In the order of seconds or even minutes is tolerable for most streaming data applications, certainly when readings are analyzed posterior and in bulk. However, when the WSN detects

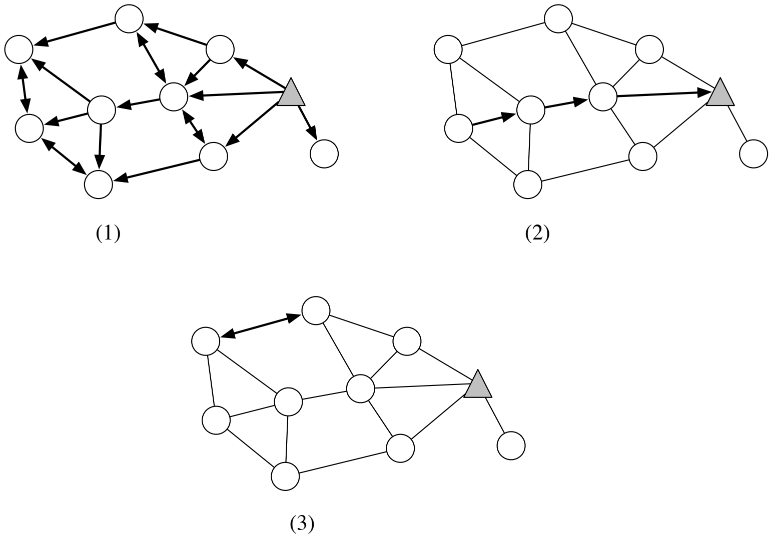


Figure 2.1 — Typical communication in WSNs: (1) injecting a *query* or configuration data, (2) reporting of sensor readings and (3) local communication (gateways are represented by gray triangles)

critical events, such as for example in the warehouse application (Section 2.2.2.ii) —where wireless sensor nodes measure temperatures of fresh goods and give warning when these are out-of-bound–, short latency is generally more important, simply because action needs to be taken to solve the unwanted condition.

Clearly, per application the trade-off should be made between latency and energy-consumption. A short latency requires highly active (multi-hop) networking and therefore the nodes consume more energy and battery lifetime is reduced.

2.3.7.iii The timescale of application related events

In Figure 2.2, the tasks of wireless sensor networks are summarized along with their time granularity. Depending on the actual application, the figure might look different, however, we give here an impression of the activities in a wireless sensor network.

Most often, the sensor network will be concerned with the transport of sensor data. Time intervals between the generating of these packets may vary between seconds to hours depending on the nature of the application. More infrequent is communication for time synchronization, localization and clustering. Packets are exchanged from once per minute to once per day or even less, assuming moderate mobility in the network. Reconfiguring and security re-keying happen infrequently and reprogramming is unlikely after debugging phases. Consecutive data exchange for these application-related issues occurs in the order of days, months or even years.

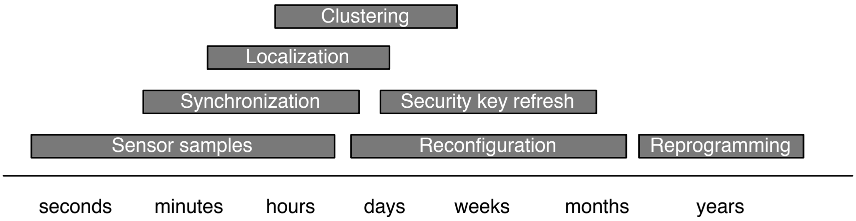


Figure 2.2 — Typical frequency of application related data exchange

2.4 Routing protocols for WSNs

In this section, a short overview is given of recent routing protocols for wireless sensor networks. We discuss *EYES Source Routing* (ESR), *Directed Diffusion* and *Geographic Random Forwarding* (GeRaF). These protocols use different approaches and strategies to forward data to its final destination and have been selected for their representativeness in WSN literature. An overview of WSN routing protocols is provided in [53].

2.4.1 EYES Source routing protocol

The EYES Source Routing (ESR) protocol [107, 42] is an on-demand routing protocol, which enables dynamic, self-starting, multi-hop routing to be established when a source sensor node wishes to send a data packet. The protocol can be used to address individual sensor nodes i.e. by the *identification number* (ID) of the node. All routing messages in ESR are small and have a fixed size.

The ESR algorithm has two main phases: (1) the route setup phase, which is used to discover new routes, and (2) the route maintenance phase, which is used to repair broken links. The two phases are discussed briefly.

Initially, when a node wants to send a data message to another node, no prior knowledge of the location of the destination is available. In this stage, the source has to *flood* the network with route requests (i.e. each node repeats requests using broadcast) in order to notify the destination that it has a packet for it. The length of the route request message is small to minimize the energy required in the flooding; it contains, in addition to the ID of the targeted destination, the ID of the source node and a sequence number to ensure freshness of the request.

The targeted destination node replies to the first received route request and discards duplicate ones. Instead of flooding the route reply to the original source node, it only replies to the node from which it received the route request first. This node in turn forwards the route confirm message only to the node from which it received the route request first. This is repeated until the message reaches the source node of the route request. The intermediate nodes on the route do not store full information of the route, but only the information to which node messages to the destination should be forwarded.

However, links between nodes can break and new links can appear, e.g. due to mobility. Constantly re-establishing routes by re-flooding the network with route requests is very energy consuming, since this requires activity of all nodes in the network. The ESR protocol can recover lost links in a local and fast manner, so that the frequency of network wide route re-establishments is significantly reduced. In other words, it has been designed with dynamic conditions in mind. For this purpose, the protocol makes extensive use of tables containing information on local neighbours (see also Chapter 6).

2.4.2 Directed diffusion

In the previously discussed routing algorithm ESR, nodes are addressed based on their IDs. The authors of directed diffusion [50] describe a different communication paradigm: instead of addressing nodes by (unique) ID, data flows in the direction of nodes that expressed an interest for that particular data. An obvious advantage is that nodes are not required to know the ID of the destination. However, there must be a network-wide naming of data. Note that in ”traditional” routing protocols source nodes are pushing the data to its destination, where in the directed diffusion framework, nodes with an interest in the data pull the data towards them, maybe a more natural concept for gathering sensor data.

As introduced above, the directed diffusion framework uses the concept of *named data* and interests. Each node in the network can specify its interest in a certain type of data and can contribute in answering interests. We focus here on the routing part of this protocol, however, it requires also *interest awareness* at application level.

First of all it is important to make clear the concept of interest. The authors give the following example [50]:

```
type = four-legged animal
interval = 20ms
duration = 10s
rectangle = [-100, 100, 200, 400]
```

In the above example, an interest is specified by a node that wants to be informed when a four-legged animal is detected in the specified area, requesting reports every 20 ms for a period of ten seconds. This interest is broadcasted into the network. All nodes are required to keep record of the interest and store the node that broadcasted the interest to them first. This information is used to forward replies back to the originator of the interest. When a device detects a ”four-legged” animal, it generates —for example– the following reply:

```
type = four-legged animal
instance = elephant
location = [15, 310]
time = 13:14:15
```

This kind of specifying interest has similarities to the querying mechanism of TinyDB (Section 2.3.6.ii, [68]). A major difference, however, is the fact that in TinyDB all data is assumed to flow to the (i.e. one) gateway in the network. In directed diffusion, the data is returned to the node that broadcasted the interest.

Note that the *interest message* compares very well to the route request message in ESR (however more information is included in the former message). This concept is generally used in routing protocols to establish routes: first step is always the flooding of the network with a request.

Directed diffusion is used as routing mechanism in Chapter 8.

2.4.3 Geographic Random Forwarding

Yet another message routing strategy for wireless sensor networks is described in [116]. Instead of advertising an interest for data, or requesting to establish a route to a certain destination device, nodes use in this scheme a routing technique based on node coordinates. Nodes are assumed to know their own position and the position of the sink node (i.e. the node where the data needs to be delivered).

The idea is that nodes advertise data along with the coordinates where it must be delivered to. Nodes closer to the sink node consider themselves candidates for relaying the message. Zorzi et al. [116, 117] combine this routing mechanism with data link mechanisms and show the effectiveness of GeRaf in a multi-hop environment.

This routing protocol deviates from the described protocols by skipping the initialization phase, which is usually broadcasting request(s). However, node positions and destination positions are required for efficient routing (see also Section 2.3.4). Hence, the GeRaf protocol performs best in static environments, in which these locations are not subject to changes.

2.4.4 Conclusions on routing protocols

Nodes in a multi-hop wireless sensor network collaborate in forwarding data to its destination(s). By the term routing protocol, we understand the mechanisms to select the "best" node out of the set of nodes in radio range for forwarding the data to its final destination. In general, routing protocols try to optimize *global* performance (i.e. for example minimize network-wide energy consumption) by making local decisions on the best node to forward the data to.

Routing protocols developed for wireless sensor networks characterise themselves by being ad hoc and on-demand, meaning that routes between source and destination nodes are only created when being used. Typically, those routes are enforced when data is transported on them and die off otherwise.

Ad hoc, on-demand routing protocols are well suited for WSNs, since the protocols are by design able to cope with mobility and node failure in the network. Time out of route information or failure of intermediate nodes between source and destination cause routing protocols to discover new routes, which guarantees a high reliability and up-to- date route information in the nodes. However, the discovery of new routes is typically an energy inefficient process, because a route-establishing message is flooded throughout the network. The medium access control protocol needs to efficiently deal with the (short) messages generated by routing protocols to establish, maintain or repair routes.

When establishing a new route from source node to destination node, the metric is often the length of the route (measured in hops), but also different metrics have been presented for wireless sensor networks: latency, energy available in intermediate nodes, capacity left of the intermediate nodes, etc.

2.5 WSNs compared to traditional computer networks

To sketch the position of *wireless sensor networks* (WSNs), we impart a brief overview of the history of computer networks. In the beginning of the 1960's [104], the first steps were taken to *packet* based communication networks. Nowadays, the notion of packets —i.e. the data, which has to be transferred, is divided in small portions with fixed length and the so called datagrams are labelled with source and destination information– is quite common and is for example used for satellite communication and even phone calls are packetized.

In 1974 protocols were being developed to be able to connect different computer networks together and to let them communicate. These protocols are now know as the *internet protocol* (IP) and the *transport control protocol* (TCP). The functionality, reliability and flexibility of these protocols enabled the fast growth of the internet. Around the same time the *open systems interface* (OSI) has been described ([115], Section 2.5.1) to provide abstraction for network communication protocols.

Ten years later, the university of Wisconsin developed the *domain name system* (DNS), an important step in making computer networks more accessible by providing human readable addressing of computers instead of numbers that had to be memorized. Without this convenient naming, browsing the World Wide Web would never have become popular.

Until the 1990's, computer networks were closely tied to military and research centres. In 1992, the internet was born and *internet service providers* made commercial connections to the internet available. Since that time more and more internet enabled devices have emerged.

Nowadays, the layered approach as presented in the OSI model is commonly accepted and the networking protocols in wireless sensor networks follow a similar pattern. However, we argue in Section 2.5.2 that the requirements for wireless sensor networks are somewhat different from traditional computer communication networks and hence the layers of the model are more compressed than in other communication networks.

2.5.1 General framework: the OSI model

The open systems interface model ([115], Figure 2.3), intends to create a framework for communication protocols between heterogeneous networking entities. The OSI model aims to make communication systems independent. In other words, it ensures that a windows-based PC can view a webpage on a UNIX web server or that a PDA can read e-mail. To achieve this inter-operability, the OSI model defines seven layers:

1. **Physical layer** — This is the bottom most layer in the model. The *physical* (PHY) layer is in charge of controlling and maintaining physical connections between devices. In wireless sensor networks, this means the control of the transceiver (i.e. transceiver state switches and feeding or accepting bits to/from the transceiver).

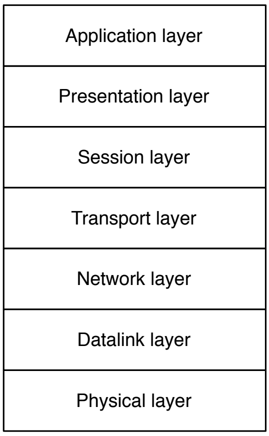


Figure 2.3 — The seven layers in the OSI model [115]

2. **Data link layer** — The data link layer establishes data links (i.e. link at packet level) between nodes. This layer controls the PHY layer in such a way that data can successfully be transferred between nodes. The MAC protocol is a special part of the data link layer. It defines rules for accessing the medium and thereby facilitates reliable communication.

3. **Network layer** — This layer takes care of efficient *routing* of data. Nodes between source and destination forward the data appropriately (and no higher layers are involved in this process). In modern computer networks, the well known *internet protocol* (IP) takes care of the network layer.

4. **Transport layer** — The transport layer relieves higher layer entities from any concern with the transportation of data between them. This layer may arrange re-attempts of erroneous packets and it controls the rate of packets in order to avoid congestion in the network. The *transport control protocol* (TCP) and *user datagram protocol* (UDP) are often used in modern computer networks.

5. **Session layer** — The session layer takes care of setting up supervised connections between end-systems.

6. **Presentation layer** — The task of the presentation layer is to interpret the received data to the benefit of the highest layer: the application layer. Operating systems are often taking care of this layer.

7. **Application layer** — This layer forms the interface to the application. It provides means of authentication and secured data exchange. In addition, it defines the services that are provided or the destination for the data.

2.5.2 How WSNs fit in the OSI model

The OSI framework is as general as possible to allow a large spectrum of services to exchange data using similar underlying structures. This means that many application protocols (like HTTP, POP, FTP, etc. for computer networks) can operate at the same time using the same protocol stack and inter-operability between heterogeneous platforms is ensured. Consequently, the underlying protocols of the OSI model are not necessarily optimized for specific applications.

On the contrary, traditional communication protocol stacks assume an excess of resources and can spare the energy and memory to send many messages. The OSI model assumes for example that every layer adds independently its own *state information* to the actual data that has to be transmitted. This is, of course, the only way a layered approach can function correctly. Each layer should only deal with its own data structures leaving other layer's data untouched. This introduces overhead, but keeps the implementation clear and ensures inter-operability. However, devices in a wireless sensor network need to save on every bit transmitted to ensure an acceptable network lifetime. So, how do wireless sensor networks fit in the layered approach of the OSI model?

First of all, it is a well-known fact that wireless sensor networks are application specific networks [50]. In order to improve their level of efficiency, it is important to investigate the possibility of designing communication protocols that are specific designed to the class of applications of wireless sensor networks. In other words, the communication protocols should be able to take advantage of certain inherent properties of the application being considered. This is especially interesting for resource-constrained devices, like the nodes in a wireless sensor network. Of course, this optimization has a negative impact on the flexibility of implementing new algorithms and applications on a real live rollout of a wireless sensor network.

An example of application specific functionality in wireless sensor networks is that sensor samples are (assumed to be) pre-processed before being transmitted. For example, a temperature sensor would obtain several samples, calculate an average result (after correcting for offsets etc.) and send the value. Often this pre-processing is even built into the sensor itself along with calibration mechanisms. The result of pre-processing is that the sensor values that are actually transmitted are compact and small, but represent an accurate measure. Here, it is tacitly assumed that local pre-processing is less energy consuming than transmitting raw sensor samples. In the layered approach, the information added by each layer is larger than the compact sensor value and wastes the effort of compacting the sensor value representation. In other words, the batteries of the sensor nodes would run empty on sending layer information instead of sending sensor samples.

The OSI model also represents functionality that is not used by applications of wireless sensor networks. Typically, all data is routed to a limited set of nodes for the entire lifetime of the node (Section 2.3.7.i). Therefore the concept of *sessions* is not needed and can be left out.

In the network, there will be only a limited number of different packet types and data types. The *presentation layer* is therefore not present in the protocol stack for wireless sensor networks. Its functionality is directly implemented in the other layers and the application itself by specifying the packet formats and types at compile time.

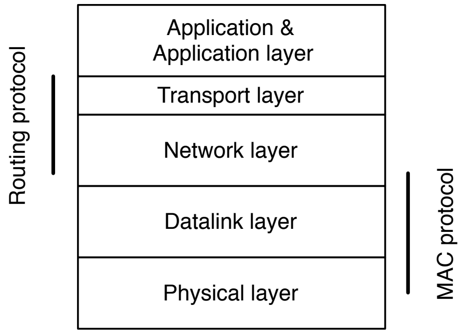


Figure 2.4 — Adjusted protocol stack for WSN requirements [4]

We conclude that the seven layers of the OSI model are —due to the fact that wireless sensor networks are application specific networks– somewhat more compressed in wireless sensor networks than in traditional communication networks [4] (Figure 2.4). Even the borders between the layers are less strict, in order to optimize the protocol stack for memory and energy consumption. The data link layer is closely implemented to the physical layer, whereas the transport and networking layer are commonly implemented by the routing protocol.

2.6 Requirements for WSN MAC protocols

In conclusion, wireless sensor networks are often application specific networks [50]. By this, we mean that depending on the inherent behaviour of the specific application, optimizations can (and should) be carried out. Consequently, wireless sensor networks do not exactly fit in the view of traditional communication systems, known as the OSI model. In other words, networking needs to be designed with energy-efficiency in mind.

Applications envisioned for wireless sensor applications can be characterized as follows:

- **Dense and large scale deployment** — Cheap and tiny sensor nodes allow for fine-grained sensing. As a result, sensors might be deployed densely and in large numbers. Nodes can be added or removed after initial deployment (Chapter 1). Hence, designed MAC protocols must be scalable, allow for unattended operation and be self-configuring.

- **Low overall data rates, but high peak loads** — The wireless sensor nodes pre-process their sensor samples to compact information. This information is streamed to a sink in the network, or the nodes detect out of bound readings themselves and report these to a sink. Although many sensors might detect the same event and report it (resulting in high peak loads), the overall data-rate is low in the network (i.e. in the order of a few bytes per detected event). Data aggregation is a very useful mechanism to reduce payload volume.

- **In-network intelligence** — Several frameworks have been proposed for putting intelligence in the wireless sensor network. For example, sensor nodes can be queried for complex data and do local filtering. A key issue in this is that nodes should be configured with what are normal readings and what readings should trigger. From MAC protocol perspective, this means that loads might vary over time.

- **Long periods of inactivity** — Sensor networks can be inactive for long periods of time. However, devices in the wireless sensor network are assumed to be always on. Many applications do not require real-time reporting of sensor readings, but in some settings latency must be short. Clearly, there is a trade-off between the reactivity of the network (i.e. latency) and its energy consumption.

- **Hierarchical structure** — Sensor readings are often of interest for only certain nodes in the network (we called these nodes gateways or sink nodes). This means that most of the sensor data needs to be forwarded to a small set of nodes in the network.

- **Dynamics** — Node failures (due to harsh environmental conditions or depletion of energy sources of nodes) and communication failures (due to mobility of nodes or interference or even malicious interference) make the topology of the network (very) dynamic. Often, redundancy in the network is assumed in order to be robust against these dynamics. The MAC protocol must be able to handle these dynamics and must be self-healing, when conflicts occur.

The following functionality is assumed by applications:

- **Time synchronization/time stamping** — Sensor readings or context describing events can only be correctly interpreted when the (relative) time is known when the sample was taken or the event occurred.

- **Origin of data** — A data packet is assumed to contain the ID of the generating wireless node or even its position.

- **Source authentication/data integrity/data encryption** — It is a key issue that any data (and its source) can be trusted. It might be required to encrypt all messages, even control messages of the data link layer.

- **Reconfiguring/Reprogramming** — In some applications reconfiguring of node functionality or behaviour is required. Applications can be written as *queries*, making the WSN virtually a distributed database. Changing the application simply comes down to rewriting the queries and distributing them in the network. A pragmatic and inefficient solution to reconfiguring would be *reprogramming*, i.e. updating the program code of the nodes.

Routing protocols are responsible for forwarding messages that are not intended for the node itself. We concluded:

- **Ad hoc and on-demand** — Routing protocols developed for wireless sensor networks characterise themselves by being ad hoc and on-demand, meaning that routes between source and destination nodes are only created when being used. Typically, those routes are enforced when data is transported on them and cease to exist otherwise.

- **Broadcasting** — When a route is unknown, routing protocols try to discover it by broadcasting a (usually short) route request. Geographic routing is an exception to this. The underlying MAC protocol must be able to deal with these short and frequent messages.

- **Often used destination(s)** — Most data packets need to be forwarded to a limited set of nodes (i.e. gateway nodes) to be presented to the user of the network or stored for future reference.

Chapter 3

State of the art in medium access protocols for WSNs

This chapter provides a taxonomy of medium access control and a survey of the state of the art of medium access control protocols for wireless sensor networks. First, we discuss three methods of medium sharing, in time, frequency and code domain. Medium sharing in time domain (i.e. TDMA) is most applicable for wireless sensor networks. We then classify this method further in random, contention-based and schedule-based access, and discuss representative WSN MAC protocols in the three categories of TDMA. We conclude that for energy consumption and delivery ratio arguments, schedule-based access is an attractive choice for wireless sensor networks. Therefore, the following chapters continue with research on this specific method of channel sharing.

3.1 Introduction

There are three basic methods for sharing the wireless transmission resource between multiple users (i.e. nodes in the wireless network). Firstly, the medium can be divided into frequency bands (so called channels), which can carry information without interfering with each other. This method is called *frequency division multiple access* (FDMA). Secondly, the medium can be shared in time between users; *time division multiple access* (TDMA). Each user gets its chunk in time to do its communication. Thirdly, users can use different code bases, which are orthogonal to each other and simultaneous transmissions can be separated at the receiver by correlating the received signal with the specific code of a transmitter. This method is generally known as *code division multiple access* (CDMA). Note that the three medium share methods are orthogonal to each other.

In systems, where users do not continuously transmit or receive, but change their role from transmitting to receiving party and vice versa, a combination of these medium sharing methods is usually used e.g. TDMA and FDMA. However, the use of TDMA is a key issue (and implicit) in such systems. Such is the case with WSNs. Therefore we limit our discussion to TDMA. Little research is done in CDMA or FDMA for WSNs, e.g. by De et al. [19] or Incel et al. [48], respectively.

TDMA can be further classified into (1) random access, (2) schedule-based access and (3) contention-based access. In the following sections, these three sub-classes of TDMA will be discussed shortly.

3.1.1 Random access

In [1] and [2], the first ideas for a random medium access protocol and packet broadcasting are described. The goal of the ALOHA project was to *determine those situations where radio communications are preferable to conventional wire communications* [1]. The ALOHA protocol was developed in the late 1960's at the University of Hawaii in order to facilitate communication between computer terminals. These terminals were located on separate islands of Hawaii. Wireless communication was therefore a convenient manner of communication.

Up until then, communication systems were point-to-point connections, which consisted of continuous transmission of data. A key decision in the project was to transmit user information in a single high speed packet burst and to broadcast the information to multiple receivers [2]. This concept created the need of some form of sharing a common communication channel resource.

The chosen communication channel sharing method is simple. The terminals in the ALOHA project were equipped with a full-duplex transceiver. When a terminal has any data to send, it just sends the information at that moment. When no other node is transmitting a packet at the same time, the transmission succeeds[1]. In the case that another terminal is already using the channel, both transmitting terminals detect this and schedule a retry of transmission after a random period. When two or more transmissions take place simultaneously in the reception area of a receiver, either using the same channel (FDMA), the same time interval (TDMA) or the same code base (CDMA), we speak of a *collision*. In that case, the receiver is not able to distinguish between the transmissions and invalid data is received.

The price to be paid for simplicity in this protocol, is its poor use of the channel capacity; the maximum throughput of the ALOHA protocol is only 18% [1]. However, a modification to the protocol can increase the channel utilization considerably. In slotted ALOHA time is divided into slots, and nodes may only transmit at the beginning of a slot. This organization halves the probability of a collision and raises the channel utilization to around 35% [59].

[1] The terminals are able to detect whether their transmission was successful (i.e. the intended receiver was able to successfully extract the information in the transmission burst). This is key-issue in the protocol.

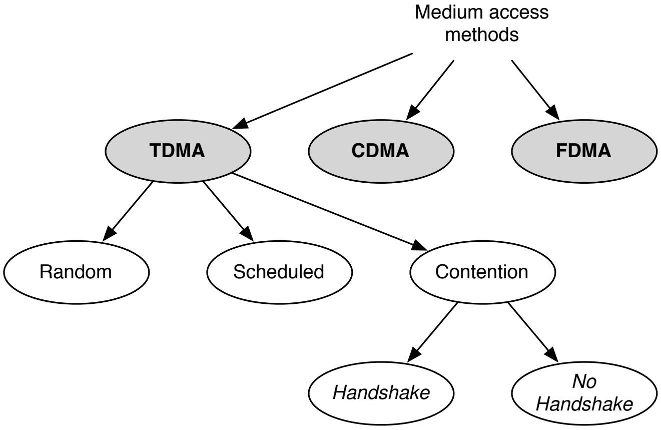


Figure 3.1 — Taxonomy of medium sharing between multiple users

Note that in the ALOHA protocol design, energy usage is of no importance. Also, the protocol functions only when all terminals are in communication reach of each other (i.e. no multi-hop communication).

3.1.2 Schedule-based access

In schedule-based medium access, each node uses its receiving and transmitting functionality according to a schedule. The schedule must be constructed in such a way that conflicts do not exist and each node gets an opportunity to use the medium.

Usually, time is divided into so called *time slots* of fixed length. The schedule determines on time slot granularity what needs to be done. Often, these schedules are repeated after a certain (integer) number of time slots. We denote this period as being a *frame*. A key-issue in schedule-based medium access is time synchronization between the nodes in the network.

In some wireless applications it is important that the channel is shared fairly (i.e. nodes with equal payloads get roughly identical access to the channel). Schedule-based access is naturally a good method for giving each node a fair share of the channel data bandwidth, however *over-provisioning* (i.e. assigning bandwidth to nodes that do not require this bandwidth) and *scalability* (i.e. the addition of nodes) are present dangers.

An example of a scheduled medium access system can be found in the telecommunication standard GSM. When users initiate a call, the base station assigns per frame a specific *time slot* (i.e. a schedule) to the terminal, which is reserved for it during the call.

3.1.3 Contention-based access

Carrier sense multiple access (CSMA) —also called contention-based access– is a special form of single channel medium control, where communication between nodes is not scheduled in a strict sense, but nodes keep track —by listening– of the usage of the wireless channel before they start transmitting. Nodes apply a listen-before-talk strategy. When the medium is in use (i.e. a carrier is sensed), nodes postpone their transmissions until the channel is considered to be free again.

To make sure that a bulk of nodes does not start transmitting simultaneously when a pending transmission ends, randomness is introduced in the start times of transmissions. This is called the (random) back-off period. However, there is a probability that two or more nodes will still try to communicate simultaneously, which results in invalid data at the receiving node.

The above discussed logic is popular and greatly improves the channel utilization compared to pure ALOHA or slotted ALOHA: 80% [56]. It is assumed that each terminal is able to correctly assess the status of the channel. The CSMA protocols will thus only be able to reach the 80% channel utilization in a one-hop network.

There are two variants of CSMA:

- **Collision avoidance** — A collision of data is avoided by first notifying other nodes that a data transmission is to start. This method is known as CSMA/CA. There are different flavours of this method concerning the handshake used. CSMA/CA is suitable to use in multi-hop wireless network and we discuss handshake mechanisms in Section 3.1.3.i.

- **Collision detection** — This variant is known as CSMA/CD. Nodes detect whether they cause a collision and stop their transmission, if so. This potentially increases the channel utilization.

The CSMA/CD method is not suitable for wireless (sensor) networks for the following reasons: (1) to detect a collision, nodes would require additional hardware (at least a full-duplex transceiver), which is often not available, as we argue in Section A.2.2, and (2) power of the RF signal attenuates with distance (Appendix C). When the distance between two transmitting nodes is large, both nodes would receive little interference of each other and would conclude that they do not cause collision. However, a receiving node, placed between the two transmitters, might experience collision. Hence, collisions occur at the *receiving node.*

3.1.3.i Handshaking mechanisms for CSMA/CA

In a multi-hop network, the above discussed CSMA protocol suffers from the *hidden terminal problem.* When two transmitters are out of range of each other, they are not able to sense each other's transmission and their messages might collide at a receiver, which is in range of both transmitters. This problem is first described by Karn in [55].

In his paper, Karn proposes a handshaking mechanism that solves the hidden terminal problem. This protocol is known as the MACA protocol [55] and requires a transmitting terminal A first to announce its transmission to the receiver B with

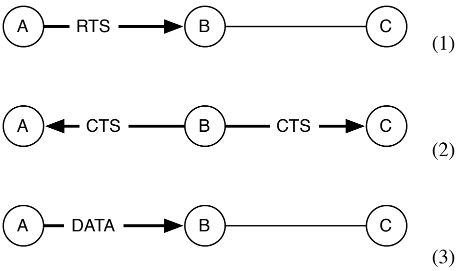


Figure 3.2 — Handshaking mechanism in the MACA protocol [55]. In step (1), A reserves the medium, C does not hear this reservation, however in (2) B acknowledges the reservation and C overhears this and postpones it transmissions. Finally, A transmits its data for B (3)

a *request to send* (RTS) message. The receiver B should then answer with a *clear to send* (CTS) message. This CTS message can be overheard by a transmitter C assessing the channel and thus C knows that the channel is busy and schedules a re-attempt for transmitting its message. The transmitter A sends its data to receiver B, when it has successfully received the CTS message of B. Figure 3.2 clarifies the handshaking mechanism.

In the case that two or more messages collide, nodes conclude that the data transfer failed. If a node did have any data to send, it will re-attempt after a random time interval. The RTS and CTS messages are typically very short messages and thus they do not have a great negative impact on the channel utilization.

An extension to the MACA protocol [55] is the MACAW [12] protocol; after the data burst of A, receiver B responds with an additional message, the *acknowledgement* (ACK) message. The message is added to be able to cope with the unreliable wireless communication channel. Transmitter A knows when it receives an ACK message, that its data message is correctly received. When A does not receive an ACK message, it re-schedules a new attempt to resend the message after a random time interval.

The above discussed four-stage handshaking mechanism is the basis for many medium access protocols in wireless applications.

CSMA/CA suffers from the *exposed terminal problem*. The protocol needs the data transmitting node to be able to receive messages —namely CTS and ACK messages– from the data receiving node. When there is a situation where data transmitting nodes are within transmission range, but they are out of range of the others intended receiver, the transmitters block each other and the transmissions can take place one-by-one. The total data capacity of the network is hence reduced by this effect. This problem is not yet solved.

3.2 Overview

Table 3.1 provides an overview of the MAC protocols discussed in this chapter. Despite the number of protocols proposed so far, there is still no clear indication of converging MAC mechanisms for wireless sensor networks according to [59]. Since different applications might require optimizing different parameters, there is most likely not a single solution, which fits all types of wireless sensor applications.

According to Langendoen et al.'s survey [59], WSN MAC protocols can be classified according to (1) the number of channels used, (2) how the intended receiver of a message is notified, and (3) how medium access is temporally organized.

1. **Frequency channel** — In terms of channels, most WSN MAC protocols use only a single channel, e.g. SMAC [111, 112], DMAC [65] and BMAC [83]. Exceptions to this are the standardized MAC protocols, like IEEE 802.15.4 [114].

2. **Message notification** — In terms of message notification, in some protocols, a communication scheduling algorithm determines when a node is listening for messages to minimize energy consumed by idle listening.

   In other protocols, a node has to determine on its own when to listen for messages. To reduce energy spent on idle listening, they typically employ some form of sleep-listen schedule. These protocols are more lightweight and hence more viable for WSNs. SMAC is an example.

3. **Temporal organization** — In terms of temporal organization of medium access protocols, e.g. f-MAC, TRAMA and TMAC belong to different categories. f-MAC uses random access to the communication medium. TRAMA divides time into time slots, and each node is allocated a time slot(s) to send packets. In the TMAC protocol, nodes contend for the medium.

We add to the classification of WSN MAC protocols the following aspects:

4. **Hierarchy** — In some MAC protocols, a hierarchy is assumed between the nodes e.g. in the IEEE 802.15.4 MAC protocol, there is a clear difference between the coordinators and other nodes. In —for example– SMAC such hierarchical structure is not assumed. The DMAC protocol is optimized for data, which has to travel to one central point. In that respect, the DMAC protocol assumes hierarchy.

Table 3.1 provides an overview of the discussed MAC protocols.

3.3 The wireless LAN standard IEEE 802.11x

The IEEE standard 802.11 [49] has been designed for (high data rate) wireless local area networks and provides similar functionality as *wired* local area networks by delivering internet connectivity wirelessly (i.e. the last few meters) to the user. This standard is widely adopted and integrated in various devices, most saliently in laptops and PDA's. In those often resourceful devices, performance of networking is a

Table 3.1 — Overview of discussed MAC protocols

Protocol	Section	Standard	Freq. channels	Hierarchy	Year
Random access					
f-MAC [91]	3.7		single		2006
GeRAF [117]	3.7		single		2003
Contention-based access					
IEEE 802.11 [49]	3.3	✓	multi		1999
IEEE 802.15.4 [114]	3.4	✓	multi	✓	2003
SMAC [111]	3.5		single		2002
TMAC [18]	3.5		single		2003
Schedule-based access					
μMAC [9]	3.6		single		2005
DMAC [65]	3.6		single	✓	2004
MERLIN [94]	3.6		single	✓	2004
MMF-TDMA [98]	3.6		single	✓	2004
TRAMA [85]	3.6		single		2003
Preamble sampling					
BMAC [83]	3.8		single		2004
CSMA-MPS [69]	3.8		single		2004
LPL [46]	3.8		single		2002
WiseMAC [28]	3.8		single		2003

key-issue and fair access to the medium is a prerequisite. The standard specifies both medium access layer and physical layer, however, our focus is on the MAC layer.

The standard supports two modes: (1) infrastructure mode and (2) distributed coordination mode. When a base station is present, which typically has a connection to high speed (wired) Ethernet infrastructure, it takes the task of coordinating traffic. In this mode, the base station periodically transmits a beacon message after which it polls associated terminals if they have data to send. This guarantees contention-free access to the medium. After the contention-free period, a contention period begins, in which —for example– new devices in the network can announce themselves.

More interesting in perspective of WSNs is the second mode in the IEEE 802.11 standard: the *distributed coordinator function* (DCF), in which the coordination of the channel access is achieved by CSMA/CA and its four stage handshaking mechanism as we discussed in Section 3.1.3.i. In this mode, peer-to-peer ad-hoc communication is provided and multi-hop communication is —in principle– possible, however, it is not part of the standard nor is the required routing functionality provided by typical higher layer protocols in (W)LANs, such as TCP/IP.

In the next sections, the handshaking, back-off and the virtual carrier sensing mechanisms are discussed in detail.

3.3.1 Virtual carrier sensing

When a packet (also called *frame* in IEEE 802.11 standard, not to be confused with the term frame we use for schedule-based MAC protocols) has to be transmitted to another device, the transmitting terminal makes sure that the medium is not in use. The carrier sensing function that is required for determining whether the wireless channel is busy or not, is performed both through *physical* and *virtual* means.

The IEEE 802.11 standard introduces the concept of network allocation vector (NAV). The NAV maintains a prediction of future usage of the wireless medium, based upon duration information of the complete transaction, including all stages of the handshaking mechanism. This length information is announced in all four types of handshake messages. Therefore, even if a node would only overhear a CTS or DATA message, it still is able to mark the channel busy until the end of the ACK message. Figure 3.3 shows typical use of the virtual carrier sensing during data transfer of other nodes.

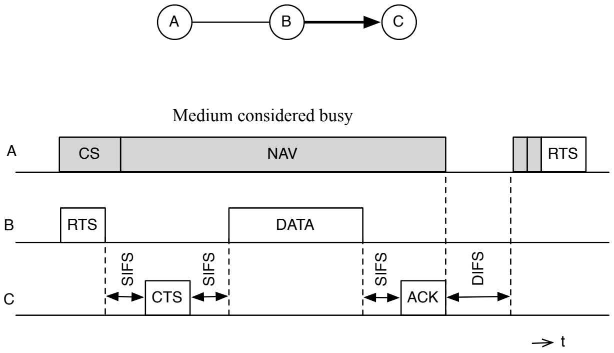


Figure 3.3 — Physical carrier sensing (CS) and virtual carrier sensing (NAV) in IEEE 802.11

Obviously, the augmentation of messages with length information of the entire handshaking procedure introduces overhead. In the IEEE 802.11 standard, duration is notated in (integer) microseconds, increasing message sizes with 16 bits.

3.3.2 Back-off mechanism in IEEE 802.11 MAC layer

Consider the case that a node has a frame to send and assesses the state of the medium. Whenever the node perceives a medium busy condition (i.e. a carrier is detected by the physical layer or the NAV indicates that the medium is in use), it observes a random back-off interval before a medium access reattempt. In this section, we briefly describe the back-off procedure in the IEEE 802.11 MAC layer.

The IEEE 802.11 standard specifies the back-off slots uniformly from the set $\{0, 1, \ldots, W - 1\}$, where W is the (fixed) size contention window. Every back-off slot (i.e. a fixed length time interval) in which the node does not observe the channel busy, it decreases it back-off counter. When the channel is either physically or virtually considered to be busy, the back-off counter is halted. When the counter reaches zero, the node begins the transmission of the frame by initiating the handshaking mechanism.

With each failing channel access attempt (for example no CTS reply was received after a RTS), the size of the contention window W is doubled, until it reaches a predefined maximum. A successful medium access resets the contention window to its predefined minimum value.

For the back-off mechanism to work correctly, it is required that the IEEE 802.11 physical layer is in receive mode whenever the channel is considered to be idle. Note that this has its effect on energy-consumption of the devices in the network.

It is interesting to note that the back-off mechanism of IEEE 802.11 favours the last winning node in successive access to the channel, simply because the contention windows grow exponentially for the other nodes, while the window remains small for the last winning node.

3.3.3 IEEE 802.11 in perspective of WSNs

The IEEE 802.11 standard is built upon the handshaking mechanisms proposed by Bharghavan et al. in [12]. The standard is very well suited for ad hoc computer communication.

The standard is not specially designed with resource constrained devices, such as wireless sensor nodes, in mind. The standard even requires the full functionality to be implemented in all devices, resulting certainly in homogeneous resources at the lower layers in the protocol stack. The standard is, however, sometimes considered for wireless sensor networks (e.g. [77]) and it is an inspiration for well-known MAC protocols for WSNs, like SMAC and TMAC.

3.4 ZigBee and IEEE 802.15.4

ZigBee and IEEE standard 802.15.4 have been designed for low-rate wireless personal area networks. The two standards are related as follows. The IEEE 802.15.4 standard specifies the physical and medium access control layer. The ZigBee Alliance relies on the IEEE 802.15.4 standard and defines higher layers on top of this standard.

Together, the standards attempt to provide an ultra-low complexity solution for long-lasting, cost effective wireless networking. The concept of heterogeneous resourced devices, already supported from MAC layer level, contributes to this. This means that not all devices in the WPAN network need to be able to support the full functionality as specified in the standards, resulting in a mix of full-function devices (FFD) and cheaper reduced functionality devices (RFD). This is especially useful for simple applications, like home lighting control, where the device holding the switch would be a cheap RFD and the control device at the lamp would be a FFD. The light switch can then be implemented using a minimum of resources, such as memory and processing power.

Devices in low-rate wireless personal area networks have many aspects that compare to devices in wireless sensor networks: they are typically battery operated (thus have limited energy available, while the devices are required to function as long as possible), they must operate in an ad hoc fashion (no fixed infrastructure is present to facilitate communication) and the data rate is expected to be low. In addition, the ZigBee and IEEE 802.15.4 standards support multi-hop networking. Therefore, it is interesting to discuss these standards as being potential candidates for adoption in wireless sensor networks and to compare them to proprietary protocols. Our focus will be on the energy-efficiency of the IEEE 802.15.4 MAC layer.

3.4.1 Assumptions of device capabilities

The ZigBee and IEEE 802.15.4 standards assume three types of devices in the wireless network:

- **Coordinators** — Full function devices that provide synchronization and coordination services by transmitting (periodically) beacon messages. Any FFD must be able to perform these tasks. In a multi-hop network, coordinator devices typically take care of routing of messages. The coordinator devices are required to have more processing power and memory available than the other devices in the network.

- **Personal area network (PAN) coordinator** — If a coordinator is the principal controller of a PAN, it is called a PAN coordinator (an IEEE 802.15.4 network has exactly one PAN coordinator). The principal coordinator chooses a personal area network identifier that is used throughout the network. ZigBee only considers intra PAN communication, but in principle the IEEE 802.15.4 standard allows for inter PAN communication as well.

- **End-device** — A device (FFD or RFD) that does not act as coordinator. These devices do not participate in establishing and maintaining routes, etc. In fact, they use resources of the coordinator with which they are associated. Since all data transfers are taking place via a coordinator, the end-devices can be in energy-conserving mode for considerable periods.

3.4.2 Supported network formations

There are two types of network topologies that are supported by the IEEE 802.15.4 standard: (1) peer-to-peer topology and (2) star topology.

In peer-to-peer network formations, any device is allowed to communicate with any other device, without interaction of a (PAN) coordinator. This network formation can be used for multi-hop wireless networks, where nodes assist each other in forwarding messages to their destination. However, the IEEE 802.15.4 standard requires devices either to receive constantly, or to synchronize with each other. In the former case, multi-hop communication can obviously not be long-lived, due to the energy spent during constant activity of the transceiver. Methods of multi-hop synchronization in peer-to-peer networks are outside the scope of the IEEE 802.15.4 standard. As a result, we conclude that an energy-efficient peer-to-peer multi-hop network cannot be created within the ZigBee and IEEE 802.15.4 standards. Therefore, we will not consider the peer-to-peer network formation.

In the case of a star network formation, all communication flows via a coordinator device, even when two devices are directly within each others communication range. A coordinator device controls all data traffic of the devices that are associated with it and this allows other devices to be in idle mode for long times. Thus, the resulting network can be long-lived. In the IEEE 802.15.4 standard, this mode of operation of the network is called beacon-oriented communication.

The ZigBee standard extends the two supported network formations with a tree topology by linking groups of star network formations on networking layer level. The resulting network allows energy-efficient communication over multiple hops. However, it needs careful deployment of the devices, especially the coordinators, and aligning of the IEEE 802.15.4 MAC scheduling. The topic of multi-hop communication in ZigBee is discussed in Section 3.4.4.

3.4.3 IEEE 802.15.4 Medium access control layer

3.4.3.i Superframes and time slots

In the beacon-oriented IEEE 802.15.4 networks we consider, end-devices are synchronized with a coordinator. This coordinator device periodically transmits a *beacon message*, which not only allows end-devices to detect the coordinator, but also gives the coordinator the opportunity to advertise pending data transmissions. The beacon message also includes important timing information of the *superframe* structure that is to be used by the end-devices. In a multi-hop network, these timing parameters are determined by the PAN coordinator. The other coordinators simply propagate an identical superframe structure.

A superframe (Figure 3.4) is divided into an *active period*, where the coordinator accepts communication using CSMA/CA mechanisms, and an *inactive* period, in which the coordinator switches to low-power (sleep) mode. Outside the active period, all end-devices are assumed to pause all pending CSMA/CA back-off counters. However, their state is preserved to continue in the next superframe. Consequently, the end-devices can be switched to low-power mode in the inactive period.

The active period of the superframe is divided in 16 time slots of equal length. At the beginning of time slot zero, the coordinator carefully times the transmission of its beacon message.

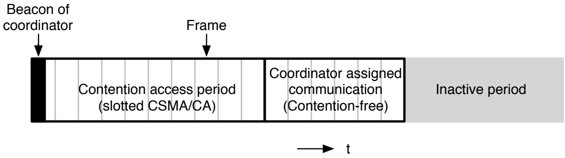

Beacon of coordinator
Frame
Contention access period (slotted CSMA/CA)
Coordinator assigned communication (Contention-free)
Inactive period
t

Figure 3.4 — IEEE 802.15.4 MAC superframe structure

The time that is left in the superframe is divided into two parts:

- **Contention access period** — In this period, devices need to compete with others (using the slotted CSMA/CA mechanism) before they can communicate with the coordinator. The beacon message of the coordinator specifies at which time slot the contention access period ends. Note that the back-off periods used in the slotted CSMA/CA mechanism are not related to the previously introduced time slots. Time slots do not have a fixed length. Their length is implementation specific (i.e. specified by the PAN coordinator), however, the length of the back-off period, the *CSMA/CA slot*, is fixed.

- **Contention-free period** — In this period, the coordinator can assign time slots to devices. In these time slots, a device can communicate (guaranteed) with the coordinator. The IEEE 802.15.4 standard specifies that these guaranteed time slots (GTS) can only be assigned by the PAN coordinator. The PAN coordinator is allowed to assign up to 7 GTSs in total, each possibly consisting of more than one time slot. A general rule in allocating GTSs is that enough slots should remain available in the contention period, for example to allow new devices to be added to the network. All devices should take care that their communication has finished before the GTS or the active period ends.

How does the data exchange between devices work in practice? We distinguish here three possible transactions on MAC level, namely (1) data from coordinator to device, (2) data from device to coordinator and (3) exchange of MAC commands (which will not be discussed). However, before any data transfer can take place, the devices must be synchronized to the beacon message of the coordinator. In the following sections, we assume that synchronization has already happened.

3.4.3.ii Data transfer to a coordinator

A device wishing to transfer data to a coordinator has either been granted a time period in the contention free period, or it has to compete with other devices for the medium during the contention period. In the former case, the device uses the medium when the guaranteed time slot starts. In the latter case, the device uses slotted CSMA/CA to access the channel. Every device makes sure that the wireless medium is not used by any other device before commencing a transmission. The CSMA/CA procedure will be described in Section 3.4.3.iv.

When the coordinator receives the data correctly, i.e. no collision occurred, the checksum matched the one in the transmitted data frame, and all addressing was correct, it will transmit an acknowledgement message if requested to do so. For acknowledgements the CSMA/CA procedure is not used, but these messages are directly transmitted after the data frame.

This method of transferring data to the coordinator implies that the coordinator must be in receiving mode during the contention period of the IEEE 802.15.4 superframe. However, for battery operated coordinators, this can be very energy-consuming. To extend the lifetime of such coordinators, the IEEE 802.15.4 standard introduces a battery lifetime extension mode that limits the interval in which the coordinator listens to the channel after it has transmitted a beacon message. The standard specifies that in this energy-conserving mode, the coordinator stops listening to the channel after 6 back-off periods. However, it will still service the GTSs as specified in the beacon message. Again, the beacon message is used to inform the end-devices of the reduced activity of the coordinator. In principle, the coordinator can switch between modes on a per frame basis. The end-devices adapt their CSMA/CA back-off procedure accordingly.

3.4.3.iii Data transfer from a coordinator

Data transfer from coordinator to an end-device occurs in an indirect manner: the coordinator notifies end-devices in its beacon message that it has data pending. To actually transfer the data, the end-devices have to request for the data by sending a request message (using the CSMA/CA procedure). If successful, the request is acknowledged by the coordinator and the data is transmitted using the slotted CSMA/CA mechanism. The data transfer is completed when the coordinator receives an acknowledgement message from the end-device. Note that the end-device requesting for data remains in receive mode until the data is received.

3.4.3.iv IEEE 802.15.4 CSMA/CA

The IEEE 802.15.4 standard uses (slotted) carrier sense multiple access with collision avoidance to access the wireless medium. This means that all devices first assess the state of the wireless channel (i.e. is there a transmission going on?) before using the medium. The algorithm operates as follows. When a device has to send a message, it first waits until the beginning of the next back-off slot. Then it schedules a random back-off period. After this back-off period the node wakes and carries out channel assessment in two successive back-off slots. If the channel was found idle in the two slots, the device starts transmitting in the next back-off slot. If not, again a back-off procedure is started. However, the device updates the parameters of the algorithm. After four attempts, the device gives up and cancels the message it wanted to send.

3.4.4 Multi-hop communication in ZigBee

A multi-hop network can be constructed by linking groups of star formations into a tree structure. This is done at the networking layer level in the ZigBee. This layer adds the following information to the beacon message: (1) device depth in the tree structure, (2) timing offset of its beacon message, relative to its parent's beacon message.

The first variable allows nodes to choose a parent node (i.e. a node closer to the PAN coordinator). Effectively, this builds the tree structure. The second variable allows new coordinator devices in the network to choose a schedule. A precise mechanism for choosing a schedule is not provided in the ZigBee specification, however, hints are given on how the communication should be organized.

In a multi-hop network setting (the tree-structure), a coordinator will use a part of its inactive period to communicate with a coordinator closer to the PAN coordinator. In fact, it will behave as an end-device during the active period of its parent, listen to its parent's beacon message, forward and accept data for its children —when necessary– using the CSMA/CA mechanisms.

Obviously, active parts in communication between parent and siblings should not overlap. The same holds for the active periods of other nodes in transmission range. Therefore, all coordinator devices discover all other beacon transmitting devices in their vicinity and choose a non-overlapping schedule. In addition, a child node takes care that it is not active during the schedule of its parent's parent (otherwise the parent is not able to communicate successfully with its parent). To establish this, the parent transmits in its beacon message the timing offset it has, in respect to its parent's schedule. Any algorithm for finding a schedule that guarantees the above prerequisites may be used in ZigBee.

3.4.5 IEEE 802.15.4 and ZigBee in perspective of WSNs

In the previous sections, we discussed IEEE 802.15.4 and ZigBee in relation to multi-hop wireless sensor networks. We concluded that for these types of networks, the IEEE 802.15.4 MAC layer is only attractive to use in the beacon-oriented mode. In its other mode, the IEEE 802.15.4 MAC layer requires nodes to receive constantly —hence not being long lived– or out-of-protocol synchronization has to be applied.

In the beacon-oriented mode, the protocol assumes a tree-based hierarchy in the network, in which energy consumption is especially optimized for the end-devices. Coordinators need to fulfil two roles, namely the coordinator function to provide communication services to its end-devices and it acts like an end-device to communicate with a device one hop closer to the PAN coordinator. End-devices can not directly communicate with each other. This makes a ZigBee network not suited for sensor to sensor communication (Section 2.3.7.i) e.g. complicating the task of node localization.

Another concern we have for the use of IEEE 802.15.4 and ZigBee in WSNs, is related to scalability. Due to the hierarchy in the ZigBee network, the PAN coordinator controls many network functions (e.g. binding of nodes to the network and routing). Yet, only one PAN coordinator is allowed in the network to carry out these tasks.

3.5 Contention-based access

In this section, a brief overview of contention-based MAC protocols for WSNs is presented. A key feature of these protocols is the RTS/CTS/ACK handshaking mechanism to overcome the hidden terminal problem. In general, the discussed protocols reduce the energy consumption in nodes by introducing sleep periods. To be able to maintain communication between nodes, some form of synchronization is required. Most authors propose additional message exchange outside the regular handshaking. Those messages indicate —for example– how long the node will still be listening to the wireless channel.

Synchronization is not required to be very precise and during their sleep period, nodes are often allowed to access the wireless medium. The robustness of the listen-before-talk and handshaking mechanism makes these protocols conveniently flexible.

3.5.1 Sensor-MAC Protocol

The SMAC protocol, presented by Ye et al. in 2002 [111, 112], recognizes two phases in transceiver usage of network nodes: an receive/transmit period (also called the listen period) and a sleep period. In the sleep period, the nodes turn off their power-consuming transceiver and application packets are backlogged. In this period, the energy consumption of the node's transceiver is minimal and hence the node lifetime is extended beyond the lifetime of a node that is always listening.

After the sleep period, the nodes wake-up and listen for communication that is addressed to them, or they initiate communication themselves. This implies that the sleep and listen periods should be (locally) synchronized between nodes. Since the protocol is CSMA(/CA)-based in the listen period, synchronization does not have to be very strict and nodes can also use their sleep period for communication, if needed.

The wireless LAN MAC protocol (Section 3.3) stand model for the SMAC protocol. Many of its features —like NAV, handshaking mechanism and interframe spacing– are included in the SMAC protocol, yet slightly adapted (message sizes, etc) for hardware limitations. We focus our description of SMAC on its synchronization mechanism, which is not present in the IEEE 802.11 standard.

When nodes are turned on, their clocks are unsynchronized, i.e. the nodes are required to discover (local) schedules. For this purpose, the nodes do not yet adapt a sleep/listen pattern, but rather maintain their listen state for several sleep/listen periods. By these means, any transmitted message is detected.

To announce schedule information, nodes exchange so called SYNC messages. The schedule information is transmitted without the typical RTS/CTS handshaking of contention-based MACs. To minimize collision of these messages, Ye et al. divide the listen period of SMAC into two parts (Figure 3.5): (1) a part is especially reserved for the exchange of SYNC messages and (2) a data communication part using the well-known handshaking mechanism.

The SYNC message information content holds the ID of the sender node and the remaining duration of its listen period. Note that the SMAC protocol assumes that the duration of the listen and sleep periods is known by all nodes.

To assure the ad hoc property of the SMAC protocol, the following steps are taken concerning listen/sleep schedules:

- **Synchronization phase** — A new node in the network listens for a network wide predefined amount of time. When it does not receive a schedule (denoted by a SYNC message) from another node, it randomly chooses a time to enter the sleep phase and transmits this information (a relative time) periodically in a SYNC packet to its (potential) neighbours. The node now defines the schedule in the network and is called the synchronizer. Other nodes will synchronize to the schedule of the synchronizer. Potentially, many unsynchronized and overlapping schedules might be created within the same WSN. Note that the predefined listen period is required to be at least the interval between two SYNC messages.

- **Joining an existing schedule** — If a node receives a SYNC message during the synchronization phase, it adjusts its schedule to the information in the SYNC message. The node follows the sleep/listen schedule in the network. A random time interval is waited before the follower transmits a SYNC message, in order to prevent collisions between SYNC packets when multiple nodes are triggered by the same SYNC message.

- **Multiple schedules detected** — If a node has already chosen a schedule and it becomes aware that one of its neighbours is following a different schedule, it keeps its own schedule and it also wakes accordingly to the schedule of the other node. However, the node will not transmit its new listen and sleep pattern

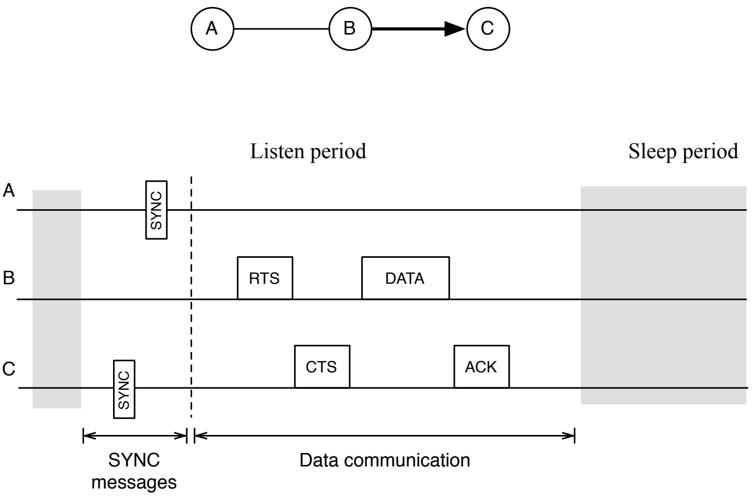


Figure 3.5 — Scheduling in the SMAC protocol for WSNs [111, 112]

in SYNC messages. Instead, it transmits its own chosen schedule to prevent a propagation of rescheduling in the entire network.

Nodes between regions with different schedules have less sleep time compared to others and therefore their energy consumption will be higher. All nodes maintain a table with the schedules of their neighbours. The SMAC protocol includes mechanisms to reduce the number of schedules by conflating almost overlapping schedules and deleting schedules used by single nodes.

After the synchronization period, nodes proceed with carrying out their schedule. At the beginning of their primary listen period, nodes listen for SYNC messages to keep a view of what nodes are in the same schedule. From time to time a node transmits a SYNC message during this period.

In the next phase of the listen period, communication follows the handshaking rules as in the IEEE 802.11 standard (Section 3.3). To reduce multi-hop delay, nodes apply *adaptive listening*, i.e. when a CTS is received at the end of the listen period, nodes remain in listen mode just long enough to await a potential RTS message.

3.5.2 TMAC, a variant of SMAC

The TMAC protocol [18] is a variant of the SMAC protocol. It attempts to reduce the total listen time —and thus the energy-consumption– of nodes in the listen period by preventing nodes from waking after a transmission has taken place. Instead, it reserves space for a future request to send (FRTS) message between CTS and data messages. In this way, nodes can be asked to be awake after an ongoing transmission because data is going to be sent to them. The other nodes can sleep, until their schedule tells them that they are again entering their listen period.

3.6 Schedule-based access

In this section, a short overview of schedule-based MAC protocols for wireless sensor networks is presented. These protocols are characterised by the fact that they establish a schedule to carry out communication.

Schedule-based MAC protocols generally perform well on energy-efficiency and data throughput [95]. A general drawback of schedule-based medium access is the fact that synchronization is required between nodes. The better the synchronization, the shorter the guard times (i.e. receive time before the actual start of a message to overcome differences in timing) can be and the shorter the idle listening. This is how more energy can be saved at MAC level.

3.6.1 MMF-TDMA

In [98], an algorithm is presented for assigning time slots in a multi-hop network. The work of Sridharan et al. [98] uses a linear programming min-max fairness approach to assign bandwidth to nodes. The goal is to assign bandwidth to nodes which require it. The authors show that all data sources in the network get a fair share of the data throughput.

The so called MMF-TDMA algorithm runs distributed, yet presumes a tree-like network organization in which parent nodes coordinate the assigning process of their siblings. Once established, the schedule remains fixed. The authors argue that the performance of their Greedy-algorithm for assigning time slots does not necessarily result in an optimal solution, but in their simulations the results were empirically found to be optimal. The algorithm ensures that concurrent transmissions can occur only at three hops or more.

When a node is added to the network, the algorithm might need to be rerun in order to assure that the new node is assigned bandwidth. This makes the protocol less suitable for mobile node environments and iterative deployment. Due to hierarchical coordination of the time slot/bandwidth assigning, the setup time of a network is dependant on the number of nodes, making the MMF-TDMA protocol not scalable.

3.6.2 DMAC

In [65, 66] a different method is chosen for assigning time intervals to sensor nodes. The main goal in the data-gathering MAC (DMAC) [65] is to reduce latency of data that is designated for the central point in the network and therefore the assignment of the time intervals is dependant on the hop-distance of the node to the central point. Within its time interval, a node is assumed to transmit data to a node that is one hop closer to the central point and one time interval before its own, a node is expected to listen to the medium and to accept data transfers. The protocol uses in total four time slots to prevent collisions. Concurrent transmissions are (exactly) four hops spaced.

The DMAC protocol schedules the medium in a very granular manner; within a time interval, messages of nodes located at equal hop-distance to the central point can still collide. This makes the performance of the protocol questionable in case of for example event detection, where potentially many nodes register the same event and report it.

A variant of this protocol is the Merlin protocol [94]. Besides targeting latency towards the gateway node, this protocol also optimizes latency from gateway to nodes, which is e.g. used in configuring the WSN. The Merlin protocol therefore extends the four time slot scheduling of DMAC by adding a similar structure in a opposite direction in the DMAC routing tree.

3.6.3 TRAMA

TRAMA [85] is a complex schedule-based protocol, where nodes compete in establishing a schedule.

The first phase of the protocol is a *random-access phase* in which nodes discover their neighbours and establish the schedule. Nodes locally compute who is the absolute winner among their two hop neighbours in certain time slots by calculating a priority function i.e. $priority = h(Node\ ID \otimes slot\ number)$, with $h(\ldots)$ a network-wide known hash function. Based upon the results of this priority function, time slots are reserved to the winner. The protocol ensures a distance of three hops or more between concurrent transmissions. New nodes can enter the network only during the random-access phase.

The second phase of the protocol is a *contention-free phase*. In this phase, time is divided in small time slots and the schedule is fixed. When the schedule is completed, the nodes fall back to the random-access phase. Note that TRAMA does not use the notion of frames.

Energy is conserved by letting nodes —that are not involved in communication in the current slot– sleep. Therefore its energy performance is dependant on the actual data load in the network.

The protocol assumes that nodes are synchronized and have knowledge about the start of the random-access phase and contention-free phase. The protocol is quite complex in terms of computational power required to establish the schedules.

Barroso et al. present μ-MAC [9]. μ-MAC assumes a single channel organized in a (1) contention period and (2) a contention-free period. The contention period is used to build the network topology and establish the schedule of the contention-free period. The contention-free period is used to carry out the actual data transfer of sensor readings. Other forms of communication take place in special reserved time slots for broadcasting (not contention-free) communication. The protocol uses a similar architecture as TRAMA [85], however the protocol uses a different time slot reservation mechanism. Unfortunately, Barroso does not compare μ-MAC with TRAMA in [9].

3.7 Random-based access

In this section, we discuss one representative random access MAC protocols for WSNs. This sub-class of MAC protocols is characterised by the random nature of the transceiver mode usage. Typically, no handshaking mechanism between nodes is applied and no particular node is addressed.

Roedig et al. present the random access protocol f-MAC [91]. The basic principle of the f-MAC protocol is the following. Messages are transmitted in framelets with fixed (known) length. Each node transmits the framelets exactly as often as there are nodes within transmission range. The framlets are spaced with a unique time interval, chosen such that the least common multiplier of the (discrete) periods is smaller than the total framelet burst length. In that case, the authors show that at least one framelet is transmitted collision-free [91].

f-MAC is only useful in sparsely populated networks [91]. In addition, the MAC protocol requires nodes to be in receive mode when not transmitting. Hence, it does not establish a long-lived WSN.

3.8 Preamble sampling

Idle listening is one of the main reasons for energy-wastage in WSNs [111]. A simple solution to this problem is to let nodes receive and sleep according to some duty cycle. In order to minimize the energy-wastage, a node would be in receiving mode as short as possible. As a consequence, a transmitting node carries the responsibility of making sure that nodes are in receive mode when it transmits.

Typically, this is achieved by transmitting a (long) preamble before the actual information content. The duration of the preamble is at least long enough to be detected by nodes, which sleep and wake only very briefly to sample the wireless channel for the preamble (Figure 3.6). Once a preamble has been detected, the receiving nodes do not switch back to sleep state, but continue receiving until the actual information has been received.

Actually, the idea of preamble sampling was to our knowledge first proposed by El-Hoiydi [27] and its general ideas are —often in adapted form– used in *low power listening* (LPL) [46], BMAC [83], WiseMAC [28], *CSMA with minimum preamble sampling* (CSMA-MPS) [69] and STEM [96]. We discuss BMAC in more detail, since it is often considered as the standard WSN MAC in the TinyOS community.

BMAC (and other preamble sampling techniques) is actually a physical layer tweak. It does not stipulate how the communication medium is shared between nodes. However, it is reasonable to assume RTS/CTS signalling or listen-before-talk is being used. When RTS/CTS signalling is used, the sender sends an RTS packet with an extended preamble. Upon detecting this long preamble, the receiver snaps out of the LPL mode, and replies with a CTS packet that has a normal preamble, since the

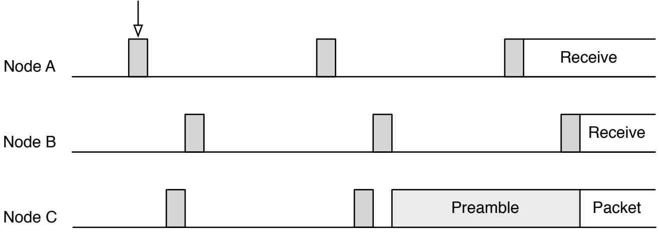


Figure 3.6 — Typical preamble sampling scheme

sender is already listening and waiting. The ensuing data packet and acknowledgement packet exchanged between the sender and receiver are all transmitted with a normal preamble. After sending the acknowledgement packet, the receiver returns to the preamble sampling mode.

A similar methodology to reduce energy consumption due to idle listening can be found in literature: *wake-up radio*, i.e. a ultra low-power radio with sole purpose to detect a RF carrier at low energy costs e.g. [73, 110]. Since preamble sampling techniques in general do not specify how the medium is to be shared between nodes, these techniques are outside the focus of this work.

This concludes our summary of the most widely used MAC protocols in WSNs.

3.9 Random, scheduled or contention, what fits WSNs best?

The following goals are often strived for in MAC protocols for wireless sensor networks: (1) energy-efficiency —to ensure an acceptable node lifetime or cheap batteries/power sources, (2) self-organization of multi-hop communication, (3) minimum delay (i.e. latency) between the detection of an event and the reporting to the WSN user and maximum delivery ratio, (4) scalability, and (5) resource consumption (e.g. computational complexity, memory footprint etc). In light of this, we evaluate the three sub-classes of TDMA.

3.9.1 Energy-efficiency

The following waste energy in MAC protocols [111]:

1. **Idle listening/channel assessment** — Listening to the channel, while there is no transmission to be received by the node, is called *idle listening*. This is the major cause of energy-waste in MAC protocols.

2. **Overhearing** — The node receives and decodes messages for which it is not the intended receiver. Overhearing can be reduced by deciding as soon as possible that the message is not intended for the node and then turning off the transceiver, only to wake again when the message has cleared the medium. The IEEE 802.11 MAC protocol makes provisions for this [49].

3. **Collisions** — Two or more messages are transmitted simultaneously or partly overlap in such a way that the receiver(s) cannot successfully decode any of the messages.

Random-based medium access naturally applies idle listening —as we have seen in f-MAC [91]– because messages can be transmitted at any time. For the same reason, overhearing and collisions are to be expected in this medium access sub-class.

Contention-based medium access relies on the assessment of the wireless channel to prevent collisions. When handshaking is used, RTS messages can be sent at any time. Therefore, nodes are required to be in receive mode continuously. Hence, energy is wasted by idle listening in this type of medium access. Note, however, that this energy wastage can be reduced by introducing sleep/active duty cycling as we have seen in SMAC [111, 112].

Schedule-based medium access is in general collision-free and idle listening is reduced to zero when nodes are able to establish perfect synchronization. This sub-class of TDMA has good outlook of being energy-efficient, however, we note that establishing the medium access schedule requires overhead, e.g. the random phase in the TRAMA protocol [85], in which nodes potentially idle listen, overhear and are subject to message collisions.

3.9.2 Self-organization

Of the three sub-classes of TDMA, schedule-based requires the most attention regarding organization. The schedule needs to be established and nodes are required to synchronize and maintain synchronization. In MMF-TDMA [98], we saw a distributed algorithm proposed to let parent nodes coordinate the time slot allocation of their siblings. The creation of the schedule has to be done before actual data communication can take place. Network start-up is therefore a difficult subject in schedule-based access.

Different are random-based, and contention-based access. The first simply lacks organization and the latter relies on the listen-before-talk strategy to organize communication on the fly.

3.9.3 Latency and delivery ratio

Contention-based MAC protocols potentially perform best on latency aspects. However, the message delay is dependant on the actual channel utilization and the number of competing nodes. The delivery ratio is also dependant on these parameters. In many comparative MAC simulations e.g. by Langendoen et al. [59], the delivery ratio drops rapidly when the message frequency is increased compared to schedule-based access. Yet, one of the characteristics of wireless sensor networks is the fact that high peak loads can occur (Chapter 2) due to (1) event detection by many nodes, and (2) route discovery.

The message delay in schedule-based protocols is obviously dependant on the actual schedule of the nodes. Although nodes have messages ready for transmission, they have to wait until the schedule allows them to transmit. Although latency is often seen as a weak point of schedule-based access, some protocols have successfully reduced the message delay, e.g. DMAC [65]. The delivery ratio of schedule-based MAC protocols is naturally good due to the fact that each node has time reserved to carry out its transmissions.

It is unclear how random-based medium access performs on (multi-hop) message delay and delivery ratio. In f-MAC [91] —for example– nodes are required to transmit many message duplicates. This obviously has a negative effect on throughput and message delay.

3.9.4 Scalability

Scalability is a weak point of schedule-based access. The underlying mechanism of this sub-class of TDMA reserves time slots for nodes. When more nodes are added than there are time slots available, the network does not scale any more. The contention-based protocols do not suffer from this effect, although adding more nodes would potentially decrease the probability that nodes can use the wireless medium.

f-MAC is not scalable in the sense that for collision-free operation nodes transmit as many duplicate messages as they have neighbours. Moreover, the lack of organization in random-based medium access makes it questionable how this sub-class of TDMA performs on scalability.

3.9.5 Resource consumption

The lack of organization in random-based access makes these protocols quite lean on resource usage. Roedig et al. [91] state that their f-MAC protocol needs a timer with predefined (unique) period. Whenever a message arrives from a higher protocol layer, the node simply transmits the message without assessing the wireless channel or backlogging the message until the schedule allows transmission. This kind of functionality is, however, required in contention-based and schedule-based MAC protocols. Moreover, to maintain a level of energy-efficiency, the sub-classes require time synchronization, which adds to resource consumption.

3.9.6 Conclusion

We conclude that schedule-based medium access is a good candidate in being energy-efficient and having high delivery ratio in networks with high peak loads [95]. These are very important aspects for wireless sensor networks, as we concluded in Chapter 2. Our work will therefore be focussed on schedule-based medium access.

In our work, we especially target message delay and self-organization in dynamic network topologies.

Chapter 4

Self-organizing algorithm for medium access scheduling

To ensure a long-lived network of wireless communicating sensors, it is necessary to have a medium access control protocol that is able to prevent energy-wasting behaviour like idle listening, hidden terminal problem or collision of packets. Schedule-based medium access protocols are in general robust against these effects, but require a mechanism to establish non-conflicting schedules. In this chapter, we present such a scheduling mechanism, which allows wireless sensors to choose a time interval for transmission, which is not interfering or causing collisions with other transmissions. In our proposed solution, we do not assume any hierarchical organization in the network and all operation is localized, making the network self-configuring.

4.1 Introduction

We argued in Chapter 3 that schedule-based MAC protocols are well-suited for wireless sensor networks, since energy-wasting effects like idle listening, hidden terminal problems and collisions of packets are minimized. In addition, schedule-based medium access provides good data throughput characteristics [95]. For these reasons, this type of medium sharing is well suited for (energy-constrained) WSNs. In this chapter, we propose a medium access scheduling principle and analyse its parameters.

With an eye to limited hardware capabilities of wireless sensors, the proposed medium access scheme is kept very simple. Each node gets periodically a time interval

—a so called time slot– in which it is allowed to control the wireless medium according its own requirements and needs. Outside this interval, nodes are notified when they are intended receivers. When a node is not needed for communication, it switches its transceiver to standby and is hence able to conserve energy.

Schedule-based MAC protocols often lack the ability to be self-organizing, which is a key issue in wireless sensor networks. Why should WSNs be self-organizing? In general, these networks are assumed to be large networks, both in number of nodes and coverage area [4]. In addition, nodes in the network can be mobile. This makes (manual or central) configuring during network deployment an unattractive option. Hence, to ensure scalability of the network, WSNs must be self-organizing. In our work, we assume base stations or central managers (which assign schedules) not to be present. We propose a scheme in which nodes are able to figure out their schedules based upon local information only. Additionally, the communication is self-starting.

During the initiating of the scheduled medium access (Section 4.4), collisions can occur due to nodes that choose simultaneously identical time slots. Yet we require that all nodes in the WSN can carry out their transmissions without causing interference or collisions. We discuss the (distributed) resolving of collisions (Section 4.5) and verify the proposed medium access mechanism with the model checker Uppaal in Chapter 5.

Allocating time slots for nodes is comparable to the NP-hard graph colouring problem. In this view, we analyse how many time slots are at least required to give all nodes an opportunity to communicate without causing interference (Section 4.6).

The chapter ends with conclusions. In Chapters 6 and 7, we present EMACs and LMAC, two MAC protocols based upon the approach presented in this chapter.

4.2 Assumptions

Our medium scheduling mechanism depends on the following assumptions:

1. **Spatial medium reuse** — The RTS/CTS handshaking mechanism proposed by Bharghavan et al. [12] and the associated channel assessment mechanisms (Section 3.1.3) are adequate to prevent collisions or interference to on-going data transmissions. Consequently, the wireless medium can indeed be spatially reused.

2. **Correctness of received information** — The physical layer is able to provide feedback to the data link layer about the correctness of the received packet. Incorrect packets are discarded (see also Appendix C.1). In addition, the physical layer is able to provide the reason of incorrect packets. In our work, we assume that the physical layer can for example distinguish between corrupt packets due to (random) bit errors and due to collisions.

3. **Timing** — Nodes can derive the moment at which packets are sent to facilitate synchronization. Synchronization is key-issue in schedule-based medium access. We assume network-wide synchronization between nodes.

Requirements (1) and (2) are discussed in Appendix C. Requirement (3) should be easy to satisfy in practice, e.g. all devices in our transceiver overview (Table A.1) provide feedback when transmissions started. In addition, we carried out experiments (Section 7.2.4) illustrating the wireless sensor node synchronization capabilities.

For self-starting of the network, we make the following assumption. We presume that at least one gateway node is present in the wireless sensor network and we require *exactly one* gateway node to initiate the scheduled medium access in order to prevent multiple misaligned schedules. This gateway is not involved in assigning time slots to nodes; its data link layer is identical to that of other nodes.

Our medium access approach is in particular useful in multi-hop networks in which spatial reuse and autonomous configuration are important. In addition, the medium access scheme is robust against mobility and dynamic topologies (e.g. due to iterative deployment), because of the inherent characteristics of its self-configuring.

4.3 Scheduling mechanism for medium access

In this section, we present a lightweight medium access scheduling algorithm that allows nodes to (autonomously) choose a time slot, which is not interfering with the communication between other nodes in the network. In addition, the algorithm resolves schedule conflicts, which might occur, for example, when nodes are mobile and travel through the network. A key-issue and novel in the presented work is self-organizing reuse of the wireless medium in combination with scheduled medium access.

First, we present the basic structure of medium access scheduling. Then, we discuss how nodes can autonomously choose a conflict-free time slot.

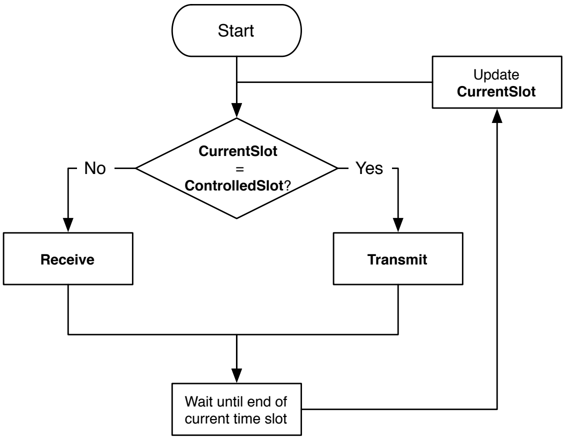


Figure 4.1 — Medium access scheduling principle

4.3.1 Scheduling principle

We propose a very simple scheduled medium access mechanism for wireless sensor networks. Each node takes control of (at least) one time slot. We refer to this time slot as *controlled time slot* and during this time slot, a node is allowed to *transmit* packets. Thus, a node uses only its controlled time slot to transfer data to neighbouring nodes.

During the time slots of other nodes, a node *receives* packets. If a node decides that it is not required for communication during the current time slot, it switches its transceiver to low-power mode in order to conserve energy. Figure 4.1 illustrates the scheduling mechanism.

In our approach, we assume that the schedule in the network is fixed and is repeated periodically. For this purpose, we introduce the concept of frames —consisting of a (integer) number of time slots– to indicate the period of scheduling (see also Section 3.1.2). The variable CurrentSlot indicates the current slot position in the MAC frame.

In principle, nodes would indefinitely (until their power source runs out) stick to their controlled time slot, unless one of the following happens:

- **Conflict/collision** — Due to, for example, mobility of nodes or dynamic quality of radio links, nodes controlling the same time slot may come too close. In this case, we speak of a collision. When a collision occurs, receiving nodes are not able to decipher packets and thus the nodes causing the collision should reconsider their controlled time slot. The topic of resolving collisions is discussed in Section 4.5.

- **Synchronization errors** — In schedule-based MAC protocols, timing and synchronization are, for obvious reasons, of vital importance. When any synchronization error occurs, action should be taken to restore correct synchronization. Synchronisation is discussed in Section 7.2.4.

- **No neighbouring nodes** — When a node does not detect any node around it, it is not able to transfer its sensor readings or exchange any other messages. To keep trying to communicate, the node wastes energy. Therefore, it should fall back to its pre-deployment state and give up its controlled time slot.

- **Active/passive role change** — In Section 6.4, we introduce a method for reducing the number of nodes that is actively participating in networking. The so called *active* nodes create a communication backbone, which can be used by *passive* nodes. The passive nodes do not participate in routing and other networking tasks, and are hence able to save energy. When the clustering mechanism determines that a node can switch to passive mode, the node gives up its controlled time slot and the time slot can be used by other nodes.

In Chapter 2, we concluded that wireless sensor networks are assumed to be *always on* networks and thereof we implicitly require the nodes to keep the communication structure intact at all times.

4.3.2 Localized algorithm for conflict-free time slot choosing

The network diameter of WSNs is expected to be larger than the transmission and interference ranges of the individual wireless sensors. WSNs are thus assumed to be multi-hop networks, which allows for *spatial reuse* of the wireless medium. Obviously, this is beneficial for the network, because more data can be transported per second per meter (i.e. higher transport capacity) [3]. But it also requires the MAC protocol to take measures for ensuring successful transmissions and to prevent problems like the well-known *hidden terminal problem.* In this section, we present a localized algorithm for conflict-free time slot choosing in which time slots can spatially be reused. In general, nodes lack sufficient memory to obtain and maintain a global view of the network topology. Hence, a localized algorithm is required.

In Section 3.1.3, we saw that MAC protocols, built upon the handshaking mechanism proposed by Bharghavan et al. [12], prevent nodes to use the wireless medium when it would disturb ongoing communications. The handshaking mechanism of these contention-based MAC protocols consists of two steps (as already discussed in Section 3.1.3.i, but repeated briefly for clarity): (1) reservation of the medium by the transmitting node and (2) reservation by the receiving node.

In step (1), the medium is reserved by the RTS message and all nodes in range of the transmitter postpone transmissions. In step (2), the receiver replies with a CTS message and all nodes —that are able to receive this message– postpone their transmissions. Note that this method of medium reservation only works under the assumption of mutual radio coverage.

We propose a similar solution for medium reservation in schedule-based MAC protocols. In our solution, we assume the scheduling mechanism as presented in Section 4.3.1. Nodes determine what time slots are available for use and what time slots interfere with other nodes. We require each node to transmit *at least once* during its controlled slot(s). Through this method, we make sure that all nodes in radio range are aware of the node, comparable to the reservation of the medium by the transmitting node (RTS) in the above described handshaking mechanism.

4.3.2.i Medium reservation by transmitting nodes

Nodes, which are trying to find a non-interfering time slot, remove all time slots in which a message is received (or a carrier is detected) from the list of potential non-interfering candidates. There are two obvious reasons not to choose a time slot, which is already in use by a node in radio range: (1) a node would not be able to exchange messages with one or more of its neighbours, since nodes are only allowed to transmit messages in their controlled time slot. And (2) a node would potentially cause collisions, such that it might block other nodes for communicating with their neighbours.

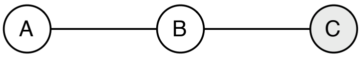


Figure 4.2 — Node C should not choose the time slot of node A as controlled time slot, since this would cause collision at node B. Node B should advertise the controlled slot of A

4.3.2.ii Medium reservation by receiving nodes

More difficult is the reservation of the medium by the receiving node. Consider the two-hop network in Figure 4.2. Lets assume that node A and B have different controlled time slots and node C is about to choose a controlled time slot. Given the above reasoning, C would not choose the same time slot as node B. However, when it chooses the same time slot as node A, collision occurs at node B and both A and C would not be able to transmit messages to node B. Node B should thus advertise to node C that it should not take the time slot of A, since a direct connection does not exist between A and C.

In general, nodes are required to transmit a list of their neighbours' controlled slots to give newly joined nodes in the network opportunity to determine which time slots can be used without interfering with other transmissions. In Section 4.3.3, we give suggestions on how this list can be efficiently be broadcasted between nodes.

4.3.2.iii Time slot selection

A newly joined node collects time slot usage information during one complete frame. After this information is collected, it can compile a set of time slots, which are not in use by its neighbour nodes or the neighbours of its neighbours (the groups might overlap). From this list *any* slot can be chosen as a controlled time slot without causing collision to any other transmission. For now, we assume that nodes (uniformly) randomly choose a non-interfering time slot from the list. In Section 7.5, we discuss different strategies with goal to reduce latency.

Note, that we assume that *communication* between the choosing nodes is not yet possible, therefore, all our strategies will have a random time slot selection nature. Consequently, (local) optimization of time slot choice is not (yet) possible.

4.3.3 Implementation aspect: Bit vector of occupied time slots

In the previous section, we argued that a node should not pick a time slot that is in use by its *first* or *second* order neighbours to ensure that transmissions are not interfering. We concluded that nodes should transmit a list of the time slots, which are in use by their neighbours. This can efficiently be implemented using bit vectors of occupied slots.

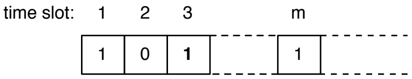


Figure 4.3 — Bit vector of occupied slots. A 0 indicates that the node did not detect any transmission in the respective time slot, otherwise a 1 is inserted. In addition, the node inserts a 1 at the position of its controlled slot

Consider a bit vector with a length (in bits) equal to the number of time slots per MAC frame. Each bit position in the vector represents a distinct time slot (Figure 4.3).

A "0" is placed when no transmission is detected and otherwise a "1" is inserted. Additionally, a node adds a 1 at the bit position of its controlled time slot. All nodes internally maintain such a bit vector and transmit the result during their controlled slot.

To get a (local) two-hop view of the network, a node simply has to collect transmitted bit vectors, while it keeps its own local bit vector up to date. When a complete frame has passed, the node can pinpoint non-interfering time slots by applying *OR*-logic to all received bit vectors and the local bit vector. A "1" in the end result means that a node choosing that slot would interfere with other transmissions and a "0" in the result means that the time slot can be taken. The node scans the resulting vector for 0's and records the respective bit positions to get a complete list of non-interfering time slots.

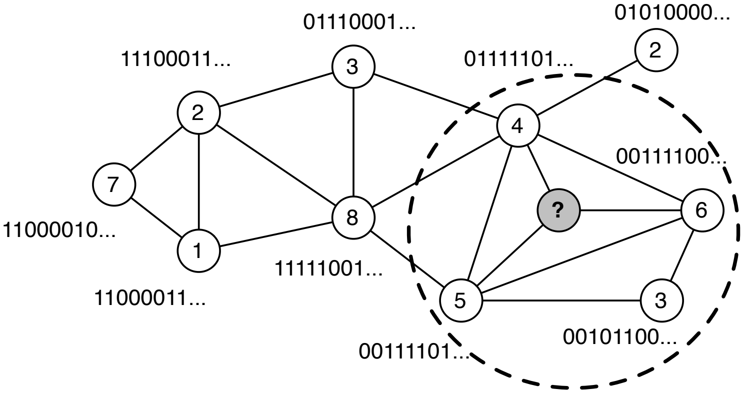


Figure 4.4 — Bit vector logic to determine what time slot can be used (numbers indicate controlled time slot of node). The grey node is observing the network to compile a list of non-interfering slots

We illustrate the mechanism with an example in Figure 4.4. Consider the grey node in the figure. It receives messages in time slots 3, 4, 5 and 6. Therefore, it constructs the following local bit vector: 00111100.... Its first order neighbours advertised the following bit vectors: 00101100... (received in time slot 3), 01111101... (slot 4), 00111101... (slot 5) and 00111100... (slot 6). The result of the OR-operation on the vectors is 01111101.... The node would find logical zero's at positions 1 and 7 in the resulting bit vector and would conclude that it can safely use time slot 1 or 7 without interfering other transmissions. In Figure 4.4, nodes using time slots 1 or 7 are indeed more than two hops —three and four hops, respectively– away from the grey node and the hidden terminal problem is prevented.

4.3.4 Outlook

In the previous sections, we have presented a self-organizing medium scheduling approach in which each node evaluates the time slot usage of its neighbours and second order neighbours. Using this information, it (locally) determines what time slots can be used without causing interference to any other node already controlling a time slot. Consequently, the wireless medium is spatially reused when this does not cause interference.

We answer the following questions:

- How can the scheduled medium usage be initiated in a newly deployed network?
- Nodes base their time slot choice upon the slot usage of other nodes. When multiple nodes choose a time slot simultaneously, schedule conflicts might occur. Similarly, when nodes are mobile and travel too close to nodes having the same time slot, collisions occur. How to detect schedule conflicts and how to solve these?

These questions are answered in Sections 4.4 and 4.5, respectively.

- Given the fact that concurrent transmissions can coexist under certain conditions, and therefore, some nodes in the network can transmit during the same time slot, how many time slots are required in multi-hop networks to provide all nodes in the network at least one time slot?

This question is answered in Section 4.6.

4.4 Initiating scheduled medium usage

In the previous section, we discussed how nodes collect a two-hop view of the (local) network to choose a non-interfering time slot. However, this solves only the self-organizing aspect and not the self-starting aspects of WSNs as we sketched in Section 4.1. In this section, we discuss how to initiate/setup the scheduling mechanism.

In traditional MAC protocols —mostly random access or contention-based protocols, like IEEE 802.11x in DCF mode (Section 3.3)– self-starting requires no additional attention. Nodes simply start listening to the wireless channel and respond to incoming requests and generate outgoing request when they see fit.

In schedule-based protocols (or any other MAC protocol that requires synchronization), initiating requires much more attention, because a random "just starting" approach —like in SMAC– would result in many different timing schemes, which do not necessarily co-exist i.e. nodes between groups with different schemes might experience collisions.

One of the characteristics of wireless sensor networks is that there is only a small set of nodes that have an interest in the sensor readings. As result, most of the information is hopping towards these —so called– gateway nodes, as we concluded in Section 2.3.7.i. Without any interest in the sensor readings, the network is by definition wasting energy by gathering them and keeping communication structures intact. Therefore, gateway nodes —as advocators of interest– are given a special role in our schedule-based protocol for network initialisation.

In our protocol, gateway nodes take —in contrast to non-gateway nodes– initiative in creating timing schemes. They do this by starting to take control of a time slot. Its one-hop neighbours will receive the transmission and will synchronize their clocks to the gateway. This triggers the nodes to choose a time slot themselves. Their neighbours will detect transmissions and, in this way, the synchronization event propagates through the entire network until every node is participating and controlling a time slot.

To establish the above described functionality, we define four operational states for nodes not performing gateway tasks (see also Figure 5.1):

- **Initialisation/pre-deployment state (I)** — Nodes sample the wireless medium (at a slow rate to conserve energy, like in [83]) to detect transmissions of other nodes. When a neighbouring node is detected, the node synchronizes (i.e. the node knows the current slot number in the MAC frame). When a new frame is due, the node switches to the wait state *W*. However, a node remains in initialisation state when it cannot retrieve the current slot number in a message. Thus a node does not commence to the wait state when it receives a collision or an erroneous packet.

- **Wait state (W)** — The purpose of the wait state is to spread out the wakening times of nodes to prevent energy-consuming schedule conflicts caused by nodes that choose identical time slots (Section 4.5). We observe that, especially at network setup, many nodes simultaneous enter the process of obtaining a time slot. This potentially leads to many collisions. Clearly, this is the case for nodes around the gateway.

  We introduce randomness in reaction time w between synchronization with the network and the actual choosing of a free time slot: $w = \{0 \ldots w_{max}\}$, expressed in (integer number of) MAC frames. After the random wait time, the node continues with the discover state *D*. Section 7.4 discusses the effects of the waiting time parameter.

- **Discover state (D)** — The node collects first and second order time slot usage information during one entire frame and records time slots to be occupied when the signal level is higher than a pre-defined threshold (i.e. carrier is detected).

  When all information is collected, the node chooses a time slot and advances to the active state *C*. If a node receives no useful information during the frame it is in discover state, it falls back to the initialization state *I*.

- **Continuous operation state (C)** — The node transmits in its time slot of choice (every MAC frame). Meanwhile, it listens to other time slots and accepts data from neighbouring nodes. The node also keeps its view on the network up-to-date. When a neighbouring node informs that there was a collision in the time slot of the node, the node immediately gives up its time slot and continues in the wait state *W*[1].

The gateway nodes omit states *I*, *W* and *D*, but start directly with controlling a time slot. A gateway node can be identical to a non-gateway wireless sensor. No additional processing power or memory is required.

[1] see also Section 4.3.1 for a complete list of conditions in which the node falls back to the I or W state.

4.5 Schedule conflicts

Collisions can occur when two or more nodes choose the same time slot to control. This can happen during network setup or when network topology changes due to —for example– mobility of nodes or variations in link quality.

As a result of collisions, nodes are not able to communicate in a proper sense with all nodes in radio range, as we have seen in Section 4.3.2. Both transmitting and receiving wireless sensors waste energy, since the contents of packets cannot be deciphered. Therefore, it is key issue to solve collisions preferably as quickly as possible. In Section 4.5.1, we introduce a collision reporting and solving mechanism.

With the algorithm presented in Section 4.3.2, nodes are able to determine which time slots can be used without interfering with other nodes. We stated that the selecting procedure of a time slot from the set of non-interfering ones has a random nature, because the nodes without time slots are not yet able to communicate with each other. So, strategies that require communication, like the node with highest ID may select a time slot first, cannot be applied, although these strategies would potentially lead to collision-free time slot allocation. The proposed random selection procedure results —even very likely– in collisions; Section 4.5.2 presents a statistical analysis of colliding time slots under assumption that time slots are (uniformly) randomly chosen.

4.5.1 Detecting and reporting collisions

The nodes that caused the collision cannot detect the collision by themselves; they need to be informed by their neighbouring wireless sensors, simply because they are transmitting when the event occurs. For exactly this reason, we concluded in Section 3.1.3 that CSMA/CD is not suitable for WSNs.

Nodes cannot extract any information from colliding packets, so the neighbouring nodes do not have any knowledge on e.g. the ID's of the nodes that caused the collision. Therefore, nodes can only report *in which time slot* a collision occurred. When nodes detect a collision, they use their *own* controlled time slot to inform neighbouring nodes that they detected a collision. The neighbours on their turn check their controlled time slot against the transmitted collision information. If it matches, they conclude that they are in collision with another node, release their time slot and initiate the procedure of obtaining a non-interfering one by returning to the wait state W (Section 4.4).

To speed up the collision reporting process, nodes prevent duplicate notifications, i.e. when a collision is reported by another node and the node did detect collision in the same time slot, it will not repeat the report. Instead, it reports another collision that potentially occurred in the past frame.

In Figure 4.5, three nodes transmit during time slot 2. This causes collision at nodes A, B and C, who record the event. During time slot 3, node B is the first node that gets an opportunity to report the collision. The two colliding nodes, in direct link with node B, reinitiate the process of obtaining a non-interfering time slot. During the next time slot, node C gets an opportunity to report the collision. However, it will not do so, since it heard node B already announce the collision. At the beginning of the next frame, A reports the collision, because it did not hear another node report

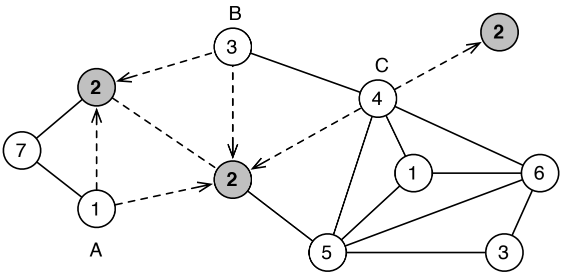


Figure 4.5 — The three grey nodes cause a collision in time slot 2. Nodes A, B and C detect the collision

it.

In this collision reporting scheme, we put the responsibility of detecting interfering time slots by the receiving party and the actual solving of the interference by the transmitting party. In Chapter 5 this mechanism is verified.

4.5.2 Statistical analysis of collisions

In this section, we analyse the statistical behaviour of nodes competing simultaneously for time slots. We determine the yield of the random time slot selection, i.e. the number of nodes that is not in conflict after choosing a slot. A low yield of unique choices makes the network slow in starting up and wastes energy in the process of self-configuration.

4.5.2.i Definitions and approach

We define $k > 0$ to be the number of nodes competing *simultaneously* for time slots, u $(0 \leq u \leq k)$ the number of nodes that makes a unique choice and n the number of free time slots. We assume that the number of time slots available is equal or greater than the number of competing nodes i.e. $n \geq k$ and that all k nodes make their choice from the same set of n time slots.

- A first observation is that a collision always happens between two or more nodes. Thus, the probability that *exactly one* node is in collision is $P(u = k - 1) \triangleq 0$.
- A second observation is that a collision is quite likely to occur (this effect is known as the *birthday paradox*). Consider the first node of the k nodes. This node can choose from n time slots without causing collision. The second can choose from $n - 1$ time slots, the third from $n - 2$ time slots etc. Thus the probability that all k nodes make unique time slot choices (i.e. $u = k$) is given by

$$P(u = k) = \frac{n(n-1)(n-2)\dots(n-k+1)}{n^k} = \frac{n!}{(n-k)!n^k} = \prod_{i=1}^{k-1}(1 - \frac{i}{n}) \quad (4.1)$$

The probability of at least two nodes choosing the same time slot is

$$P(u < k) = 1 - P(u = k) = 1 - \frac{n!}{(n-k)!n^k} \tag{4.2}$$

In Figure 4.6, the above probability is plotted for different values of n. Clearly the term n^k in Equation 4.2 grows faster than $n!$ (assuming $n \backsim k$), resulting in an increasing probability of collision with a growing number of nodes. A higher ratio of non-interfering time slots per node results in a smaller probability of conflict.

Diaconis et al. [21] approximate $P(u < k) = 1 - \prod_{i=1}^{k-1}(1 - \frac{i}{n})$ by $1 - e^{\frac{-k^2}{2n}}$, resulting in the simple relation $k \leq 1.2\sqrt{n}$ for $\leq 50\%$ probability of collision. In other words, the number of free time slots should roughly be equal to the square of the number of competing nodes in order to have no collisions in *half* of the cases. In practice, this vast amount of unoccupied time slots is —for latency reasons– unfeasible.

We conclude that it is quite likely that at least one collision will occur and two or more nodes will have to redo the procedures of obtaining a time slot.

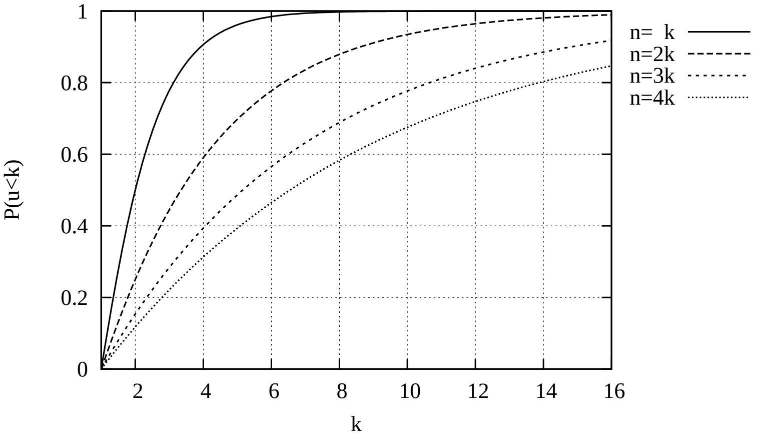


Figure 4.6 — Probability that at least two nodes choose identical time slots (i.e. $u < k$)

4.5.2.ii Average number of surviving nodes

How many nodes make on average a unique choice? We derive a method for counting the number of non-conflicting nodes based on the well known *principle of inclusion and exclusion* ([64], Chapter 10). This principle is a generalisation of what is shown in Figure 4.7: $|A \cup B| = |A| + |B| - |A \cap B|$, with $|A|$ the number of elements in set A.

Lemma 4.5.1. *Let k nodes simultaneously choose a slot from n non-interfering time*

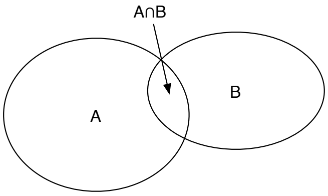


Figure 4.7 — Two partially overlapping sets A and B

slots. The number of combinations with exactly u *surviving nodes is given by*

$$E_k^n(u) = \frac{(-1)^u n!k!}{u!} \sum_{i=u}^{k} \frac{(-1)^i (n-i)^{k-i}}{(i-u)!(n-i)!(k-i)!} \tag{4.3}$$

Proof. The result follows directly from the principle of inclusion and exclusion. Consider a set S and conditions c_i $(1 \leq i \leq k)$ which are satisfied by some elements of S. In our case, condition c_i is satisfied when node i is not in collision after the time slot selecting process. Let S_t be the number of combinations in which t requirements are fulfilled, ignoring remaining conditions.

Let $E_k^n(u)$ be the number of elements of set S which fulfil exactly u of the conditions (including $k-u$ conditions not fulfilled), then

$$E_k^n(u) = \binom{u}{0} S_u - \binom{u+1}{1} S_{u+1} + \binom{u+2}{2} S_{u+2} - \ldots + (-1)^{k-u} \binom{k}{k-u} S_k$$

$$= \sum_{i=u}^{k} (-1)^{i-u} \binom{i}{i-u} S_i \tag{4.4}$$

A thorough explanation and proof of this principle can be found in [35].

Next, we derive an expression for S_t. First, we consider t surviving nodes and place them uniquely in a time slot. We can arrange these t nodes in $k(k-1)(k-2)\ldots(k-t+1) = \frac{k!}{(k-t)!}$ different ways and select time slots for them $\binom{n}{t}$ ways. Hereby, we fulfil t conditions. The remaining $k-t$ nodes can be arranged in over the $n-t$ remaining time slots in $(n-t)^{k-t}$ different ways. Note that the later term also includes combinations in which more conditions are fulfilled; for this reason the principle of inclusion and exclusion is used.

Thus, for S_t we can write

$$S_t = \frac{k!}{(k-t)!} \binom{n}{t} (n-t)^{k-t} \tag{4.5}$$

Substituting Equation 4.5 in Equation 4.4 gives the result of Equation 4.3. □

Equation 4.3 gives us exactly how many combinations there exist that result in a given number of surviving nodes. Dividing this by the total number of possible combinations i.e. n^k gives the associated probability (Figure 4.8):

$$P(u) = \frac{E_k^n(u)}{n^k} \tag{4.6}$$

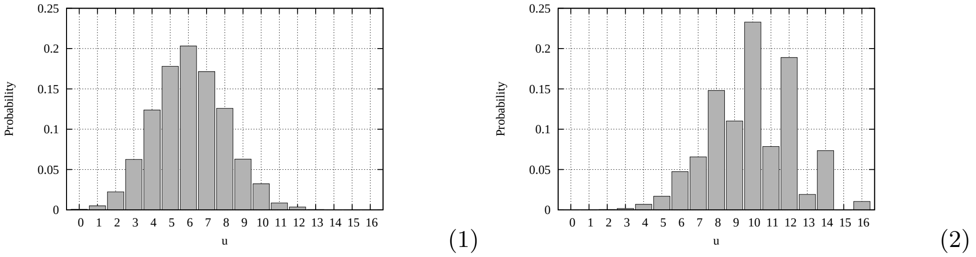


Figure 4.8 — Probability distribution of u for (1) $n = k$, and (2) $n = 2k$. In both cases $k = 16$

The average yield of surviving nodes is given by (Figure 4.9)

$$\bar{U}_k^n = \sum_{u=0}^{k} uP(u) = \frac{1}{n^k} \sum_{u=0}^{k} u \frac{(-1)^u \, n!k!}{u!} \sum_{i=u}^{k} \frac{(-1)^i \, (n-i)^{(k-i)}}{(i-u)! \, (n-i)! \, (k-i)!} \tag{4.7}$$

with standard deviation

$$\sigma_k^n = \sqrt{\frac{1}{n^k} \sum_{u=0}^{k} \left(u \;\; E_k^n(u) - \bar{U}_k^n\right)^2} \tag{4.8}$$

For both cases in Figure 4.8, in the range $u = \bar{U}_k^n - \sigma_k^n \ldots \bar{U}_k^n + \sigma_k^n$ approximately 50% of the combinations is captured.

4.5.2.iii Number of rounds

Interestingly, the probability that a collision occurs in one or more time slots is quite high, as we argued in Section 4.5.2.i. This implies that it is most likely that multiple rounds are required before all k nodes use distinct time slots. By rounds we understand the following: (1) nodes make a (uniform) random choice from the set of non-interfering nodes, and (2) some nodes might pick distinct time slots, whereas others might be in collision. The latter group is notified in which time slots collisions occur and release their time slots. These two steps are repeated until all nodes have distinct time slots. We conclude that multiple rounds are necessary before each node has a distinct time slot.

On the other hand, the probability that *no* nodes survive the slot choosing process is quite small for such a large number of nodes. For example, when we look at $u =$

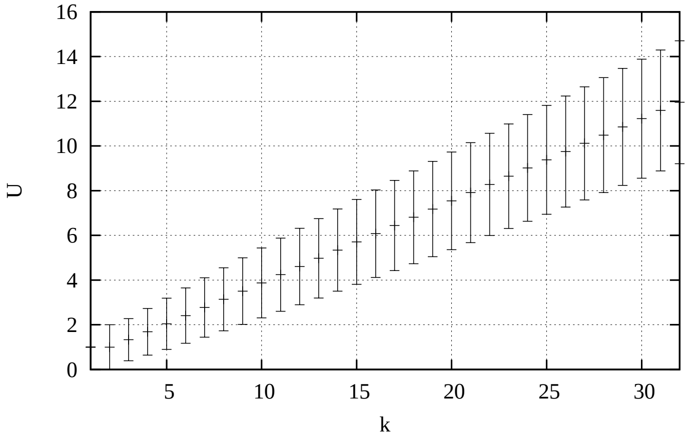


Figure 4.9 — Expected number of surviving nodes $\bar{U}_k^n$ and its standard deviation for $n = k$

0 cases in Figures 4.8(1) and (2), we find probabilities of 0.0513% and 0.0002%, respectively, for $n = 16$ and $n = 32$. Since these probabilities exist, the time slot choosing process is not guaranteed to finish in finite time. However, the probabilities are comfortably small, that quite likely in every round some nodes survive the process and progress is made in putting every node in a distinct time slot.

The progress of the (uniformly) random time slot choosing is less evident for smaller number of nodes. For example, when two nodes can choose from two time slots, then with equal odds they both survive, or both do not survive. However, when there are slightly more non-interfering time slots than nodes, the probability of no surviving nodes at all (i.e. $u = 0$) decreases. Adding one extra time slot to the $k = 2$ case already gives a probability of $\frac{2}{3}$ that both nodes survive. Moreover, the random wait time in state *W* —the state to which colliding nodes return, Section 4.4– spreads the number of competing nodes over time, thereby increasing the ratio of non-interfering time slots versus nodes.

Based upon the average yield of surviving nodes (Equation 4.7), we determine the average number of rounds that is required to give nodes distinct time slots (Figure 4.10). As expected, the required number of iterations drops with an increased ratio between non-interfering time slots and nodes. Thus, to speed up the network setup process, plenty of non-interfering time slots should be available.

Figure 4.9 suggests that there is almost a linear relationship between k and $\bar{U}_k^n$ of Equation 4.7; more than $\frac{1}{3}$ of the nodes would get on average distinct time slots in the $n = k$ case (i.e. worst case). We denote the average number of required rounds as $\bar{R}$. Then $\bar{R} - 1$ satisfies $k(\frac{2}{3})^{\bar{R}-1} = 1$ to reduce the number of non-surviving nodes to one. The average number of rounds required can therefore be (over)estimated by

$$\bar{R} = 1 - \log_{\frac{2}{3}}(k) \quad \text{where } k > 0 \tag{4.9}$$

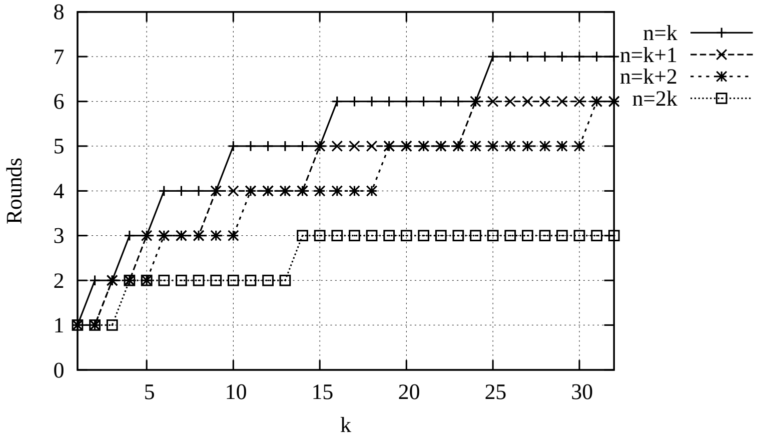


Figure 4.10 — Average number of rounds required before k nodes pick a non-colliding time slot

This concludes our analysis of colliding schedules.

4.6 Required number of time slots

The number of time slots in the MAC frame is an important parameter. It clearly has an influence on the probability that collisions occur, as we have seen in the previous section. Additionally, the parameter determines (best case) message delay (Section 7.5), because nodes are only allowed to transmit during their controlled time slot(s).

In this section, we answer the question of how many time slots are required to give each node in the network at least one time slot to carry out its transmission conflict-free from a network density perspective.

4.6.1 Theoretical approach: offline graph colouring

The problem of assigning time slots in a network is comparable to the NP-hard *graph colouring problem* [74, 98], in particular E^2-colouring. In the time slot assigning, second order neighbours have to be taken into consideration to reuse time slots not sooner than at three hops or more. The question that now arises is how many time slots are necessary to give each node in the network opportunity to use a non-conflicting time slot? In this section, we tackle this problem using graph theory.

Lemma 4.6.1. *Consider an arbitrary graph G with nodes V and connections between the nodes E. In such graph, the number of required time slots is bounded by*

$$\Delta + 1 \leq N_{timeslots} \leq \Delta^2 + 1 \tag{4.10}$$

with maximal connectivity $\Delta(G) = max\{d(v)|v \in V\}$ *and* $d(v)$ *the number of edges* $|E(v)|$ *of vertex* v*.*

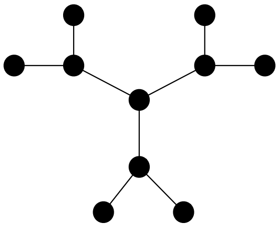


Figure 4.11 — Example graph showing lower bound $(\Delta + 1)$ of E^2-colouring

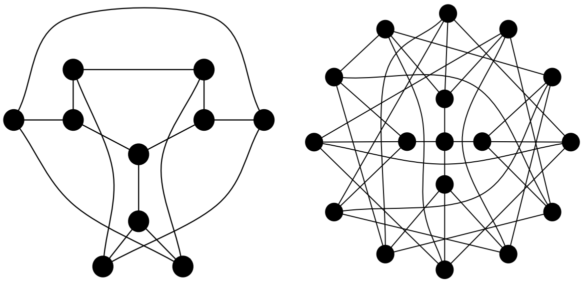


Figure 4.12 — Graphs showing upper bound of Equation 4.10 for $\Delta = 3$ and $\Delta = 4$

Proof. For successful communication, all first order neighbours of a node must communicate at different times, including the node itself. Therefore, at least $\Delta + 1$ time slots are required. An example of a graph that can be E^2-coloured with $\Delta + 1$ colours is shown in Figure 4.11. In this example, the node in the middle of the drawing (i.e. one of the vertices with highest connectivity) is the only node needing a unique time slot. We denote this node v_0. Time slots of v_0's first order neighbours can be reused by second order neighbours, in such way that no additional time slots are required.

We focus now on the second order neighbours of node v_0. In Figure 4.11 these nodes have Δ first and second order neighbours. However, we show in the next paragraph that second order neighbours of v_0 can be connected in such way —without affecting the maximal connectivity of the graph– that they become first or second order neighbours of all nodes in the graph. The created structure requires a unique time slot for every node, resulting in $N_{timeslots} = \Delta^2 + 1$. More nodes cannot be added to such a graph without affecting the maximum connectivity. This proves the upper bound. Figure 4.12 shows two graphs ($\Delta = 3$ and $\Delta = 4$) that require $N_{timeslots} = 10$ and $N_{timeslots} = 17$ time slots, respectively.

The first order neighbours of the node (i.e. v_0) with highest degree can have at most $\Delta - 1$ connections to other nodes, without changing the maximal connectivity. Thus v_0 has at most $\Delta(\Delta - 1) = \Delta^2 - \Delta$ second order nodes. We now count the number of connections these nodes can make. Each second order node of v_0 can make $\Delta - 1$ connections by itself and $(\Delta - 1)(\Delta - 2)$ second order connections, resulting in $\Delta - 1 + (\Delta - 1)(\Delta - 2) = \Delta^2 - 2\Delta + 1$ connections per node.

Each of the $\Delta^2 - \Delta$ second order neighbours of v_0 already connects to $\Delta - 1$ nodes, thus each second order node needs to connect to $(\Delta^2 - \Delta) - (\Delta - 1) = \Delta^2 - 2\Delta + 1$ nodes to make it first or second order neighbour of all nodes in the graph. This matches the number of connections a node can make. Hence, the number of time slots for arbitrary graphs is bound by:

$$\Delta + 1 \leq N_{timeslots} \leq \Delta^2 + 1 \tag{4.11}$$

□

4.6.2 Empirical approach: distance constrained graphs

Note that the upper bound on the number of time slots (Equation 4.10) necessary in a network, grows rapidly ($O(\Delta^2)$) with increasing maximal connectivity. This would be devastating for message delays in our schedule-based medium access.

In practice, however, not all situations will or can occur in a graph and, therefore, the practical required number of time slots is closer to the lower bound. The reason for this is the fact that in communication graphs connections are distance constrained. We demonstrate this empirically by colouring communication graphs in which radio coverage has been modelled circularly (i.e. a *unit disk graph*), as customary. We consider 500 connected random deployed networks to empirically determine the number of time slots necessary in this class of graphs. The topologies are generated with average connectivity $\bar{d} = 2\pi$ and contain 100 nodes.

We divided the topologies in sets with equal maximum connectivity Δ and per set the (minimum) number of necessary time slots is determined with offline exhaustive search. A certain fraction of the networks within a set could be coloured with $\Delta + 1$, $\Delta + 2$, etc colours. The results per set are shown in Figure 4.13.

We conclude from the figure that in these scenarios, the necessary time slots in random deployed networks is likely to be $O(\Delta)$ instead of $O(\Delta^2)$. In the worst case scenario, $\Delta + 3$ time slots per frame were necessary. More than 70% of the considered random network topologies could be E^2-coloured with $\Delta + 1$ colours.

These are important results for practical reasons. When the number of time slots per frame is large, nodes have to wait a long time before their time slot comes up in order to get the opportunity to transmit. This increases the reaction time of the network considerably: an undesired effect. To limit latency, it is key issue to keep the number of time slots in a frame to a minimum. Note that in practical networks the connectivity can be scaled with the transmission power settings.

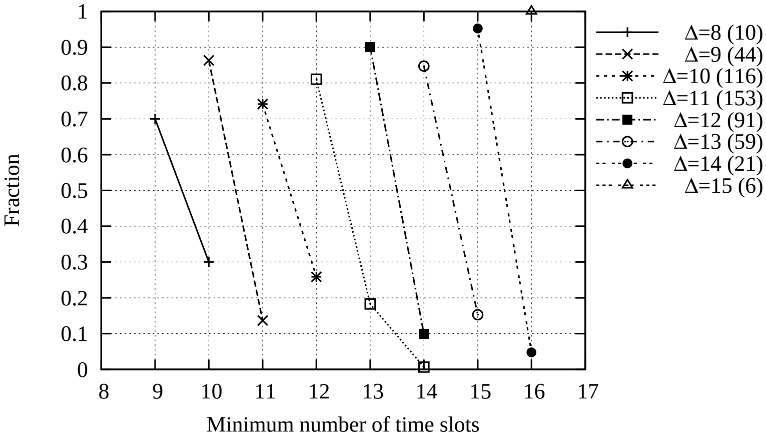


Figure 4.13 — Results of assigning time slots to random network topologies (offline exhaustive search). Set sizes are indicated between brackets. Per maximum connectivity category, the fraction of topologies is shown versus the required number of time slots

4.6.3 Practical approach: online Greedy graph colouring

In the previous sections, we determined from a theoretical perspective how many time slots are at least required in given topologies. In this section, we determine the performance of our (local) time slot assigning algorithm by experimental analysis. In the previous sections we considered offline and exhaustive searches. Now, we switch to the distributed and local approach of finding free time slots as executed by individual nodes in the wireless sensor network.

4.6.3.i Expected performance

In literature, many studies have been conducted in order to find heuristics that are able to find the minimum number of colours for given graphs. Since graph colouring is an NP-hard problem, heuristics trade in general accuracy for reduction of execution time. It is shown by Fiala et al. [33] that first-fit techniques like Greedy-approaches [103], approximate the minimum amount of colours by a factor 3 (in worst case). Our approach is based also on the first-fit (Greedy-approach) heuristic (yet we try to spread out the slot choice to minimize the number of collisions during rollout of the network). Therefore, we expect our algorithm to perform in the similar order.

4.6.3.ii Evaluation of performance

By simulation, we verify the performance of the algorithm. The same 500 network topologies are used as in Section 4.6.2. One node is assigned the first time slot and is put into the active state C. This node acts as gateway. Other nodes are put in the initialization state I.

We vary the number of time slots used by the algorithm and record in which topologies non-interfering time slots can be successfully assigned to all nodes. To rule out the effects of randomness in the algorithm, we only record runs as successful if for ten different random seeds all nodes choose non-interfering time slots. In Figure 4.14, the results are shown.

Again, we divided the topologies in sets with equal maximum connectivity and plotted the fraction of a set that could be coloured with a specific number of time slots. We conclude that the localized algorithm is successful within a factor 2 of the minimum necessary time slots $\Delta + 1$; well in range of the expected performance.

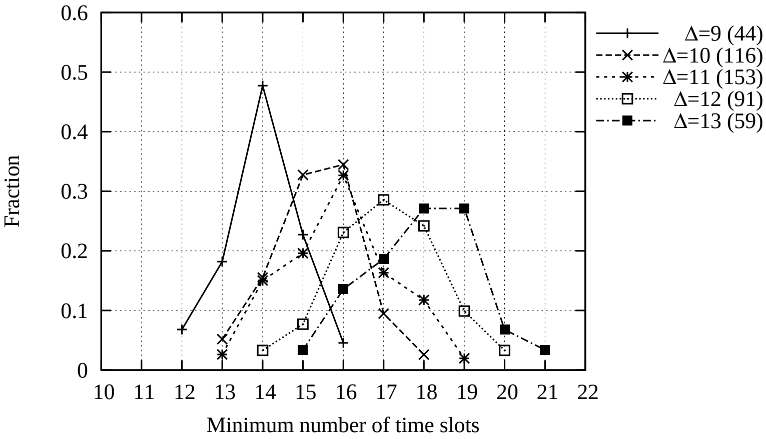


Figure 4.14 — Number of time slots necessary in random networks for different maximal connectivity's using the localized algorithm of Section 4.3 (simulation)

The results suggest that $N_{timeslots} = 2\Delta$ is sufficient for our medium access scheduling mechanism to provide every node with a non-conflicting slot. However, it is questionable whether this is enough in practical networks for the following reasons: (1) grey area effects (i.e. on/off RF link conditions [87]), (2) node mobility, and (3) iterative deployment. The estimation of Δ might be difficult in reality, however, it should be possible to coarsely foresee the network density.

A trade-off should be made between tolerable message delay and a not thrifty number of time slots to allow guaranteed operation of the network. We do not want to use the *minimum* number of time slots, but we merely want to use a reasonable number that allows for adding new nodes to the network or for changes in network density due to mobility. Over-provisioning and scalability remain potential problems in our schedule-based medium access solution.

4.6.4 Related work

Sridharan et al. present a time slot assigning mechanism, in which parent nodes select time slots for their siblings [98]. The work assumes that (1) the parent/sibling relations are established before initiating the assignment procedure, and (2) nodes can communicate collision-free whilst executing the algorithm. The algorithm adapts breadth first search in assigning the time slots and parent nodes communicate with all one hop neighbours to make sure that time slots can be used without causing collisions. The algorithm is a Greedy polynomial heuristic, but according to the authors, its performance matches optimal graph colouring results in small topologies [98].

Due to its hierarchical organization, the algorithm of Sridharan [98] is less suited for dynamic topologies or iterative deployment; the network has to re-examine the time slot allocation when the topology changes. However, the algorithm is useful to obtain a coarse estimate of the required number of time slots. Our scheduling mechanism (in which we assume the number of time slots per frame fixed) can use this estimate to adapt the number of time slots to the needs in the network. This topic is left to future work.

Moscibroda et al. [74] see the assigning of time slots in a wireless network as a chicken-and-egg problem, i.e. a MAC protocol is required to establish graph colouring and vice versa. But during initialization of the network, there is no MAC or time slot allocation.

The proposed algorithm elects cluster heads (i.e. a set of mutually independent nodes), which assign colour-ranges (i.e. time slots) to all its slaves. Since slaves of different clusters can be within radio coverage, they need to verify the assigned time slot(s). The slave nodes try the first time slot ϕ of the assigned range. Whenever a neighbour announces it already uses time slot ϕ, the slave node continues with verifying slot $\phi + 1$ etc. When it finds a free time slot, the node claims it and periodically transmits the above mentioned announcement. The authors show that this algorithm results in correct colouring with $O(\Delta)$ time slots in $O(\Delta \log N)$ time with high probability in unit disk graphs [74] (N represents the number of nodes in the network).

Interestingly, Moscibroda et al. [74] assume nodes without possibility to detect collisions, i.e. all decisions are based upon positive enforcement. Once a node has made its slot choice, it keeps using the slot. Therefore, the algorithm is only suited for static wireless networks.

4.7 Conclusion

In this chapter, we have presented a medium scheduling approach in which (1) nodes are self-configuring, (2) the wireless medium is spatially reused, and (3) conflicts (i.e. collision of schedules) are resolved when detected.

In the schedule-based medium access method, time is organized into time slots, which are then grouped into frames. Each frame has a fixed length of a (integer) number of time slots. Each time slot has a distinct number indicating its position in the frame. Each node takes control of (at least) one time slot and a node is allowed to transmit packets during this time slot. Thus, a node uses only its controlled time slot to transfer data to neighbouring nodes. During the time slots of other nodes, a node receives packets when it is addressed. Otherwise, the node switches its transceiver to low-power mode in order to conserve energy. In our approach, we assume that the schedule in the network is fixed and is repeated on a frame basis. In principle, nodes indefinitely stick to their controlled time slot.

We required nodes to transmit at least the following information during their controlled time slots: (1) the number of the current time slot, and (2) a list of time slots used by first order neighbours to ensure medium reservation by receivers.

1. **Self-configuration** — We proposed four operational states associated with the medium access scheme: (1) the initialization state in which transmissions of other nodes act as a trigger to move to the next state, (2) a wait state, (3) a discover state in which a node assesses what time slot(s) can be used conflict-free, and (4) the continuous operation mode. In the continuous operation mode, nodes apply the proposed medium scheduling. We assumed one node in the network, which initiates the entire network. This task is well-suited for a gateway since it acts as advocator of user interest in sensor readings.

2. **Spatial reuse of medium** — In the well-know RTS/CTS handshaking mechanism, the wireless medium is reserved by both transmitter and receiver of the data message. This ensures that collision and interference to the data message are prevented. The medium can be reused outside this blocking area. We adapt a similar blocking of concurrent transmissions in time slots.

   We used graph theory to determine how many time slots are at least required in topologies with particular maximum connectivity. We mention that in practical networks a thrifty number of time slots should not be applied e.g. for mobility or iterative deployment arguments. A balance should be established between the mobility/iterative deployment requirements and latency requirements.

3. **Conflict resolving** — During initiation of the network by the assumed gateway, many nodes potentially enter the time slot choosing process. Since communication between these nodes is not yet possible, nodes might choose identical time slots, which obviously lead to schedule conflicts. We proposed and verified a mechanism to resolve these conflicts.

   We analyzed how many rounds are on average required to provide nodes a non-conflicting time slot. We concluded that a higher ratio between unoccupied time slots and choosing nodes reduces the number of required rounds. To increase this ratio, the wait state W has been introduced, which spreads the choosing of nodes in time. We explore the effect of this state in Section 7.4. Again, a balance should be established between the number of time slots and the start-up time of the network.

In the proposed medium scheduling concept, it is assumed that all nodes in the network have consensus of time, i.e. both current position in the MAC frame and beginning of the time slots. However, in some applications, it might occur that networks with different timings intertwine. This topic is an interesting challenge for future research.

Chapter 5

Verification of the algorithm

In the previous chapter, we presented general ideas on schedule-based medium access. Based upon local information only, each node autonomously decides when it can transmit without causing collision or interference to other transmissions. However, during network initialization or in dynamic topologies (e.g. due to node mobility) collisions might occur and, therefore, collision reporting has been introduced. In this chapter, we verify the algorithms proposed in Chapter 4.

5.1 Introduction

The basic ideas of the presented medium scheduling, its initiating and collision resolving mechanisms, are reasonably simple. However, due to its distributed and localized operation, the possible behaviour gets too complex to be overseen by pure insight. Formal analysis is, therefore, a possibility to increase confidence in the correctness of the protocol.

We apply model checking as an experimental approach [14]. Formal analysis can only increase the *trust* in the correctness of an implementation, but not guarantee it. We are aware that a formal correctness proof is only about a model, and not about the implementation. Moreover, we will not proof correctness for the general case, but only for instances of topologies.

The added value of model verification also lies in the formalization of the *protocol description*. As such, we have used model checking extensively [31, 32].

5.2 Approach

Our approach is to investigate systematically (small) networks with the model checker Uppaal (`http://www.uppaal.com`). In Section 5.2.1, we discuss our network model, which is composed of (1) node models (i.e. instances of our medium access scheduling scheme of Chapter 4) and (2) node interconnect (i.e. a shared wireless channel). In Section 5.2.2, we discuss the verified properties.

5.2.1 The model

Systems are modelled as non-deterministic, timed automata, by the Uppaal model checker. The non-deterministic aspect allows us to implement the randomness in time slot choice (Section 4.3) and wait times (Section 4.4).

Time is modelled using real-valued clocks and time only progresses at the locations of the automata. The transitions between the automata are instantaneous. Uppaal allows parallel composition of automata. Several automata may also synchronize on transitions using handshakes. With the use of shared variables it is possible to model data transfer between automata. Locations may be urgent, which means time is not allowed to progress, and committed, which means time is not allowed to progress and interleaving is restricted. If only one automaton is in a committed location at any one time, its transitions are guaranteed to be atomic. These timing aspects allow us to model several independent, yet synchronized nodes.

Our modelling approach is as follows (see [31, 32] for complete modelling details). We model one system wide (i.e. network wide) clock, which determines the time slot pace. This clock is modelled as a two phased clock. In the first phase, nodes, not in control of the current time slot, are put into receive state. Owners of the current time slot wait until the second phase before transmitting. With this mechanism, we overcome the fact that Uppaal non-deterministically chooses an automata transition to execute, when multiple transitions are possible. Without the two phased clock, it can be that time slot owners transmit before the other nodes are put into receive state.

Further, we created a general node model, which includes the four states presented in Section 4.4, the time slot choosing process of Section 4.3, the wireless medium scheduling (Section 4.3.1) and collision resolution mechanism (Section 4.5). Actually, we used the model checker to establish a clear description of the proposed protocol.

Systematically, we investigate all (small) network topologies. For each topology, we generate (automatically) a model, which captures the communication relations between nodes in the network. The networks are modelled as static. Uppaal checks properties of models by means of exhaustive search through the state space of the system. To satisfy the constraints of memory and time available for verifying, we limit ourselves to check all four and five node topologies.

5.2.2 Verified properties

The following properties are of interest for the self-organizing MAC scheduling:

- **Start-up property** — Every node should get an opportunity to choose a controlled time slot. The MAC frame length is in the models long enough to give each node a unique time slot. This property translates to the reachability of state C.

- **Collision property** — The goal of our schedule-based medium access method is to give each node an opportunity to communicate collision-free with its neighbours. Thus, when two nodes collide, at least one should reconsider its slot choice.

We mention one other property that we do not consider, although it immediately comes to mind: the finiteness of the slot choosing process. In Section 4.5, we already argued that the process is not necessarily finite, due to its random nature.

The two properties are checked and we record traces of topologies in which properties are not fulfilled. Note that when properties are found to be *not satisfied* by the model checker, this does not imply that in *all* instances of the topology, the property is not satisfied. However, it indicates that at least one trace exists in which the property is not satisfied. The model checker results do not show the significance of the protocol defects: how often do defects occur in particular topologies?

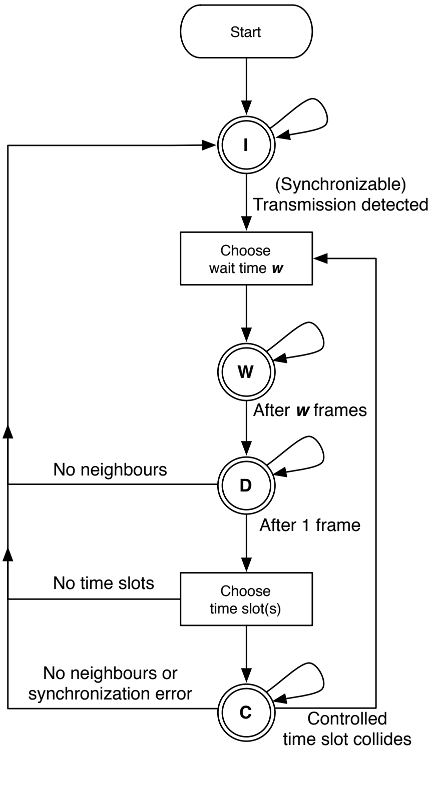


Figure 5.1 — State diagram of the node's behaviour. The operation in state C is sketched in Figure 4.1

5.3 Verification results

Figures 5.3, 5.4 and 5.5 show model checker results for the four and five node topologies. For a legend to the figures, refer to Figure 5.2.

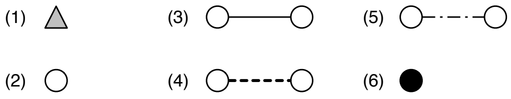


Figure 5.2 — (1) Gateway node (starting in state *C*) and (2) node (starting in state *I*). Connections (3) and (4) determine the topology. (3) Indicates that one or both nodes redo the time slot choosing process when a collision occurs. Type (4) denotes the cases where collisions *might* not be solved and, (5) similarly, for second order connections. When a node *might* not reach state *C*, this is indicated by (6).

5.3.1 Four node topologies

First, we focus on four node topologies of Figure 5.3. In topology (5), the collision property is not satisfied. It may happen in the particular topology that a collision occurs which is not resolved. In all other topologies, the properties are satisfied.

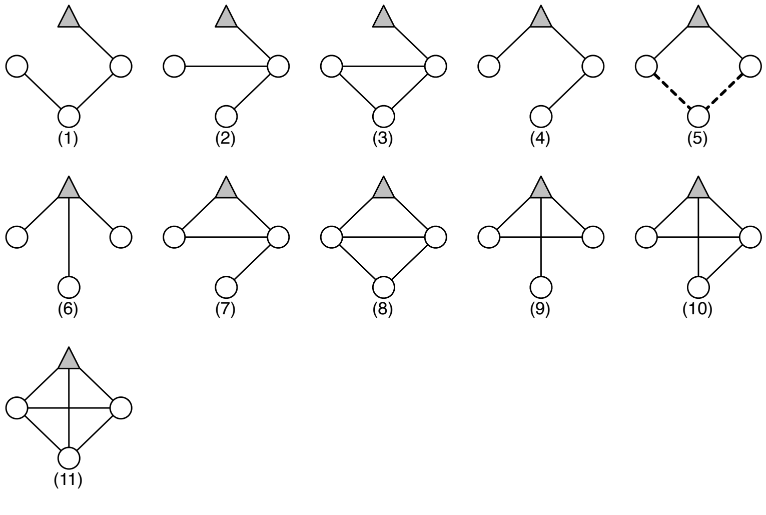


Figure 5.3 — Verification results for all topologies of one gateway and three nodes

Inspection of the recorded traces reveals that the bottom node in topology (5) might choose a time slot exactly at the same moment as the node on the left (or right). When those two nodes also choose the same time slot, a collision occurs. There is no node in the network, which detects the collision, and therefore, it is not reported and consequently the collision is not solved. Note that the occurrence of this defect is related to the probability of nodes choosing identical time slots, while simultaneously reaching state D (i.e. 8.3% for $W_{max} = 6$ and $n = 2$ free time slots for the nodes).

Square shaped or ring shaped topologies naturally lack additional node(s) to report collisions. Therefore, we expect the collision property not to be satisfied in these network structures.

The start-up property is satisfied in all four node topologies, even in topology (5) of Figure 5.3. In that particular topology, the nodes with conflicting time slots are still able to communicate (directly or via one hop) with the gateway in the network. The significance of the defect is, therefore, negligible from a communication perspective.

5.3.2 Five node topologies

The collision property is again not satisfied in the following five node topologies (Figures 5.4 and 5.5) in which the square structure can be recognized: (5), (17), (18), (20), (23), (24), (29), (34), (39), (41), (45) and (47). In these topologies, two nodes can choose the same time slot without an additional node noting the collision.

In some of the five node topologies, the start-up property is not satisfied, namely in topologies (14) and (19). We discuss these topologies briefly.

In topology (14), we see a special form of the square problem, in which both properties are not necessarily satisfied. The general problem in this topology is that a node can be sandwiched between two nodes with identical time slots. The sandwiched node detects the collision, however, when it did not obtain a time slot yet, it is not able to report the collision. Consequently, the node *never* reaches state C. Note that this happens to at most one node per instance of topology (14).

In topology (19), a different problem occurs in addition to the discussed square/ring problem. In this topology, two separate clusters of nodes can arise, which are separated by collisions. To elaborate this situation: the node at bottom right might pick the same time slot as the gateway after it synchronizes to the node on the bottom right, while the two nodes in direct connection with the gateway are in collision. In this situation, no data traffic is possible to and from the gateway at all.

5.4 Conclusion

We conclude that the proposed self-organizing algorithm for medium access scheduling does not satisfy the *start-up property* nor the *collision property* in all four and five node topologies (although the properties are satisfied in most topologies). There are three reasons for this

1. **Undetected collision** — When two nodes are in collision and there is no additional node to detect the collision, the collision is not solved, e.g. Figure 5.3(5). It is questionable whether this can (guaranteed) be solved, due to the random and distributed nature of the time slot choosing process.

   A solution to reduce the effect of this defect is to let nodes randomly receive during their own time slot. When a transmission is detected, the node concludes

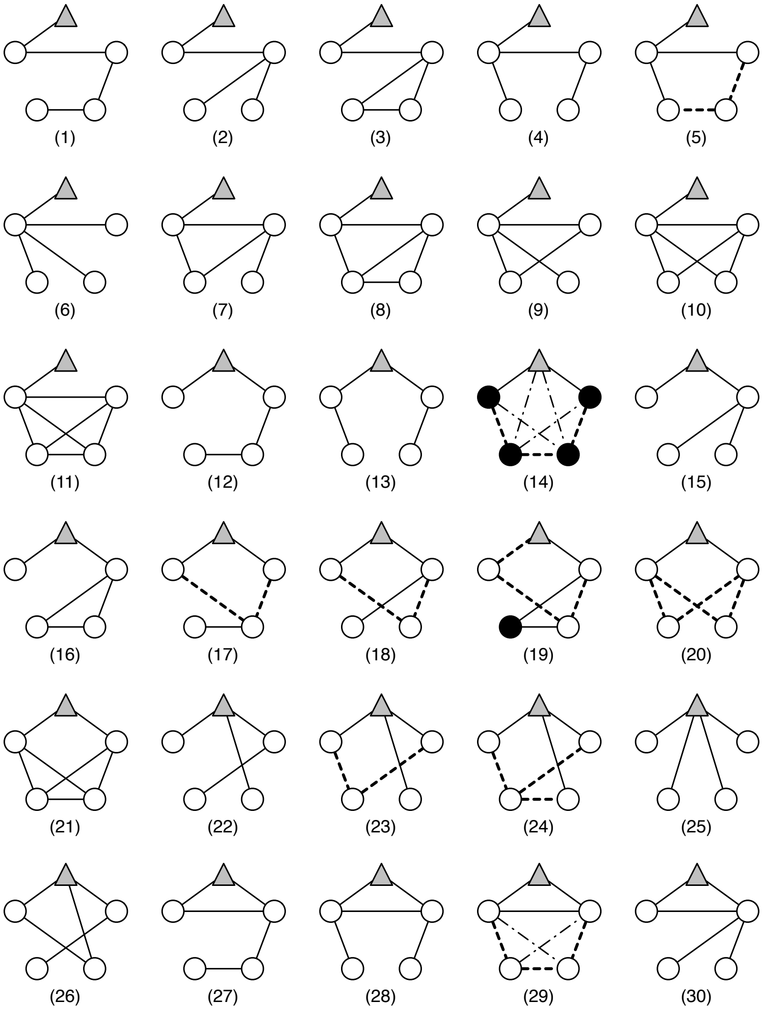


Figure 5.4 — Verification results for all topologies of one gateway and four nodes (Part I)

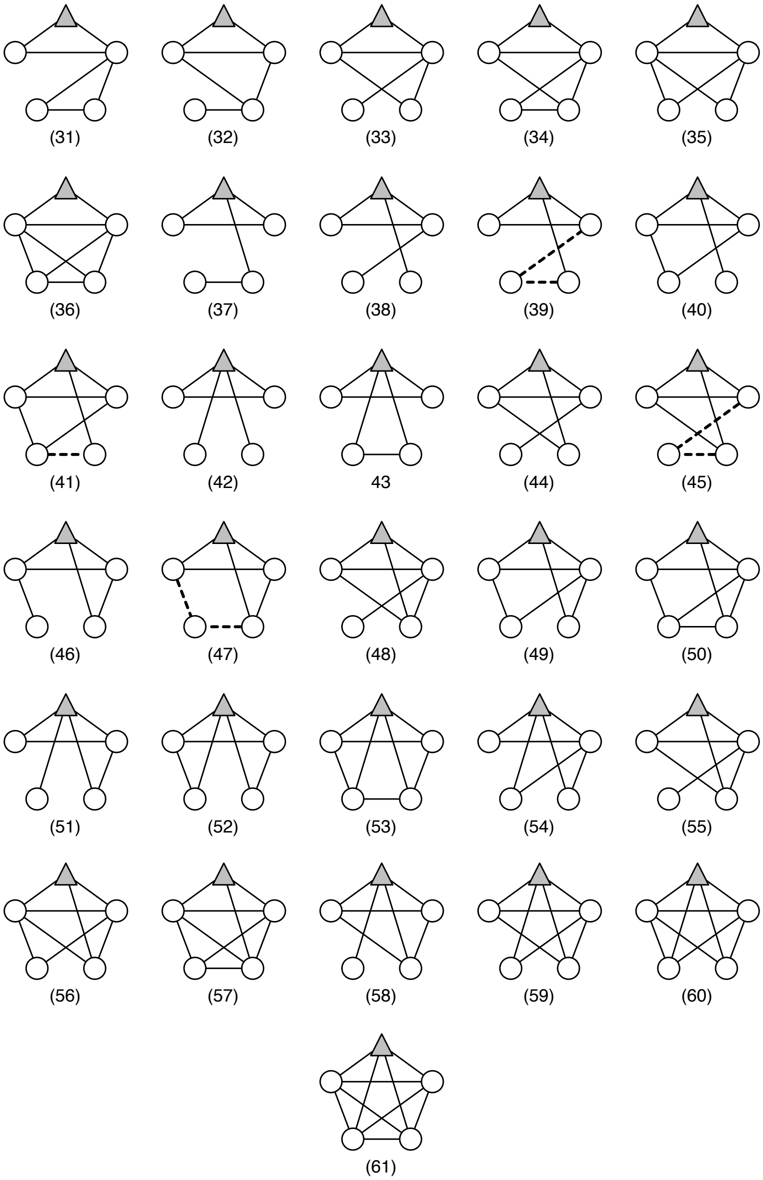


Figure 5.5 — Verification results for all topologies of one gateway and four nodes (Part II)

that its time slot is conflicting and reconsiders its time slot choice. This solution reduces the probability of the collision remaining undetected, however, it does not necessarily satisfy the collision property.

Since nodes are still able to reach the gateway node, the significance of this defect is negligible. However, it can have a clear influence on, e.g. localization mechanisms (Section 2.3.4), for which sensor-to-sensor communication is required.

2. **Sandwiched between collisions** — When a node is sandwiched between two nodes in collision before it obtains a time slot and there is no additional node to notify the collision, that particular node is not able to obtain a time slot e.g. Figure 5.4(14) and (19). A solution to this problem might be to use out-of-band signalling for collisions.

3. **Slot choice during conflict** — In some cases, it might occur that nodes reach state C, while their initiating node is (again) in collision. In Figures 5.4(19) this effect created two by collisions separated clusters. A solution to this problem is, e.g. to check the liveliness of a message path to the gateway node. When no path is available, nodes fall back to the initialization state I. This method is adapted in the LMAC protocol, discussed in Chapter 7.

In general, the properties are more often satisfied in abundant connected topologies.

By verification with the model checker Uppaal, we showed that nodes in a multi-hop network reach the continuous operation state, when there is at least one neighbouring *node* not in collision. Additionally, we investigated the capability of the protocol to solve time slot conflicts and concluded that collisions are solved when detected and reported. To guarantee start-up of all nodes, out of band collision reporting is required. We leave this subject to future work.

Chapter 6

EMACs and cross-layer optimization

In this chapter, we present EMACs, a medium access control protocol based upon the self-organizing algorithm for medium scheduling, presented in Chapter 4. The EYES medium access control (EMACs) protocol has been designed especially for resource constrained wireless sensor networks. In the MAC protocol, we combine schedule-based medium access control with an implicit backbone creation that identifies redundant wireless sensors. These redundant sensor nodes are not required in multi-hop forwarding and can thus save energy by following a sleep pattern. An inherent property of the schedule-based MAC protocol is that it collects information about local topology. This information is valuable for routing mechanisms. We exploit the MAC neighbour list and its ability to broadcast short messages efficiently to optimize the functioning of the EYES source routing (ESR) protocol. Multi-hop forwarding of messages, especially in dynamic environments, becomes more efficient, showing that cross-layer optimization is a prerequisite to build long lasting sensor networks.

6.1 Introduction

The technology that lets tiny and smart devices create their own networks, allowing them to transport sensor data while requiring little power and transmission range is potentially *the next big thing to happen* [45]. Recent advances in sensor technology, low

This chapter is a minor revision of the article "Prolonging the Lifetime of Wireless Sensor Networks by Cross-layer Interaction" in IEEE Wireless Communications, Vol. 12, 2004, [42] power analogue and digital electronics, and low-power radio frequency designs have enabled the development of these cheap, small, low-power sensor nodes, integrating sensing, processing and wireless communication capabilities.

Sensor nodes collaborate to be able to cope with the environment: they operate completely wireless, and are able to spontaneously create an ad hoc network, assemble the network themselves, dynamically adapt to device failure and degradation, manage movement of sensor nodes, and react to changes in task and network requirements.

There are many challenges in WSNs. In this chapter, we address in particular energy efficiency and the dynamics of a WSN. Where traditional communication protocol stacks assume an excess of resources and can spare the energy and memory to send many messages, the sensor nodes need to save on every ”bit” that is transmitted to ensure an acceptable network lifetime, as we argued in Section 2.5.2.

Some nodes in the WSN can be mobile, while others are fixed in walls or other immobile objects. In order to conserve energy, the sensor nodes are in low-power or off state for a significant amount of time. Communication during those periods is not possible. From the network point of view, these two aspects mean that the network topology is changing over time. Hence, the networking protocols must be able to cope with mobility and changes of network density.

Sensor nodes have to assist each other in forwarding their sensor readings to a data sink in the network. A routing protocol has the task to establish an efficient route for messages to travel in the multi-hop sensor network. Nodes along the route can suddenly fail or simply move away. In that case, the routing protocol has to defer the messages to a new route. The highly unpredictable environment makes this a challenging task.

This chapter presents a cross-layered approach for networking in wireless sensor networks, as part of the European research project EYES (IST 2001-34734). The approach addresses a self-organizing MAC protocol, that uses a (distributed) algorithm to decide the grade of participation of a sensor node to create a connected network based upon local information only, and a tightly integrated, efficient routing protocol.

The lessons learned in developing network protocols for wireless sensor networks in the last couple of years, show that using the traditional layered networking approach has several drawbacks in the resulting performance and efficiency of the system. Quite often, significant improvements are possible for the network protocols; yet they require a significant amount of information to be passed along the layers of the system. Although this approach allows, in principle, independency between the various protocols, it incurs a significant overhead in parameter transfer. Moreover, improvements performed in a specific layer can cause impairments and even be counterproductive for other layers.

Optimization can be more effective when taking into account the overall system, and with the use of all the available knowledge. When this information has to be distributed to other sensor nodes, the effect is even larger. A solution in which such information is piggybacked to other messages can limit this extra message exchange. During the development of various protocols and services (like localisation protocols), the lowest layers of our system (e.g. the MAC layer) were increasingly being used to pass information to these higher layers. The overall result of these developments has lead to the cross-layered approach, as described in this chapter.

6.1.1 Overview

The medium access protocol consists of a fully distributed and self-organizing TDMA scheme, in which each *active* node periodically listens to the channel and broadcasts a short control message. This control message is needed for medium access operation and is also used to piggyback various types of information at low energy costs.

Information in the control message is used to create a *maximal independent set* of nodes. This set of nodes creates a connected network and nodes in the set are *active*, while other nodes are *passive* and save energy by exploiting the infrastructure created by the connected network.

The control message is also used by the routing protocol to establish and maintain efficient routes in a dynamic topology. The routing protocol uses local topology information gathered by the medium access protocol and is therefore efficient in reestablishing routes when they become disconnected.

The presented approach is compared with the DSR routing protocol on top of the SMAC medium access protocol. Our approach to networking protocols for WSNs benefits clearly from the cross-layer interaction we are able to use. In a dynamic network topology, a network lifetime of at least 3 times the lifetime of a DSR and SMAC network could be reached in simulation.

In Section 6.2, we give an overview of related work. Section 6.3 discusses the design of the EYES MAC protocol, that is especially designed for WSNs and exploits the benefits of the cross-layer approach discussed in this chapter. We pay special attention to the decision mechanism that sensor nodes use in Section 6.4 to either actively take part in the network or to save energy by using resources of the backbone nodes in the network. The designed routing protocol is presented in Section 6.5 and Section 6.6 discusses simulation results. This chapter ends with conclusions.

6.2 Related work

6.2.1 MAC protocols for WSNs

Although the research field of WSNs is relatively new, some interesting studies of MAC protocols can be found in the literature (see Chapter 3). One of those protocols is the SMAC protocol ([111], Section 3.5.1), which we will use later on to compare results.

To refresh our memory, we summarize the basic ideas behind SMAC. SMAC recognizes two phases in transceiver usage of nodes: a listen and a sleep period. In the sleep period, nodes turn off their power consuming transceiver. After the sleep period, nodes wake up and listen whether communication is addressed to them, or they initiate communication themselves. This implies that the sleep and listen periods should be (locally) synchronized between neighbouring nodes. The protocol uses CSMA/CA during the listen period.

The SMAC protocol requires fine-tuning of the schedules, e.g. length of the sleeping intervals for different data traffic patterns and network densities. A dynamic topology results in many overlapping schedules, reducing the amount of possible sleeping periods of each node. Our approach attempts to reduce fine-tuning to a minimum. Especially, the connected active set incorporated into the MAC protocol eliminates the need to adjust the lengths of sleeping intervals in order to obtain, e.g., a connected sub network at all times or reduce the latency induced by routing through (partially) sleeping nodes.

6.2.2 Clustering

For the decision which nodes have to remain active to ensure an operational and connected network, we use ideas coming from clustering techniques. In the context of WSNs, clustering is mostly used to group the nodes for routing protocols. A designated node usually controls such a cluster.

Several authors, e.g. [7, 10, 34], focus on clustering schemes where the controlling nodes form an independent set in the wireless network. A set of local protocols that create and maintain a set of independent control nodes in face of dynamic environments, i.e. mobility, is given in [10].

Some ideas from these clustering algorithms are applied directly to the MAC-layer in order to create a backbone of the network consisting of the active nodes. In order to obtain an overall connected structure, so called bridges are introduced that are used to create connections between the controlling instances of the clusters. Our mechanism provides nodes with the ability to be idle and in a low power mode for a long period of time, yet nodes still have the possibility to quickly use the connected communication infrastructure, and creates an efficient and connected backbone.

6.2.3 Multi-hop routing

In a WSN, data generated by one or more sources usually has to be routed through several intermediate nodes to reach a destination due to the limited range of each node's radio. There has been significant work and research on routing mechanisms that deal with this problem [53], also in face of frequent topology changes. Related work for WSNs include directed diffusion [50], DSR [51], and AODV [82].

Directed diffusion [50] (Section 2.4.2) is a data-centric routing scheme which relies on local interactions between the nodes to create efficient paths for the data flow. Directed diffusion does not scale well when the nodes become mobile due to an end-to-end, four-way handshake protocol which has to be repeated every time the destination (sink) moves.

Dynamic source routing (DSR) [51] and ad hoc on-demand distance vector routing (AODV) [82] are routing protocols designed for dynamic networks. In this chapter, we compare our routing protocol with DSR (on top of SMAC). In brief, DSR works as follows. Each node stores known routes in its *route cache*. A new route is needed if information to the destination of a message is missing in this cache. The route discovery process is initiated to create the new route.

The network is flooded with *route request* messages. Each node adds itself to the route list in the message in order to build up possible routes to the destination, and transmits the modified message to its neighbouring nodes. When a request reaches the destination node, a route reply message is sent back once to the source. This is also done when an intermediate node has routing information to the destination stored in its cache. Nodes on the return path to the source will update their cache with the new route information.

6.3 EYES medium access control (EMACs) protocol

Sensors equipped with transceiver, processor and memory will be deployed by the millions. Hence the costs of a single smart sensor must be at a minimum. This does not only translate to scarce resources —like energy and memory– in the sensors, but also to the complexity of the hardware. During the design of the EYES medium access (EMACs) protocol, we assumed a single channel transceiver, which has three operational states: transmit, receive and standby. Typically, transmitting consumes more power than receiving and standby lies beneath the power consumption of receiving by a factor 1,000 or more.

In our research on energy efficient wireless sensor networks, we explore a medium access protocol whose operation is entirely distributed and localized. The main goal in designing a MAC protocol for WSNs is to minimize energy consumption, while limiting latency and loss of data throughput. Therefore, we have three modes of operation in our MAC protocol: active, passive and dormant mode.

When a node is in active mode, it will contribute to the network by taking part in forwarding messages to a destination and accepting data from passive nodes. Passive nodes on the other hand conserve energy by only keeping track of active nodes, which can forward their data and inform them of network wide messages i.e. broadcasts. The nodes in dormant mode put themselves in a low power state for an agreed amount of time or, for example, when their power source runs out of energy and has to be charged again using ambient energy, like light. We see the dormant mode as a mode that has to be initiated from the application side.

MAC protocols for WSNs must be able to cope with mobility and changes of network density. We assume that the change of network topology is low compared to network events and thus the mobility is assumed to be limited.

6.3.1 Time slot structure

The medium access protocol is based upon the medium scheduling mechanisms presented in Chapter 4. Time is divided into time slots, which nodes can use to transfer data without having to content for the medium and without having to deal with energy-wasting collisions of transmissions.

We assign one time slot to each active node and give this node control over this time slot. After the frame length, which consists of several time slots, the node again has a period of time reserved for it. Passive nodes (Section 6.4), in general, will neither control, nor claim a time slot. However, they are still able to communicate to the network by filling its requests to an active node. This allows significant energy conservation in the passive nodes and the lifetime of the network is largely extended, particularly if the role of active and passive nodes is changed over time.

To limit the number of time slots necessary in the network, we allow time slots to be reused at a non-interfering distance. But unlike in traditional schedule-based systems, the time slots in our protocol are *not* divided among the networking nodes by a central manager. In Section 4.3, it is explained how the wireless sensors can autonomously pick time slots with only local network knowledge.

A active node performs three tasks in the time allotted for its slot. Each time slot is divided into three sections (Figure 6.1): (1) communication request (CR), (2) traffic control (TC), and (3) the *data* section. We now discuss these three sections in detail.

- **Communication request (CR) section** — In the CR section, nodes can make requests to the node that is controlling the current time slot. The main purpose of this section is to allow passive nodes to notify an active node that it has data available, since passive nodes do not control a time slot themselves. The actual data transfer takes place in the data section.

  Nodes —that have a request to the time slot owner– apply listen-before-talk (Section 3.1.3) and pick a random start time in the short CR section to make their request. These messages are comparable to RTS messages in SMAC. Communication in this section is not guaranteed conflict-free. However, since the data rate in WSNs is generally low, collisions are expected to occur seldom.

  A special request is a node announcement (NA). This request is used by newly added nodes in the network to announce their presence. Again, this is particularly useful for passive nodes.

  Nodes that do not have a request for the current slot owner, keep their transceiver in a low power state during the entire CR section. However, the time slot owner listens to incoming requests during the entire section.

- **Traffic control (TC) section** — The owner of a time slot in every frame broadcasts a short TC message in the time slot, indifferent whether a request was filed or not. The beginning of each TC message is carefully timed by the time slot owner and the message is virtually the heartbeat of the EMACs protocol. Consequently, all (active) nodes put effort in receiving these messages, since they contain control information, such as the identifier of the time slot owner, possible acknowledgement to a request as well as the occupied slots bit vector (Section 4.3.3) and the current position in the frame.

  Routing protocols that allow messages to be routed over the ad hoc network, typically require the knowledge of the actual topology in order to efficiently route the packets over the network and to deliver them at the destination. By listening to TC sections of neighbouring nodes, nodes have knowledge of the *local topology*. This assists routing and reduces the number of routing messages in the network. A special portion of the TC section is reserved to efficiently transmit the short broadcast messages that are generated by the routing protocol.

  The time slot owner also indicates in its TC message what communication will take place in the data section. If a node is not addressed in the TC section nor its request was approved, the node will then resume in standby state during the entire data section. The TC message can also indicate that the controlling node is about to send a broadcast message.

  When a time slot is not controlled by any node, all nodes will remain in sleep state during that time slot.

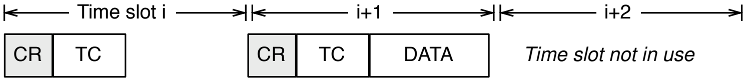


Figure 6.1 — Time slot structure of EMACs

- **Data section** — The remainder of the time slot is reserved for the data section. In this section, the actual transfer of higher layer packets takes places. The format and length of the data section is left to higher protocol layers. Its length, however, is bounded by the CR section of the next time slot.

  The owner of the time slot decides what action to take in the data section (and indicates this in its TC). It can either transmit a message to one specific node, broadcast a message to all neighbouring nodes or receive a message, e.g. from a passive node. When none of these actions is appropriate, the time slot controller puts its transceiver into low-power mode.

Due to the scheduled nature of the EMACs protocol, nodes can switch their transceivers to low-power mode at times they are not involved with communication. Note that the CR and the TC sections are short compared to the data section. Passive nodes only listen to one TC per frame, since they only need to maintain synchronization with a single active neighbour to enable message transfer. This allows passive nodes to conserve even more energy.

From time to time (dependent on the expected mobility in the network), active nodes give up their time slot and re-execute the algorithm to pick a time slot. This prevents collisions when active nodes travel through the network and meet another active node that has claimed the same time slot. Additionally, nodes can notify collisions to active nodes by sending a special request during the CR section of the conflicting time slot.

6.4 Active and passive nodes

In the EMACs protocol, we separate the set of wireless sensors into active and passive nodes. The idea behind the introduction of active and passive is that WSNs are densely deployed networks (Chapter 2) and thus that redundancy is inherently high. Hence, many nodes can become passive and can conserve energy. This extends the lifetime of the wireless sensor network, while it remains sufficiently operational to carry outs it sensing tasks.

In this section, we present a (local) algorithm that is used to identify the nodes that actively participate in the networking tasks, such as routing, and nodes that can conserve energy by entering the passive mode. The decision algorithm is based upon clustering techniques, in particular the creation of a maximal independent set. The control information for the algorithm is small and fits easily in the TC section of EMACs.

Since passive nodes do not actively participate in the routing process of the network, the set of active nodes is required to form a connected set. In this way, each node of the network can eventually be reached by an ad hoc routing process (Figure 6.2). The set of active nodes is later on referred to as the *connected active set* of nodes. Nodes that need to be active to ensure the above properties are contained in this set.

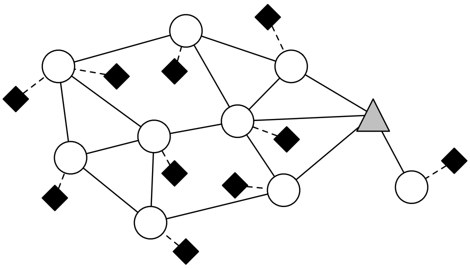


Figure 6.2 — Connected communication backbone in the active set. Black nodes realized their redundancy and switched to passive mode

In addition, the active set allows for heterogeneous node platforms in the WSN, e.g. μNode and SmartTag platforms (Appendix A). The SmartTags have considerably less resources and are typically not able to store necessary routing tables and message queues. The SmartTag platform is therefore not suited to participate in networking in the sense that they actively can participate in creating a connected mesh structure. In relation to EMACs, SmartTags are better suited to fulfill the role of passive nodes and rely on active nodes for delivering messages to their final destination. Consequently, the network functionality in SmartTags can be very lean. More resourceful platforms (e.g. the μNode) implement EMACs and active set completely and decide upon their active/passive role according to local requirements in the WSN or application requirements.

6.4.1 Roles of active nodes and their encoding

We define four roles for active nodes, i.e. nodes that control a time slot and participate in networking [80]:

- **Anchor nodes** — These nodes form the base set, i.e. a maximal independent set. This is a dominating set and no two anchors are neighbours. The maximality of the independent set of anchors ensures that anchors are not more than three hops apart [79].

- **Bridge nodes** — These nodes connect anchor nodes and ensure that the backbone forms a connected structure. There are two types of bridge nodes: (1) direct bridges that receive two anchors or more and create a connection between these anchors, and (2) distributed bridges that connect two nearby anchors at (maximum) distance 3.

- **Undecided nodes** — Indicating the role of nodes that did not decide yet, e.g. newly added nodes.

- **Non-member nodes** — A node that decided to be active based upon requirements from higher layers.

The anchor nodes are locally created to cover the network so that no two anchor nodes are direct neighbours. If an anchor node can reach (via other active nodes) all anchor nodes that are at most three hops away [79], the entire set of active nodes is connected. To achieve this, *bridge* nodes are introduced. There are two types of bridging nodes. A node that receives the TC sections of two or more anchor nodes is called a direct bridge. If two intermediate nodes are needed, these two nodes form a distributed bridge. Note that the construction of the active set results in a mesh-like backbone in the network. This does not introduce unnecessarily long message routes. Routing protocols are thus not hindered by the active set.

These roles are efficiently encoded in the *anchor identification* (AID) field (Table 6.1) that is broadcasted by active nodes in their TC section. Hence, by listening to all transmitted TCs, an undecided active node is able to assess the decisions of surrounding nodes in order to decide whether it is required to stay active for ensuring a connected network, or whether it can conserve energy in the passive mode.

The first bit of the AID field is always set to 0 (assuming that this bit is not actually used in the node's ID). This is done to identify bridges, which have a leading 1 in the *AID* field. Also, the value given in the AID is not mistaken for a possibly non-existing node ID. For the created structure, not much difference exists between these types of bridge nodes; they are encoded and used in the same way.

Nodes that are not part of the connected active set (passive nodes), but participate in the network by owning a TC section, are identified by having an AID corresponding to the neighbouring anchor node with lowest ID. This encoding also helps in identifying distributed bridges.

Table 6.1 — Encoding of the active node roles

Role	Anchor identification (AID) field in TC
Anchor node	ID of node
Bridge	$(ID_{Anchor1}$ XOR $ID_{Anchor2}) +$ b100...
Undecided Active	0
Non-member	$\min(ID_{Anchor1}, ID_{Anchor2}, \dots)$

6.4.2 Local decision algorithm

Each node that enters the network, e.g. by waking up or being deployed has to decide whether it is needed as part of the connected active set. This is achieved by the following algorithm. Additionally, this decision process is performed when a change in the local topology given by the active nodes occurs. This is detected from the local neighbour table that is kept up-to-date by receiving TC sections of active nodes.

A schematic overview of the decision algorithm is presented in Figure 6.3. Next, we present the individual steps and decisions in more detail.

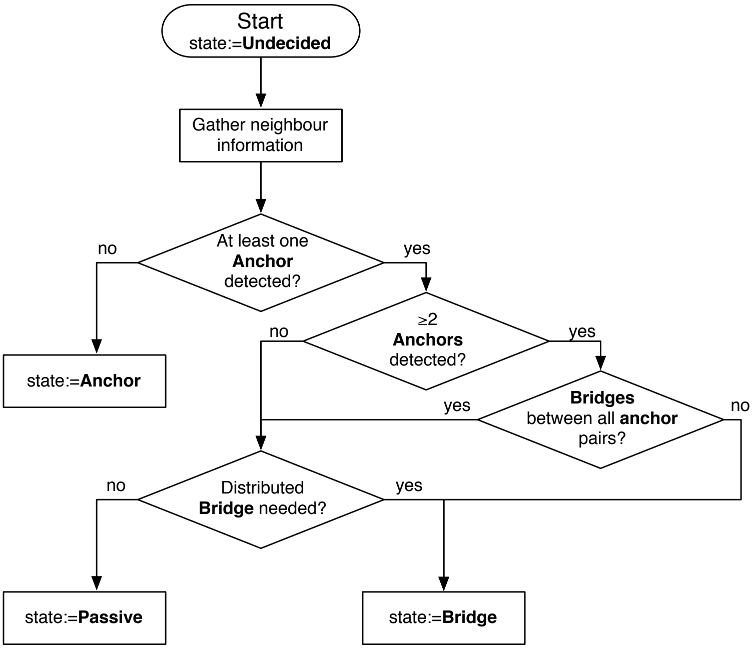

Start
state:=**Undecided**
Gather neighbour information
At least one **Anchor** detected?
no
yes
state:=**Anchor**
≥2 **Anchors** detected?
no
yes
Bridges between all **anchor** pairs?
yes
no
Distributed **Bridge** needed?
no
yes
state:=**Passive**
state:=**Bridge**

Figure 6.3 — Local active/passive decision algorithm

- **Anchor decision** — When there are neighbouring anchors, the node cannot become an anchor itself. However, if there is no anchor identified, the lowest node ID criterion is used to elect an anchor. For this, a node checks whether it is the undecided active node with the lowest ID in its neighbourhood and becomes anchor node when this is the case. Else, it waits for undecided nodes with lower IDs to decide first. This follows the idea behind lowest ID clustering [34]. Note that other decision parameters can be used instead of ID of the node, however ties must be resolved.

- **Bridging decision** — When there are two or more anchor nodes in the neighbourhood, a node checks whether there is already a (direct) bridge in the neighbourhood connecting pairs of anchor nodes. It can check this by locally calculating the XOR of the anchor IDs and comparing it against AID fields of surrounding nodes. In the case that a pair of anchors exists that is not yet connected, the node performs the direct bridge function and updates its AID.

- **Distributed bridging decision** — When a node decides not to become an anchor or a direct bridge, it still might turn into a distributed bridge. For this decision, the node must first discover whether there exists an anchor node at hop distance three of a surrounding anchor node. The node therefore transmits in its AID the lowest anchor ID (i.e. it mimics a non-member node). In this way, a distributed bridge connection becomes possible. Another node responds by becoming a (direct) bridge and a connection between the anchors is established.

- **Become passive** — A node that has come to the decision that it is not needed in the connected active set, does not drop out of the process immediately. For the next frame, it transmits its neighbouring anchor with the lowest ID for distributed bridging detection. If after that no change in the neighbourhood is detected, it can become a passive node or it can maintain its non-member state, if this is desired by higher protocol layers.

Note that if there are undecided nodes, the undecided node with the lowest ID in the neighbourhood is always able to decide on its role other than undecided. The undecided role is thus only a temporary one.

Obviously, a node that participates in the network as part of the connected active set consumes more energy than passive nodes. Therefore, the principle of role rotation is supported in our scheme. An active node can drop its status and become undecided. Surrounding nodes will detect this and adapt by creating new anchors or bridges, if needed for connectivity. Especially in a dense network, many nodes are capable of performing the connecting duties of bridges. In our approach, only a few bridges actually have to remain active, as other nodes in the area realize their redundancy by the AID field [79]. Thus, overall we obtain a connected dominating set given by the active nodes that uses only few nodes.

The active set has an important advantage in densely deployed networks. At initial deployment of the network, all nodes try to obtain a time slot and take the undecided active role. Recall that the frame size is network-wide fixed and needs to be tuned to the maximum connectivity in the network (Section 4.6). As a result, many time slots per frame are required in dense networks and, consequently, message delay is significant. However, when using the active set, the required time slots can inherently be reduced, because many nodes realize their redundancy, give up their time slot and become passive.

6.5 EYES source routing (ESR) protocol

In Section 2.4.1, we discussed the on-demand EYES source routing (ESR) protocol, which enables dynamic, self-starting, multi-hop routing to be established when a source sensor node requires to send a data message [107, 42]. In this chapter, the ESR protocol is used by *active* nodes. Passive nodes simply deliver their data to active nodes, which then take care of the routing of the message to its final destination.

Cross-layer optimization for ESR using EMACs consists of the following two aspects:

- **Efficient transmission of maintenance messages** — The operation of ESR relies on maintenance messages, e.g. route requests. These messages are typically short and need to be broadcasted to all surrounding nodes. These messages can be piggybacked in the TC section, which already fulfils the required broadcasting. It can be easily seen, that this saves on overhead as opposed to sending (broadcast) data messages containing the short routing messages.

- **Neighbour list** — The ESR protocol tries to repair routes efficiently. For this functionality, the neighbour list of EMACs can be exploited, since changes in local topology —for example, due to mobility– are captured in this table. Detecting such events is generally quite difficult and time consuming. In our approach we are able to detect this very efficiently and quickly, because our MAC protocol already has this kind of information present. Within one time frame the MAC protocol notifies the routing layer if any of the monitored neighbours is no longer available in the vicinity. This increases the possibility of successful local and fast route recovery and self-healing.

Otherwise, the protocol is left unaltered.

6.6 Simulation results

For the simulation of our combined cross-layer optimized networking protocols, the OMNeT++ (`http://www.omnetpp.org`) discrete event simulator, together with a framework for a mobile, wireless network, is used. We compared the protocols presented in the previous sections with DSR and SMAC. The same network setup is used for comparing the two implementations of medium access and routing protocols.

In the simulator, a physical layer with energy model is implemented to record the sending and receiving energy consumption of the transceiver. Additionally, switching between sending and receiving takes time and consumes energy, which is also considered in the simulation. The respective data for the transceiver are taken from an RFM TR 1001 [89], which is also used in our prototype sensor nodes [39] (Table 6.2). Although our prototype design can adjust its transmission range, we only consider the transmitting power to be fixed to a high level, which yields an approximate coverage radius of 150 m.

In the simulation, 45 sensor nodes are randomly placed in a rectangular area of 5 by 5 times the radio range. Five of them are chosen to be source nodes, which actually produce sensing data. The length of a data packet is 16 bytes and the data rate is

Table 6.2 — Transceiver data (RFM TR 1001) [89]

Parameter	Value
Energy consumption transmit	21 mW
Energy consumption receive	14.4 mW
Energy consumption standby	15 μW
Switch time transmit/receive	518 μs
Switch time receive/transmit	12 μs
Switch time standby/receive	518 μs
Switch time standby/transmit	16 μs

varied in different simulation runs. One (active) node is designated to be the data sink, which receives the data from these source nodes.

The nodes move in the area according to the random way-point model (RWP) with random speed (2-10m/s) and waiting times (10-30s). A node that has reached its destination does not immediately pick a new way-point, but waits for a given period of time before moving again. In this way, a mix of moving and static nodes is achieved.

We use the *network lifetime* as the measure to evaluate the performance of our cross-layer optimized protocols. In wireless sensor networks, the metric of actual interest is not the transmission energy of individual packets, but the total operational lifetime of the network. Network lifetime measures the amount of time before a certain percentage of sensor nodes run out of their battery power.

Both the data sources and sink have an infinite energy budget, so they will not affect the network lifetime. During the simulation, when 30% of the normal sensor nodes are depleted of battery, the whole network is considered to be down.

Figures 6.4 and 6.5 show the network lifetime of our approach and the reference DSR and SMAC, under different network loads. Note that the graph is normalized to SMAC and DSR in the static scenario. It is shown that our scheme prolongs the lifetime of the network significantly in the mobile scenario. A lifetime of at least 3 times the lifetime of DSR and SMAC could be reached.

Although our protocols were designed to be efficient in dynamic networks, we also compared the protocol performance for static networks. In that scenario our scheme extends the lifetime 25% to 50%.

It is interesting to see that the lifetime in the case of SMAC and DSR is almost not dependent on the message frequency. This can be explained by the fact that the nodes use their receiver anyhow in the time interval they are awake. The additional energy, which is necessary to exchange messages at relatively large intervals in our simulations, is negligible compared to the energy used in the listen period. In fact we would expect that the lifetime of the network gets larger to some extent when the message rate is enlarged, because of the effect that neighbouring nodes of the transmitter and receiving nodes will switch their transceiver to standby to prevent energy-waste in overhearing.

Our cross-layered approach performs better in scenarios where the nodes are mobile than it does in static cases. This can be explained by the fact that the roles *active* and *passive* are not changed in the latter case, while in the mobile case the dynamic changes in network topology force the nodes to reconsider their role. This leads to better and more even energy consumption between the nodes, which results in a longer network lifetime.

In contrast to our protocols, SMAC and DSR perform better in the static case than in the mobile case. This is clearly due to the overhead in routing; in the static scenario, routes have be to established *only once*, while in the mobile scenario routes have to be updated regularly.

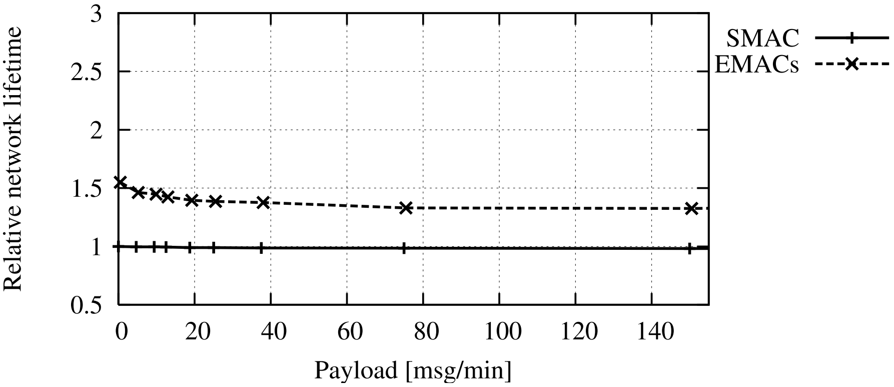


Figure 6.4 — Comparison of network lifetime of SMAC and EMACs. Static scenario (Simulation)

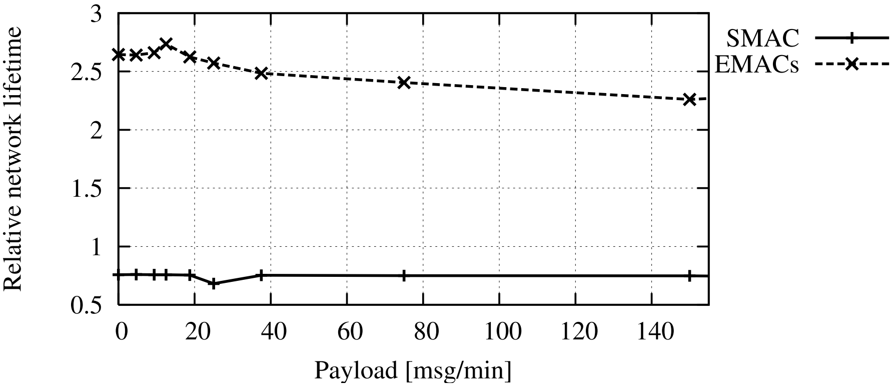


Figure 6.5 — Comparison of network lifetime of SMAC and EMACs. RWP scenario (Simulation)

6.7 Implementation

EMACs, in combination with the active set, has been implemented as a final demonstrator of the European EYES project (IST 2001-34734). The EYES node, as described in [39], has been used as prototype wireless sensor. However, to demonstrate multi-hop communication in a limited space, the receive sensitivity of the platform was significantly reduced. The effective transmission range after the alteration was 4 meters.

In total, 100 nodes were deployed in a rectangular grid (10×10 nodes in 20×20 m^2). The nodes implemented EMACs and the active set. The EMACs frame consisted of 32 time slots and each time slot had a duration of 100 ms. To allow easy observation of the node states concerning the active set, different coloured LEDs were used for active and passive. Additionally, distinct LEDs indicated the path messages took to travel to one central node. The memory usage of the protocol implementation was little: 5 kB of program memory (i.e. 8% of total) and 200 bytes RAM.

The demonstrator showed that (1) EMACs is able to successfully organize communication, (2) active/passive roles are taken by nodes, (3) passive nodes are able to use the backbone created by active nodes, and (4) energy consumption of passive nodes is 100 μA on average (energy-consumption of LEDs excluded). For complete results we refer, to the final EYES project deliverables [30].

6.8 Conclusion

The lessons learned in developing network protocols for wireless sensors show that using the traditional layered networking approach has several drawbacks in the resulting performance and efficiency of the system. Quite often, significant improvements are possible for network protocols. In this chapter, we showed that cross-layer optimization is indeed a useful approach for WSNs.

We discussed EMACs, a schedule-based medium access protocol, of which the operation is not dependent on a central manager or base stations. The nodes in the network are capable of choosing their own time slot, based upon local information only. Nodes in the network can communicate with each other collision-free.

Not every node is needed to actively participate in communication in the network for global connectivity. Hence, the medium access protocol allows some nodes to be *passive*. These passive nodes save energy by not controlling a time slot, but make use of a backbone in the network that is formed by *active* nodes. In this way, the medium access protocol overhead is greatly reduced for passive nodes. Passive nodes can communicate with active nodes, although this communication is not guaranteed to be collision-free. This chapter presents a simple, yet effective algorithm for nodes to make the decision between the medium access protocol states active and passive. Again, this decision is only based upon local information. Results in [79] show that the number of active nodes can be quite low. Consequently, this reduced the number of time slots required per MAC frame.

Routing protocols benefit from local topology information that is already present in the medium access protocol, in the case of EMACs. Only active nodes assist each other in forwarding messages to a destination that cannot be directly reached by the source node. An initial route is established by flooding of the active nodes of the network by the source. This is very energy consuming, yet inevitable. In this stage, the benefits of active and passive network participation of nodes already becomes clear. When a route gets disconnected, e.g. due to highly dynamic topology of a WSN or due to energy depletion of nodes along the route, the protocol is able to efficiently re-establish routes between nodes.

We compared our cross-layer optimized networking protocols with traditional protocols for WSNs: SMAC and DSR. One of the key issues in WSNs is the network lifetime. Simulations show —in equal network configurations, message frequency and size assumptions– our cross-layered approach delivers a longer network lifetime, especially in the case when nodes are mobile. In the static case, the difference is smaller, which is mainly due to the fact that the routing protocols have to establish a route only once. Our protocol has standard a small amount of data reserved for route updates and in the static case this space is wasted.

Our approach clearly benefits from active and passive modes in the medium access protocol. In the mobile case, the roles are changed often, resulting in high network lifetimes. In the static case, nodes keep their medium access mode until their energy is depleted. Therefore, the role should be reconsidered every now and then. We leave the role change mechanisms to our future work.

A cross-layered optimization is in our eyes a good solution to reach the target of highly energy-efficient WSNs.

Chapter 7

Lightweight medium access control protocol for WSNs

The lightweight medium access control (LMAC) protocol has been designed especially for WSNs. The protocol is based upon the general medium scheduling mechanism as has been presented in Chapter 4. The key features of the protocol are its self-organizing and routing capabilities.

Latency is one of the main concerns in schedule-based MAC protocols. We classify sources of latency and propose several methods to reduce latency, which are based on adapting the self-configuring time slot choosing mechanism. Additionally, we present a framework for making the protocol adaptive to predictions of data traffic volume.

7.1 Introduction

Sensors, equipped with RF transceivers and processing units —which allow local pre-processing of the sensor readings– form the first step in acquiring a high level description of their physical environment. The devices creating the wireless network are typically powered by batteries and are deployed on a large scale. Therefore, one of the most interesting challenges is the organization of wireless communication in these massive networks, where nodes collaborate and push data towards dedicated gateway nodes. To ensure long-lived WSN applications, it is key issue to minimize energy wastage at MAC level —e.g. due to collisions of messages and idle listening– while limiting latency and loss of data throughput.

This chapter is based upon the following publications [40, 41, 43]

In the previous chapter, the EYES medium access control (EMACs) protocol has been discussed. In this chapter, we present a derived protocol –the lightweight medium access control (LMAC) protocol. Again, it is based on the general medium scheduling mechanism as presented in Chapter 4.

In Chapter 2, we introduced three different communication patterns and we argued that sensor readings are most often travelling to the gateway nodes in the network. The LMAC protocol includes efficient routing of messages to designated gateways in the network at low additional costs.

The main properties of the LMAC protocol are that it is self-configuring —even when nodes are mobile–, robust against high peak loads and energy-efficient [59], ensuring a long-lived network. The protocol has a low complexity and can be implemented using a small memory footprint.

This chapter provides several design aspects of the LMAC protocol. First, the protocol is described (Section 7.2). As for any schedule-based MAC protocols, consensus about timing is of vital importance for LMAC. This topic is highlighted in Section 7.2.4. Then, we discuss LMAC's performance compared with state of the art WSN MAC protocols in Section 7.3.

In Section 7.4, we investigate the effects of different waiting times on the network setup time and the number of energy-wasting collisions. Next, we target latency – one of the main concerns of schedule-based MAC protocols. In Section 7.6, LMAC is made adaptive to the expected data volume that has to be transported by the network. Finally, the chapter ends with conclusions and recommendations for future work.

7.2 Protocol organization

The LMAC protocol is based upon scheduled-based access as has been presented in Chapter 4. To repeat briefly, each node periodically gets a time interval in which it is allowed to control the wireless medium according to its own requirements and needs. Outside this interval, nodes are notified (i.e. by receiving shortly) when they are intended receivers. Whenever a node is not needed for communication, it switches its transceiver to standby and is, therefore, able to conserve energy.

Schedule-based MAC protocols have the advantage that nodes are never using their power consuming transceivers while not needed. Therefore, this type of medium access has good prospect in being energy-efficient. Since each node gets its own turn in using the medium, there will be little collision of messages and idle-listening which are in other types of MAC methods —such as CSMA– major reasons for energy waste [111].

Of course, any protocol requires some overhead to function properly, as is the case with schedule-based MAC protocols. Nevertheless, the philosophy of the designed MAC protocol is to keep it simple, remove handshaking mechanisms and thus save energy consumption on physical layer overhead and to make it robust to function completely autonomously in a distributed environment.

7.2.1 Frames and time slots

In the schedule-based LMAC protocol, time is organized into *time slots*, which are grouped into *frames*. Each frame has a fixed length of a (integer) number of time slots. The number of time slots in a frame should be adapted to the expected network node density (Section 4.6) or system requirements. In our LMAC protocol implementations, we consider in general 32 time slots per frame with a total length of one second. Note that, when setting these network wide parameters, a trade-off must be made between message latency and network lifetime.

The scheduling principle in the LMAC protocol is very simple (hence its name *lightweight*): every node gains control of one time slot to carry out its transmission. The schedule is repeated every frame, i.e. a node transmitting in —for example– time slot 7, uses the same time slot in the next frame. The heart of the protocol is the scheduling mechanism presented in Chapter 4. When a node has some data to transmit, it waits until its time slot comes up, addresses a neighbouring node (or multiple) and transmits the packet without causing collision or interference to other transmissions.

In order to be capable of receiving messages, nodes always listen at the beginning of time slots of other nodes to determine whether they are addressed either by node ID or by broadcast address. In the LMAC protocol, nodes can receive multiple data messages per frame, but are only allowed to transmit once per frame. A higher layer in the protocol stack should combine data fragments into one message for transmission whenever possible.

A time slot is further divided into two parts of unequal length (Figure 7.1): *control message* (CM) and *data message* (DM). Between the CM and DM is a small gap, which allows the MAC layer to process the just received CM. In the following sections the control and data messages will be discussed in detail.

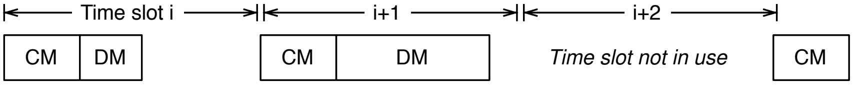


Figure 7.1 — Time slot contents of the LMAC protocol. Note that the data message (DM) does not have a fixed length and is even omitted when a node does not have any message to send

7.2.1.i Control message

In Table 7.1 the contents of the CM is specified:

- **Identification** — We assume that every node in the network has an *identification number* (ID). This number is transmitted at the beginning of the CM. By listening to all CMs, a node always has knowledge about its neighbouring nodes. This is valuable information for routing protocols and helps reduce route setup time or route recover time. The information can also be used for node localization protocols.

- **Current Slot** — This number reflects the position of the time slot in the current frame and is used by new (i.e. unsynchronized) nodes in the network.

- **Distance to Gateway** — With this number, nodes advertise their hop distance to the closest gateway in their vicinity. This is used for synchronization (Section 7.2.4) and simple, shortest path routing of messages to the gateways (Section 7.2.3).

- **Occupied Slots** — In this bit vector, nodes keep track of controlled time slots around them and share this information with neighbours. Each position in the *occupied slots* bit vector represents a time slot. When a node receives a CM successfully or detects energy in the wireless medium, it updates the vector by setting a logical 1 at the position of the time slot, otherwise a 0 will be inserted. A node inserts a 1 at the time slot it controls.

  This bit vector is of vital importance to the LMAC protocol, because it lets nodes know which time slot does not interfere in a two hop radius, and thus can be used. This procedure is discussed in Section 4.3.

- **Collision in Slot** — During network setup or changing network topology, two or more nodes that control the same time slot might come into range. The CMs will be transmitted at the same instance of time, resulting in a collision. This collision can be detected by neighbours of the nodes. These nodes use the *collision in slot* field to indicate that a collision has occurred. If nodes are notified that their transmission collided, they immediately stop controlling their time slot and return to the process of finding a free time slot. We discussed this topic in Section 4.5.

- **Destination Address** — This field is used for addressing neighbouring nodes. The protocol also supports broadcast messages (i.e. a message that should be received by all neighbours). When a node is addressed in this field, it will also try to receive the DM that follows the current CM.

  When nodes discover —by listening to the CM– that they are not the intended receiver, or that the transmitting node simply has no data to transmit, they immediately turn off their power-consuming receiver (after the CM) and wait until the next time slot comes around. This allows the protocol to be energy-efficient, i.e. no idle-listening.

- **Acknowledgement** — In many MAC protocols, data messages are acknowledged by the recipient. For this purpose, the acknowledgement bit vector has been added to the LMAC protocol. Nodes acknowledge correctly received data messages by using this field. Nodes keep track in which time slots they received data messages correctly. If so, a logical 1 is placed at the position of the appropriate time slot in the *acknowledgement* field. If no message or an incorrect message was received, a logical 0 will be placed. Knowing their own time slot, nodes can derive from this information whether their messages have arrived correctly.

> A special feature of this acknowledgement method is that the recipients also acknowledge broadcast messages. This can greatly improve the reliability and efficiency of those messages.

A node *always* starts its time slot by sending out a *control message* (CM), even if it does not have any data to send (comparable to the beacon message in IEEE 802.15.4 MAC specification, Section 3.4). One could argue that transmitting a CM imposes overhead, but doing so provides many advantages. In addition to (collision-free) addressing other nodes, the CM is also used for maintaining (relative) synchronization, for allowing distributed operation of the MAC protocol, for maintaining shortest path routing to gateways and for neighbour discovery.

Table 7.1 — Contents of the control message (assuming 32 time slots in a frame)

Description	Size [bit]
Identification	16
Current Slot	5
Distance to Gateway	8
Occupied Slots	32
Collision in Slot	5
Destination ID	16
Acknowledgement	32
Total	**114**

7.2.1.ii Data message

This part contains the *data* that a node needs to send. The DM has a fixed maximum length (typically 255 bytes). Typically, the actual length of the packet is transmitted at the beginning of the DM together with a type indication of the message, its source and its final destination. However, formatting of the data message is up to higher layers in the protocol stack, which can include combining of multiple data fragments to be efficiently transmitted in one DM.

Addressed nodes keep their receiver on until the data packet has been completely received and then switch it to standby for the remainder of the time slot.

7.2.2 Unused time slots

It might also occur that a time slot is not used by one of the neighbours of a node. Since the topology of the network might change over time due to mobility of nodes or due to wake-up of sleeping nodes, et cetera, nodes continuously sample the wireless channel for transmissions at the beginning of time slots. For energy-efficiency reasons this sampling must be as short as possible. It can also be a trade-off to let nodes only periodically detect new neighbours. This clearly slows the response of the network to changing topology, but results in better energy-efficiency. This problem is left to future work.

When there are no more free slots (i.e. the local connectivity is higher than expected), the node remains in initialization state, periodically monitoring frames to find an empty time slot. In this state, energy-efficient techniques can be used for sampling of the wireless radio channel, such as those described in [38] and [83].

7.2.3 Routing to gateway(s)

For reporting to designated gateways in the network, we use ideas from [67] to efficiently route the data messages. Each node keeps track of its hop-distance to a designated gateway node (Figure 7.2) and broadcasts this information efficiently in its control message. This virtually creates a gradient towards the closest gateway in the network.

When a message arrives, either generated by the node itself or received from another node, the node will look in its neighbour table for a neighbouring node that is closer to the gateway than itself and will pick this node as the destination for the message. In case of multiple neighbours that are closer to the gateway, the node will randomly pick one from the candidates. Eventually, the messages arrive at the gateway.

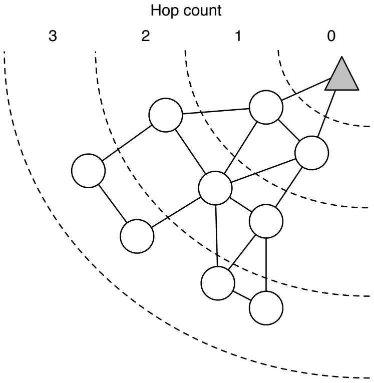


Figure 7.2 — Nodes keep track of their hop-distance to the closest gateway. This information is used for efficient routing to gateways

Note that we reuse the routing parent/child structure for synchronization (Section 7.2.4).

7.2.4 Time synchronization

A key issue in scheduled medium access is that it needs a common sense of timing in order to create a long-lived network. Without precise (local) synchronization, nodes have to use long guard intervals and time outs to ensure that receivers are ready when transmitters start transmitting, wasting valuable energy (see Figure 7.3 for definition of these intervals). The more accurate synchronization is, the smaller tolerated timing differences can be, and thereby wasting less energy.

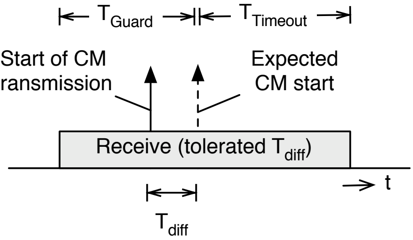


Figure 7.3 — Nodes observe a guard period to tolerate timing difference of the owner of a time slot

To demonstrate the energy costs of guard intervals and time outs, we present the following energy consumption model per frame of the LMAC protocol (see Table 7.2 for explanation of symbols):

$$E_f = E_{tx} + d\,(E_{Guard} + E_{rx}) + (N_{timeslots} - d - 1)(E_{Guard} + E_{Timeout}) \quad \text{per frame} \quad (7.1)$$

Equation 7.1 contains three components: (1) transmission of CM, (2) energy consumption during receive of CM (assuming $T_{diff} = 0$), and (3) energy costs during empty time slots. In this model, we neglect DMs entirely i.e. nodes do not transmit or receive data messages. Also, the energy consumed in the sleep mode of the transceiver is neglected. Table 7.2 gives an overview of the energy costs per LMAC frame.

Table 7.2 — Clarification of symbols used in Equation 7.1 and realistic values. For $T_{Guard} = T_{Timeout} = 0.5$ ms is assumed

Symbol	Description	Value	Percentage of E_f
E_{tx}	Energy costs of transmitting a CM	0.21 mJ	6 %
E_{rx}	Energy costs of receiving a CM	0.22 mJ	65 %
$E_{Timeout}$	Energy costs of time out interval	0.02 mJ	11 %
E_{Guard}	Energy costs of guard interval	0.02 mJ	17 %
$N_{timeslots}$	Number of time slots	32	
d	Connectivity of the node	10	

In fact, the energy costs of the guard interval return $N_{timeslots} - 1$ times in the above equation. When plugging in the realistic numbers obtained from our prototype hardware platforms (Appendix B), we find that (potential) idle listening during the guard interval consumes roughly 17% of the total transceiver energy consumption per LMAC frame in these settings. More accurate synchronization allows for reduction of the guard interval and time outs. Consequently, the energy consumption of a node will drop. This stipulates that accurate synchronization positively adds to the lifetime of the wireless sensor network.

7.2.4.i Timing experiments

In the relatively simple hardware of wireless sensor nodes, the sense of time is typically derived from crystal oscillators. These oscillators suffer from inaccuracies, like temperature drift and jitter. In the manufacturing process small dimension variations exist between the individual crystals. This results in slightly unequal resonance frequencies. Typically, the difference with the nominal frequency —offset– is within ±20 ppm.

Another factor that plays a role in timing is the processor itself [71]. Depending on its state, it introduces (unknown) latency in handling interrupts generated by the clock. Our prototype nodes ([39], Appendix A) are able to update their timing for low additional energy cost; it requires only very few processor instructions of which the timing is —due to timing support in hardware (timing capture registers)– non-crucial. Note that this particular hardware [39] is not able to change the frequency of the crystal oscillator; it can only adjust synchronization by adapting the length of intervals relative to the node's timing.

Two timing experiments were conducted to get a feeling for the clock accuracy of prototype wireless sensor node hardware [39]. Figure 7.4 shows the results of an experiment with nine prototype sensor nodes, all using crystal oscillators running at 32.768 kHz. One node acts as a timing reference and transmits a short message (comparable to the CM) every second, i.e. comparable to frame length of LMAC. The other nodes receive the message and compare the arrival time with the expected arrival time. The time difference is recorded to memory (sufficient space for 18 hours), together with the message number. This gives an indication of differences between clocks frequencies between transmitter and receivers. The measured difference in (relative) frequency is less than 7 ppm for the worst performing node.

In the above scenario, the nodes synchronize *only once* to the transmitting node during the 18 hours of the experiment. However, when receiver nodes are allowed to synchronize to every message, they can maintain relative synchronization with little error (caused by their own clock drift and clock drift in the transmitting node).

The results of this second experiment are plotted in Figure 7.5. In the graph, we plotted the results of the worst performing node in this and the previous experiment (node B). For this node, the timing difference falls within ±2 clock ticks for 99.93% of the cases. In 77% of the cases, the arrival time of the message matches the expected time and the node did not need to change its timing.

In our experiments, we paid no heed to time of flight of messages before they arrive at the receiver. The travel delay time of messages is small compared to the granularity of the clock tick of the 32.768 kHz oscillator (i.e. 31 μs). When we assume a (maximum) distance between transmitter and receiver of $d = 300$ m, it takes $d/c = 1$ μs, where c is the speed of light. The travel time of messages is thus smaller than the timing difference that we would expect from e.g. phase shift between clocks. It can, therefore, be ignored.

We conclude that nodes can maintain relative synchronization when timing is

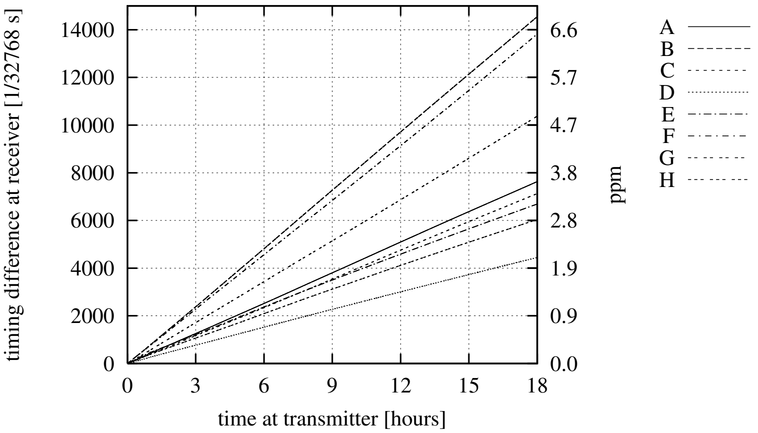


Figure 7.4 — One transmitter sends a message once per second. Eight other nodes record difference (plotted) in arrival and expected message arrival time

updated regularly. Consequently, guard intervals can be kept small. The updating of timing is in the discussed prototype wireless sensor demanding minimal energy consumption (i.e. energy costs of a few microcontroller cycles) and is not crucial in timing itself.

7.2.4.ii Hierarchical synchronization scheme

In our timing experiments, we saw that the prototype wireless sensor nodes can maintain relative synchronization with little error. To establish multi-hop synchronization in the network, we propose a hierarchical synchronization scheme (i.e. all timing is relative to the timing of a gateway node). We reuse the structure created for the routing tree (Figure 7.2).

In this synchronization scheme, every node synchronizes itself to its parent node and makes sure that it transmits its CM at a fixed time offset of its parent's transmission. A node qualifies to be a parent when it belongs to the set of nodes which is closer to a gateway. To create such a tree structure in the network, nodes indicate their hop distance to the closest gateway in a special field in the CM (Section 7.2.1.i). This has the advantage that for virtually no costs shortest path routing (ideas for a similar routing mechanism are presented in [67]) to the closest gateway is included in the LMAC protocol (Section 7.2.3).

The idea is to synchronize often, before large timing differences can occur between neighbouring nodes. In the LMAC protocol, nodes typically synchronize every frame. To allow for small timing differences, we let nodes observe a small guard and time out interval, as discussed before. The results of our experiments —in the previous section– suggest that these intervals can be equivalent to ±2 clock ticks i.e. $T_{Guard} = T_{Timeout} = \frac{2}{32768} \approx 0.06\,10^{-3}$ s. However in our experiments the nodes were all at the same temperature, thus ruling out effects of e.g. temperature drift. These factors

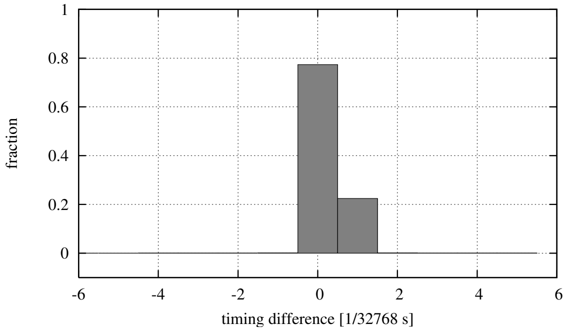


Figure 7.5 — Histogram of the clock difference if receiving node synchronizes every second (Node B)

need to be taken into consideration when carrying out the trade-off between length of guard intervals (i.e. energy consumption) and tolerance of timing (i.e. stability of the network).

The method of propagating (relative) synchronization comes in danger when there are unconnected paths in the network of equal hop length, which merge at a certain point. In the unconnected paths, the (local) timing requirements hold (i.e. each node is synchronized with its parent respecting a short guard interval), but when a node connects these branches (by definition of equal hop distance to a gateway), the timing difference between them (because of different accumulation of small timing errors in the paths) might have an offset, which is larger than the allowed guard interval.

For (uniformly) random deployed networks, Muthukrishnan et al. [75] show that the ratio between *the shortest route* (i.e. the distance a message needs to travel) and *the distance* between any two nodes in the network is bounded. In particular, the ratio is bounded by $1 + \alpha/2$ [75] where α is dependant on the density of node deployment ($0 < \alpha \leq 1$). The existence of this relation indicates separate branches —in which timing difference can occur– are not likely to exist.

7.3 Comparison with other WSN MAC protocols

In this section, we compare the performance of LMAC with other state of the art MAC protocols for wireless sensor networks. In general, the WSN MAC research community does not reach a consensus on how to compare performance of MAC protocols, stipulating that networking protocols are designed to optimize efficiency of specific WSN applications.

7.3.1 Comparison with TMAC, SMAC and LPL

In [59], a comparison has been made between MAC protocols designed for WSNs. In Figures 7.6 and 7.7, summarized results are shown (for details on simulation settings and complete results, the reader should refer to [59]).

From these results we conclude that —in the considered scenario– LMAC performs well on energy-efficiency and delivery ratio compared to TMAC [18], SMAC [111] and LPL [38]. The reason is that LMAC targets the two main causes of energy-waste in MAC protocols: (1) collisions and (2) idle listening. The LMAC protocol is able to schedule the access to the wireless medium and, therefore, it does not suffer from collisions when the payload increases. This results in good delivery ratios.

SMAC, TMAC and LPL have built in periods in which the transceiver is listening to the channel while there is no communication to receive. LMAC is able to reduce this idle listen time to a large extent; the protocol becomes even more energy-efficient for increasing payloads (Figure 7.7), due to an increased ratio of data bits and "overhead" (i.e. CM).

As customary, wireless links have been simulated to be perfect, i.e. no packet errors. Obviously, this is a departure from reality. Additionally, results are dependant on the (simulated) transceiver hardware, e.g. a different transceiver might result in different energy consumption results.

7.3.2 Comparison with EMACs

In this section, we compare LMAC with EMACs (Chapter 6). Both protocols have been implemented in the discrete event simulator OMNeT++ (`http://www.omnetpp.org`) using an identical transceiver abstraction layer that maintains transceiver usage statistics. The modelled transceiver is discussed in Appendix B. We compare transceiver usage statistics for the following: (1) EMACs with all nodes active, (2) EMACs with active/passive roles, and (3) LMAC. Both LMAC and EMACs frames consist of 32 time slots and have a duration of 1s. This length of frame has been verified to be sufficient to provide all nodes a non-conflicting time slot. For the protocols, identical timing parameters are used, e.g. during unused time slots.

Table 7.3 — Simulated scenarios

Scenario	Average connectivity	Nodes	Simulated MAC frames
1	2π	100	1000
2	3π	100	1000

The simulations consist of one gateway node and 99 other nodes. The nodes are (uniform) randomly deployed in a $l \times l$ square area. The circular transmission range of the nodes is tuned in such a way that the average connectivity is 2π or 3π (Table 7.3). In total, 25 (uniformly) random connected topologies have been considered to avoid the peculiarities of any specific non-uniform topology having an effect on transceiver usage statistics.

In our simulations, the protocols are first allowed to initiate, i.e. nodes are given 100 MAC frames (i.e. 100s) to reach the continuous operation state C (Section 4.4). This means that for EMACs the nodes have decided upon their active/passive role. After the initialization phase, the transceiver usage statistics are reset. By doing this, we exclude the setup phase of the network from having an effect on our results.

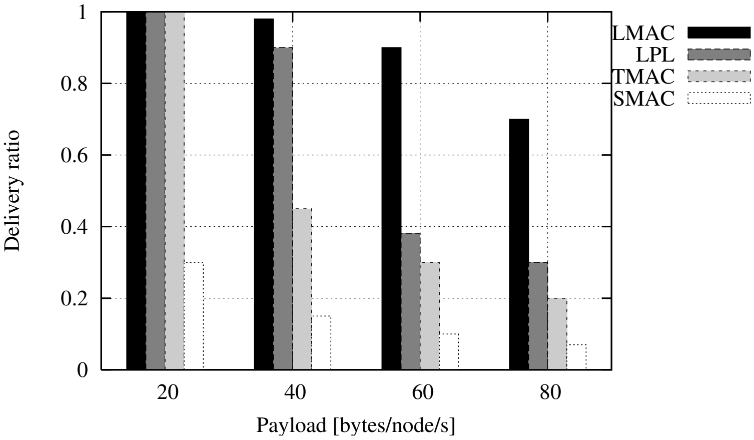


Figure 7.6 — Delivery ratio for different payloads [59]

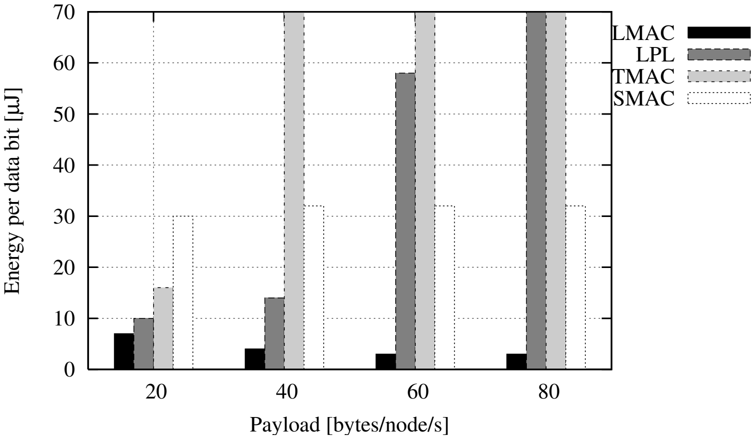


Figure 7.7 — Energy consumption per data bit for different payloads [59]

After the setup phase, transceiver usage statistics are collected during 900 frames (equivalent to 15 minutes). During this period, nodes generate messages (64 byte in length) at an average rate of one message per x MAC frames, resulting in average payload rate of $64/(x/2)$ bytes/s/node. Messages are addressed to random (active) neighbours.

Figures 7.8 and 7.9 show average transceiver statistics i.e. the percentage of time that the transceiver is *not* in its energy-efficient low-power mode.

Passive nodes in EMACs typically only have a few active neighbours. Therefore, certainly at high average message rates, collisions of CR messages occur. In practical implementations, nodes would try to resend messages. However, our simulations ignore CR collisions. A side effect is that the passive nodes do not always transmit their data messages, which has an obvious effect on transceiver usage and energy consumption. Both active EMACs and LMAC are collision-free.

First, we discuss the results of Figure 7.8. LMAC is, in the first simulation scenario, the most attractive choice regarding energy-consumption. The results show that active nodes in EMACs consume —due to listening to the CR section– more energy than nodes using the LMAC protocol. However, when the EMACs active/passive roles are taken by the nodes, the average energy-consumption of nodes drops, but remains above the level of LMAC.

In the case of EMACs with active set, messages were lost due to request collisions in the CR section (Figure 7.10). Consequently, LMAC and EMACs with active nodes only out perform EMACs with active set regarding delivery ratio.

Secondly, in Figure 7.9, we increase the average connectivity. Consequently, EMACs with active set assigns relatively more nodes to become passive compared to the scenario in Figure 7.8. Therefore, we expect that the efficiency of the EMACs protocol (with active set) —regarding energy-consumption– is increased. Note however, that receiving a TC (or CM) is more energy-consuming that idle listening during an empty time slot (we demonstrated this for LMAC in Section 7.2.4). Hence, when the connectivity in the network is increased, the overall energy-consumption of nodes increases. This is also reflected in Figure 7.9.

We conclude that there are two important parameters which determine whether EMACs or LMAC has to be used: (1) network connectivity, and (2) payload. EMACs with active/passive roles is more efficient concerning (average) power consumption of the transceiver than LMAC in highly connective networks. However, a prerequisite is that the payload of (passive) nodes should be very low. EMACs is not resilient against high (peak) loads generated by passive nodes.

7.4 Effects of different waiting times at network setup

We observe that, especially at network setup, many nodes receive an impulse to synchronize at the same time. This leads to collisions, as we discussed in Section 4.5. Therefore, we introduce randomness in reaction time W in the wait state *W* between synchronization with the network and the actual choice of a free time slot: $W = \{1, \ldots, W_{max}\}$, expressed in (integer number of) MAC frames [43].

To evaluate the effects of waiting time, we make use of the discrete event simulator OMNeT++ (`http://www.omnetpp.org`). The simulations consist of one gateway node

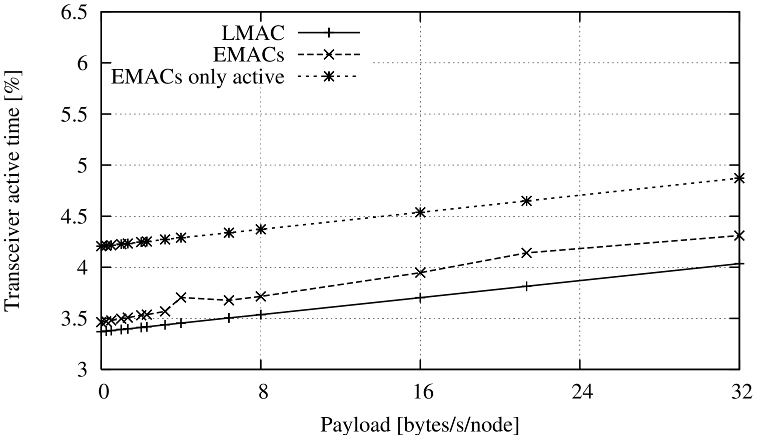


Figure 7.8 — Comparison between the LMAC and EMACs protocols. The figure shows averaged transceiver duty cycles for average connectivity 2π (simulation)

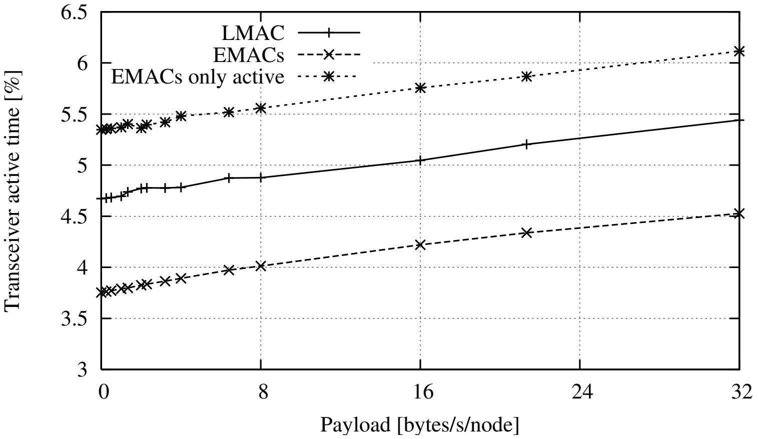


Figure 7.9 — Comparison between the LMAC and EMACs protocols. The figure shows averaged transceiver duty cycles for average connectivity 3π (simulation)

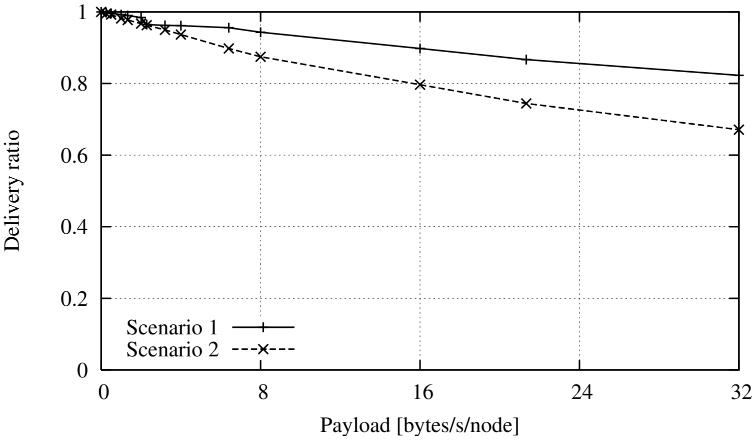


Figure 7.10 — Average delivery ratio of EMACs with active/passive nodes (simulation)

and 99 other nodes. The nodes are (uniform) randomly deployed in a $l \times l$ square area. The circular transmission range of the nodes is tuned in such a way that the average connectivity is 2π. In total, 500 (connected) topologies have been considered. The MAC frame consists of 32 time slots, which has been verified to be sufficient to provide all nodes a non-conflicting time slot. Nodes choose a time slot with equal probability from the set of free ones.

In our simulations, we count how many nodes released their time slot due to a collision report of other nodes. Consequently, only *notified* collisions are counted. Additionally, we keep track of how many nodes are in the continuous operation state C and are thus ready to participate in multi-hop networking.

In Figures 7.11 and 7.12 we show the effects of different maximum waiting times on the number of collisions and the network setup time. To improve readability, the standard deviations have not been drawn in the figures.

When the waking-up of nodes is more evenly distributed in time by incrementing W_{max}, the number of collisions in the network drops as expected. A side effect is —obviously– that the network takes longer to be fully synchronized and ready for communication.

The S-shape of the start-up curves in Figure 7.12 can be explained as follows. In Figure 7.13, the (integrated) distribution of nodes is shown against their hop distance. This distribution clearly has a similar shape. In the LMAC protocol, the hop distance determines the start-up time of the WSN.

When we look —for example– at the curve of $W_{max} = 1$ (i.e. the nodes take *two* frames to reach state $\boldsymbol{C}$), all nodes in the network are in the continuous state after 20 MAC frames. This matches our findings in Figure 7.13, where most nodes are within a 10 hop distance of the gateway.

We conclude that the W_{max} parameter influences the number of collisions and the start-up time of the LMAC network. In the settings we presented in this section, it is

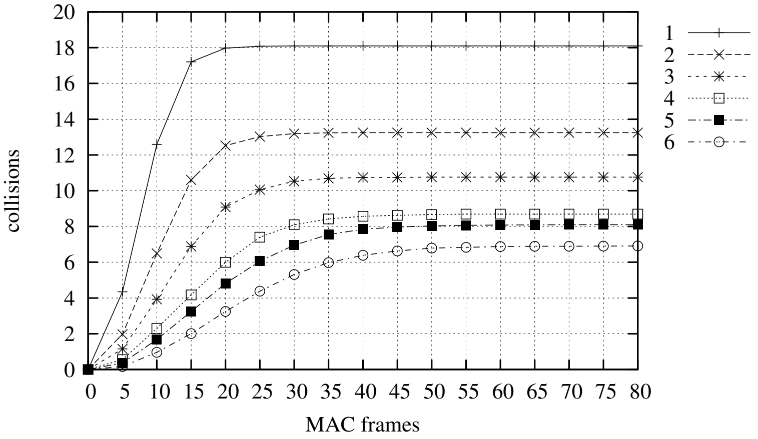


Figure 7.11 — Average number of notified collisions for different maximum start-up times W_{max} (uniform time slot choice)

questionable why we would choose $W_{max} > 1$. Namely, although more collisions occur for $W_{max} = 1$, the network is still faster active than in other settings. However, in the next section we present methods to reduce (best case) message transfer times. In these methods, we adapt the strategy by which nodes choose their time slots. Inherently, more collisions occur and the spreading of nodes entering the discover state D in time is relevant.

7.5 Reducing (best case) message delay

Message delay —or latency– is one of the main concerns in schedule-based MAC protocols. Namely, nodes potentially have to postpone transmissions according to their schedule. In e.g. CSMA, nodes are not tied to a schedule, but can immediately transmit when the channel is idle. The latter channel sharing approach intuitively has better latency performance.

In this section, we will look at three latency-reducing strategies for LMAC, which focus on picking time slots during network setup. It is important to note that we discuss here the *best case* latency imposed by the scheduling nature of communication in LMAC (i.e. we assume that nodes have empty message buffers and immediately try to forward or broadcast any recently received message).

7.5.1 Classification of latency

The scheduling principle in LMAC requires a node to wait until its time slot comes up before it can transmit any data message. This delay time is the source of latency (Figure 7.14). We distinguish two types of latency sources: (1) the time between generation of the message and the first opportunity to transmit it ($t_1 - t_0$), and (2)

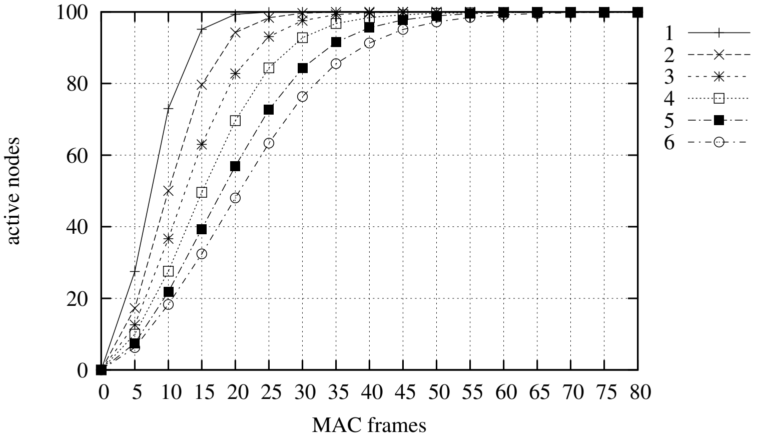


Figure 7.12 — Average network start-up time for different maximum start-up times W_{max} (uniform time slot choice)

latency in the multi-hop forwarding process $(t_2 - t_1)$. The first type introduces an average latency of

$$E[t_1 - t_0] = \frac{1}{T_{frame}} \int_{t=0}^{T_{frame}} t \; dt = \frac{1}{2} T_{frame} \tag{7.2}$$

under the conditions that the message generation is uniformly distributed in time and the node's message buffer is empty i.e. the message is sent within one MAC frame $(0 \leq t_1 - t_0 \leq T_{frame})$.

Latency caused by the first source cannot be reduced without tuning the MAC scheduling to the generated data of the node. Therefore, we leave aside the first type of latency and focus on the second type: the latency of the multi-hop forwarding of the message.

In Section 2.3.7.i, we concluded that messages containing sensor readings are normally forwarded to gateway nodes. These nodes, in turn, configure the WSN by injecting a description of their interest. Therefore, we classify multi-hop latency into two classes:

- **Uplink latency** — The latency from *sensor nodes* to the point where the interest for the sensor readings lies, i.e. *gateways* (communication type (2) in Figure 2.1). Note that the mechanism for routing messages to the gateway(s) is already included in the LMAC protocol by reusing information necessary for synchronization. Hence, we assume that routing overhead does not contribute to uplink latency.

- **Downlink latency** — The latency from the *gateway* to the *sensor nodes* (communication type (1) in Figure 2.1). This is the time it takes before a node is informed of a change of interest of the gateway. We assume here that the gate-

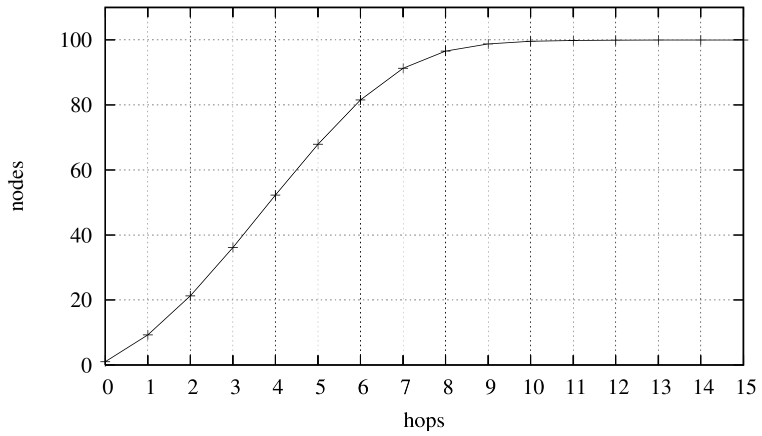


Figure 7.13 — Average (integrated) hop distribution of nodes in random topologies

way floods the network with a message, which is repeated by all nodes (although nodes remove duplicate messages in order to terminate the flooding).

We expect that most of the generated messages in the WSN contain sensorial information and thus travel towards gateways. Therefore, our goal is to reduce uplink latency.

7.5.2 Approach

We observe that *uplink latency* can be reduced when sibling nodes transmit just before their parent is going to transmit. The parent node receives the message and immediately is able to forward it to the next node in the routing tree. This reduces the time that the message resides in the parent, and results in better uplink latency performance. A similar principle is used in the DMAC [65] protocol to reduce latency.

To enable the reduction of latency, we adapt the time slot choosing mechanism (Section 4.3.2). In this approach, we let sibling nodes prefer time slots, which are shortly positioned before the time slot of a (potential) parent node in the MAC frame, yet collision of schedules are solved in the regular way (Section 4.5). Note that we maintain our assumption that nodes are not able to transmit information before they have acquired a time slot. Therefore, our strategy has a random nature (i.e. no collaboration between the nodes). Due to the fact that some time slots are more attractive than others, we expect an increase of collisions during network setup.

Actually, in the LMAC protocol, uplink and downlink latency are related to each other. When we try to reduce uplink latency —by choosing the siblings time slot just before the parents time slot; say time slot 8 for the child and time slot 9 for the parent–, the downlink latency will increase, because the message has to wait in the child node from time slot 9 until the current frame has ended and time slot 8 has been reached.

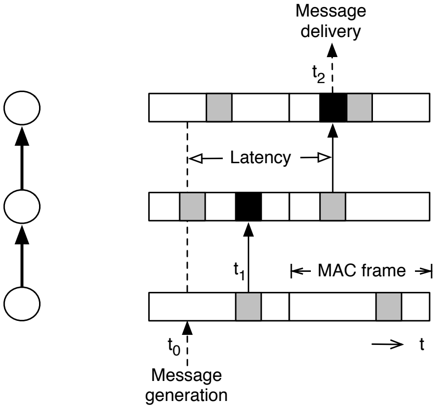


Figure 7.14 — Message latency in a multi-hop network

Thus, reducing the uplink latency will have as a side effect an increase of the downlink latency. However, when flooding the network with interest messages, messages travel via the quickest routes, as shown in Figure 7.15. We show in the next sections that this effect causes the downlink latency to be less affected by the different time slot choosing adaptations than the uplink latency.

7.5.3 Experimental analysis

To evaluate the latency performance of the LMAC protocol, we make use of the discrete event simulator OMNeT++ (`http://www.omnetpp.org`). The simulations consist of one gateway node and 99 other nodes. The nodes are (uniform) randomly deployed in a $l \times l$ square area. The circular transmission range of the nodes is tuned in such a way that the average connectivity is 2π. In total, 500 (connected) topologies have been considered, similarly as in Section 7.4, to be able to compare the number of reported collisions and network setup progress. Due to the fact that more collisions are expected, we put $W_{max} = 6$ in the experiments to spread the activation of nodes over time.

The best case latency statistics are gathered offline after the network has been completely set up (i.e. 100% of the nodes are in the continuous operation state). To make the results independent of implementation choices (e.g. the number of time slots in a frame or the MAC frame length T_{frame}) latency is expressed in MAC frame units. Note, however, that we use 32 time slots per frame, which is sufficient to give each node a non-interfering time slot in the considered topologies.

We consider four methods in our latency experiments: (1) Uniform slot choice from the set of free time slots (i.e. the time slot choosing method of LMAC discussed so far), (2) choose only the best (latency-wise) time slot, (3) choose a latency-wise better time slot with higher probability than a less desirable time slot, and (4) divide the list of free time slots into a better half and worse half, and choose uniformly a

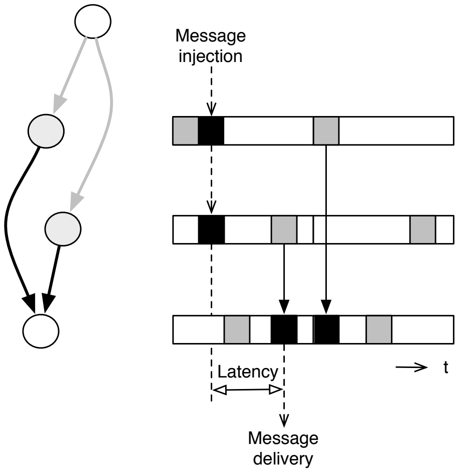


Figure 7.15 — Broadcasted messages (i.e. flooding) travel via quickest routes. Both grey nodes forward the query/business rule to the node at the bottom. However, the arrival of the first message determines the downlink latency

time slot from the better half. These four methods have been implemented in our simulation environment.

7.5.4 Results and discussion

Results of the latency experiments are shown in Figures 7.16 and 7.17. For readability, the standard deviations have been omitted in the figures.

The methods and results are discussed in more detail:

1. **Uniform time slot choice** — Nodes choose a time slot with uniform probability from the set of free ones. On average, there is half a frame length difference between the time slot of the parent and the child node (similarly as Equation 7.2) and thus we expect $\frac{1}{2}T_{frame}$ uplink latency per hop. This is also reflected in our simulations (Figure 7.16).

   Figure 7.17(1) shows downlink latency results. Obviously, the downlink latency will be a little shorter than the uplink latency due to the fact that broadcast messages are not sent to a specific node, but are registered with all receiving nodes. Hence, such messages are able to travel a little faster through the network than in the uplink case i.e. 0.4 MAC frames per hop.

The above method of choosing time slot is also proposed in Chapter 4. For the other three strategies, we let nodes rank the free time slots, so that a time slot which would give a lower local latency gets a higher weight. In addition, we let sibling nodes choose a parent in the routing process, which has a time slot closest (after) the sibling's time slot.

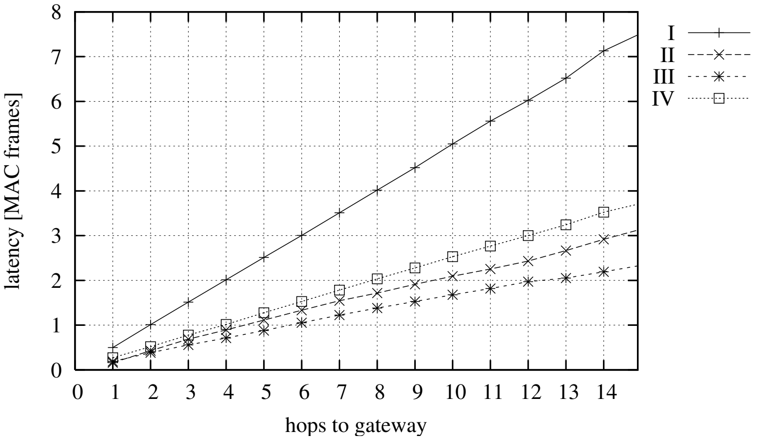


Figure 7.16 — Uplink latency (expressed in MAC frames); (1) Uniform, (2) Best slot, (3) Binomial, and (4) Uniform from the best half of the free time slots

2. **Best (uplink latency-wise) time slot choice** — In this scenario, we let nodes only choose the *best* possible time slot after the random wait interval. This aggressive method results in better (local) uplink latency, but since many nodes compete for the same best time slot (and thus the same parent in the uplink routing process), many collisions will occur and network setup will be delayed (hence our choice of W_{max}). This is shown by our network setup results in Figures 7.19 and 7.18. The number of notified collisions has increased from 6.9 (standard deviation 3.89) in the uniform method (1) to 104.9 (standard deviation 19.1), resulting in a delayed network setup. Yet, the uplink latency is reduced. In the next uplink latency reducing strategy, we propose a different method that reduces the number of notified collisions.

   We conclude that this method has better uplink latency performance than the uniform method (1), however in comparison, Figure 7.17 shows an increased downlink latency for this method.

3. **Better time slots get higher probability** — In this latency-reducing strategy, we let nodes consider *every* free time slot, not only the best. The difference with method (1) is that we give time slots with higher weight (and thus shorter uplink latency) a higher probability of being chosen. In our simulations, we have implemented this as follows. Nodes sort the list $\mathcal{T}$ of free time slot from high to low weight and choose a time slot using a Bernoulli-process. The probability that the i^{th} time slot is chosen is

$$P\{\text{time slot i is chosen}\} = \begin{cases} p\,(1-p)^{i-1} & 1 \leq i < |\mathcal{T}| \\ 1-(1-p)^{i-1} & i = |\mathcal{T}| \end{cases} \tag{7.3}$$

In the simulation, we used a probability of $p = 0.3$. In our simulations, this

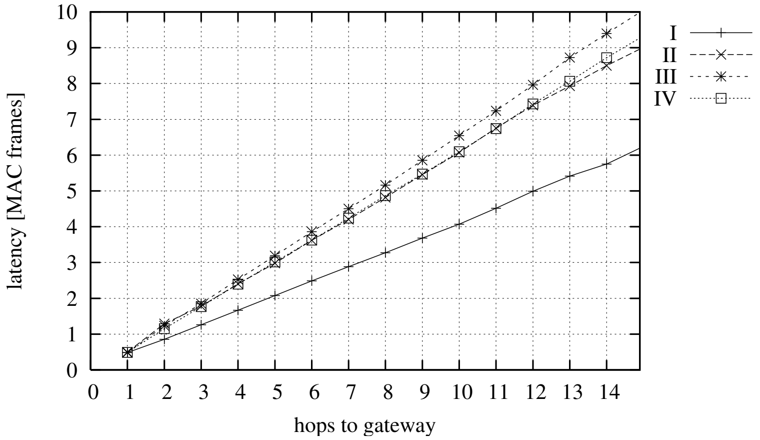


Figure 7.17 — Downlink latency (expressed in MAC frames); (1) Uniform, (2) Best slot, (3) Binomial, and (4) Uniform from the best half of the free time slots

value gave good results. However, we leave it to our future work to obtain a solid argument for this parameter.

In strategy (3), nodes will give a *high* probability of choosing a time slot with low (local) latency, but not all nodes will pick the same, best time slot and, therefore, it does not come as surprise that this method gives the best uplink latency results of the methods we compared. The latency is reduced with a factor 3 compared to the uniform method (Figure 7.16). The increased probability for the second best, third best and so further time slot also makes that the downlink latency is better than for strategy (2). Also the number of collisions is lower than in method (2) (Figure 7.19).

The good performance of this method is mainly due to the fact that this method spreads —with the spreading of time slot choice– the choice of the parent node for the uplink message transfer. Hence, multiple time slots qualify for good latency performance.

4. **Uniform slot choice from the best half of the free time slots** — With this simple strategy we are able to improve uplink latency by a factor 2 compared to the first method (uniform time slot choice).

   In this method, nodes divide the list of free time slots into a better half and worse half, delete the worse half and choose uniformly a time slot from the remaining list. This method results in a comparable number of collisions as in the uniform case method (1) (Figure 7.19).

We showed that latency (for messages containing sensor readings travelling over multiple hops) can easily be reduced in the LMAC protocol, using different strategies for choosing a time slot to control from the set of free ones. Our general approach was to let nodes prefer time slots that are located just before the time slot of a

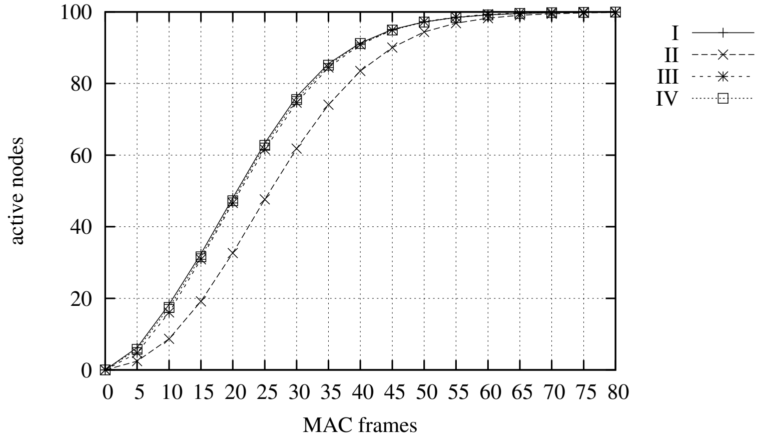


Figure 7.18 — Network startup times for different strategies to choose a free time slot ($W_{max} = 6$); (1) Uniform, (2) Best slot, (3) Binomial, and (4) Uniform from the best half of the free time slots

routing parent. The other functionality of the MAC protocol remained unchanged. The resulting (average) uplink latency is $\frac{1}{6}$ MAC frame per hop (thus messages can travel 6 hops per frame), assuming that nodes have no backlogged messages. The latency for disseminating *queries* is in that case $\frac{2}{3}$ MAC frame per hop.

7.6 Adapting LMAC to expected data volume

Existing MAC protocols for WSNs often operate independently of the application e.g. the injected queries or the kind of data that flows through the network [9, 18, 40, 65, 85, 111]. Additionally, the closer nodes are to a gateway, the more data they have to forward (Figure 7.20), assuming that the data cannot be efficiently aggregated. In this section, we present mechanisms to make LMAC adaptive to the volume requirements of the actual sensor application.

Woo et al. [105] present a rate control mechanism that takes fairness into consideration. It defines fairness as giving every node in the network an equal opportunity to transmit its data. We, however, are not interested in having all nodes transmit data simultaneously.

We consider a heterogeneous network where injected queries will be distributed both in the spatial and temporal sense. We define fairness as follows: only nodes that are able to service an incoming query or nodes that have children, which can service a particular query, should be given the chance to transmit their data. Nodes that are not involved should be given lower priority. Thus, the priority given to a node is directly proportional to the amount of data a node is expected to transmit. This is only a general definition of how we interpret fairness.

Our mechanism depends on two aspects:

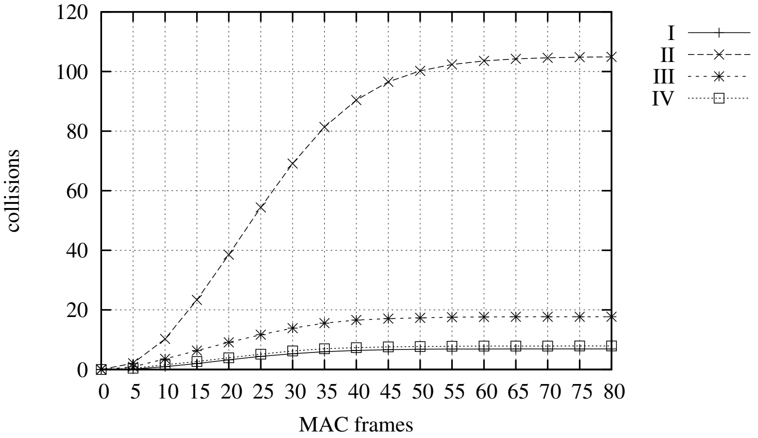


Figure 7.19 — Number of notified collisions for different strategies to choose a free time slot ($W_{max} = 6$); (1) Uniform, (2) Best slot, (3) Binomial, and (4) Uniform from the best half of the free time slots

- **Estimation of data volume** — In our previous work [15, 41], we have presented the *data distribution table* (DDT), which resides in every node in the network and is used to predict the amount of data that will flow through the node for a newly injected query.

- **Adaptation of LMAC** — We adapt the LMAC protocol by tuning the number of time slots a node controls, i.e. the more time slots a node controls, the greater its share in the channel capacity and the more data it can transport. The prediction of the DDT is translated into a (discrete) number of time slots that would be required to transport the data and together with fairness considerations, this is used to obtain more time slots or to release them. The rules for obtaining time slot(s) are not altered.

The latter aspect has our focus. In the next section, we present *adaptive, information-centric* (AI-)LMAC. In Section 7.6.2, the simulation results of the adaptive version of LMAC are discussed.

7.6.1 Adaptive and information-centric (AI-)LMAC

We now describe how AI-LMAC can adapt its operation using the information provided in the DDT [15, 16, 41]. Unlike LMAC, which allows every node within the network to own only one slot [40], AI-LMAC allows a node to control multiple slots. Also, AI-LMAC is able to vary the number of slots a particular node owns depending on the amount of data that is expected to flow through it. This ensures fairness in the sense that the bandwidth allocated to a node corresponds to the traffic it is expected to encounter. For example, it would be pointless and energy-wasting to allocate a large number of slots to a particular node that is not generating or relaying significant amounts of data.

7.6.1.i Hierarchical advice

In AI-LMAC, we assume that a parent-child relationship exists between all the nodes in the network, such that the gateway of the network can be considered to be the highest parent in the hierarchy. This relationship is automatically created by the routing included in LMAC (Section 7.2.3).

The DDT allows nodes to predict the data volume their children will generate based upon a *current* query set in the network and *history* [15, 41]. A node is not aware of the data generated by the other children of its own parent node as they may not be in direct range. Thus, the parent is the only node that has knowledge of the proportion of data that will be contributed by each of its immediate children.

The idea here is that if a node realises that a subset of its immediate children is going to transmit large quantities of data, then more attention needs to be paid to this particular subset of child nodes. In this case, when we say more attention, we are actually referring to assigning multiple slots to a particular child.

However, even though a parent node knows which child node deserves more slots to be assigned to it, it cannot send such a rigid instruction (i.e. what time slots to claim) to its children. This is because in LMAC, when a node performs slot assignment, it has knowledge of the time slot ownership of its first and second order nodes. In this case, the parent node does not know time slot ownership information about the slot assignments of its child node's second order nodes since they are three hops away. Thus, the responsibility of the parent node is simply to *advise* the child, i.e. the parent node sends a message to every one of its children indicating the ideal number of slots that a particular node should take up under the current conditions. It is then up to the child node to follow the advice as closely as possible. This naturally depends on the number of empty slots available.

7.6.1.ii Horizontal and vertical fairness

The process of giving advice starts at the gateway node of the tree when a query is first injected into the network. This process then percolates down the branches of the tree towards the leaf nodes. If, however, the process of giving advice started at an intermediate node, this increases the chance of performing unfair slot allocations. This is because a node assigning time slots would not be aware of the bandwidth requirements of all its sibling nodes which are not within its direct range. From this argument, it is obvious that if we apply this rule repeatedly, the gateway node is the only node which can assign slots fairly at the beginning. We term this as *horizontal fairness* as the mechanism ensures that all sibling nodes (i.e. at the same level) under a certain parent are allocated slots fairly.

Apart from establishing a horizontal relationship between nodes, we also introduce a mechanism to include *vertical fairness*. In order to prevent congestion problems, our mechanism ensures that the total number of slots assigned to the immediate children of a certain parent node does not exceed the number of slots owned by the parent. This reduces the likelihood of data packets being dropped due to lack of bandwidth.

Furthermore, leaf nodes are prevented from being allocated excessive bandwidth using this mechanism.

Thus, introducing *two dimensional fairness* ensures that the number of slots taken up by a node does not only depend on its siblings but on its parent as well.

7.6.1.iii Executing the given advice

Once a node has received the ideal number of slots it should take up, it checks to see which slots are free following normal protocol rules of LMAC and it will try to take up as many slots as advised. To ensure a balanced slot allocation between children nodes, nodes will increment the number of controlled slots in turns at a rate of one time slot per frame.

Our framework provides a mechanism to assign more bandwidth to those parts in the network that encounter more data traffic than others. In fact, the assigned bandwidth is proportional to the expected traffic. Hence, our framework is able to minimise (1) the overall *latency* in the network and (2) the number of messages which need to be buffered in the nodes can be substantially reduced.

7.6.2 Experimental setup

Simulation results are obtained by using the discrete event simulator OMNeT++ (`http://www.omnetpp.org`). Results are averaged over five different network topologies consisting of 49 nodes and one gateway.

In our simulated scenarios, the routing capabilities of LMAC are used to create the necessary parent-child relationships. Additionally, the nodes keep track of their number of siblings and propagate these results to the gateway node. When all nodes are active (i.e. have at least one time slot), the gateway node starts providing advice to its direct siblings. The siblings follow the advice as closely as possible and create advice for their children according to the above description. We allow the networks 200 MAC frames ($N_{timeslots} = 32$) to initiate and execute the advices, which is plenty of time as seen in the results of Section 7.4.

The DDT is not modelled, however, to simulate a query, all nodes in the network generate after the initialization phase a message burst consisting of five messages — filling the entire DM and aggregation is not applied– at a rate of one message per four MAC frames. Thus the expected data volume is in this case proportionally the number of siblings a node has. Hence, this parameter serves as input for the given advices.

In our simulations, we vary the maximum number of time slots a node is allowed to control $N_{max\ slots}$. Obviously, the given advice does not exceed in any case this maximum. Ten different runs were carried out per topology and per setting. We collect average message delay and message buffer sizes i.e. nodes are modelled with infinite memory space to backlog messages.

7.6.2.i AI-LMAC results

The results clearly indicate that latency is proportionally reduced with the maximum number of controlled slots. However, this holds true only until $N_{max\ slots} = 8$

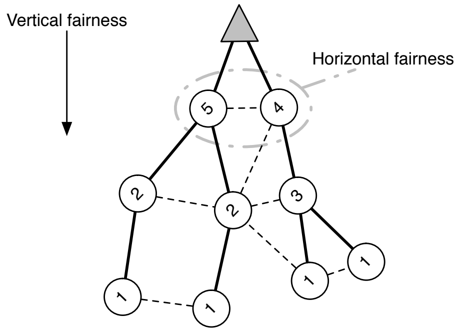


Figure 7.20 — Two-dimensional fairness in AILMAC: (1) Horizontal fairness divides bandwidth between siblings to the ratio of the required bandwidth, and (2) vertical fairness ensures that siblings do not get assigned more bandwidth than their parent

slots. For the $N_{max\ slots} = 12$ and $N_{max\ slots} = 16$ slot scenarios, the number of free slots in the network rapidly decreases with every hop from the gateway and, thus, the nodes are not able to comply with the advice. Consequently, a bottleneck is created at a few hops (6 to 8) from the root, resulting in higher latency for messages created in those areas.

In the LMAC protocol, nodes are able to *receive* up to $N_{timeslots} - 1$ messages per frame, however nodes can transmit only one message per frame. When the number of incoming messages exceeds the number of messages that can be transmitted during a frame, the additional incoming messages have to be buffered, imposing memory requirements. In AI-LMAC, the number of messages that can be transmitted per frame is dependent on the number of time slots the node controls. Therefore, we expect that AI-LMAC requires less memory to backlog messages.

For each of the scenarios, we have collected data about the maximum number of messages backlogged per node (Table 7.4). These results reflect the same trend as in Figure 7.21: assigning more time slots to nodes reduces memory requirements. However, when nodes are too greedy in obtaining time slots (i.e. high $N_{max\ slots}$), siblings cannot comply with advice because all time slots are occupied. Consequently, these siblings lack bandwidth and, therefore, require more memory to backlog messages.

Note that in real-life implementations, the capacity to hold back logged messages will be limited due to scarce memory resources in the sensor nodes. For example, the wireless sensor platforms discussed in Table A.2 have 4 kB of RAM memory. Assuming that a quarter of the memory is used for application message queues and that messages are 64 bytes each, a wireless sensor is able to backlog 16 messages. Consequently, many messages get lost in these scenarios due to congestion. This stipulates the importance of congestion control in WSNs.

Note that the message delay results (Figure 7.21) are —in every case– considerably

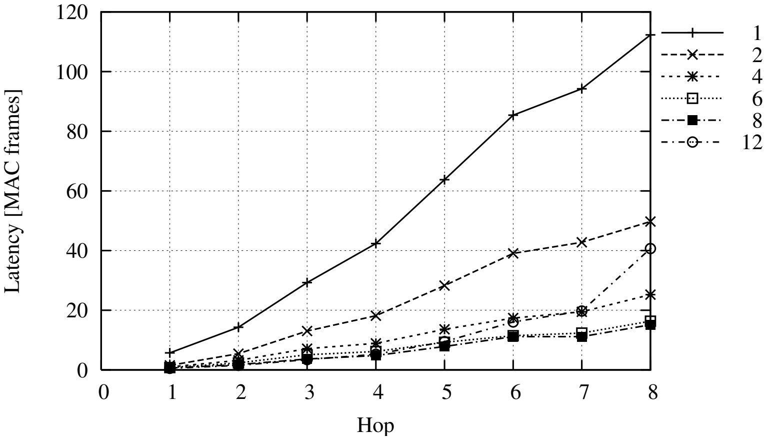


Figure 7.21 — Latency (expressed in MAC frames) of AI-LMAC for different maximum controlled time slots (Simulation). The maximum advice scenario of $N_{max\ slots} = 16$ time slots is not plotted since it matches the 12 slot scenario

Table 7.4 — Maximum number of backlogged messages per node (worst case)

$N_{max\ slots}$ [time slots]	Maximum messages
1 (equivalent to LMAC)	105
2	63
4	34
8	32
12	54
16	56

worse than the results we presented in Section 7.5. Hence, we conclude that latency is reliant on the actual volume of data that is transported by the network.

Future research will look into methods that can be employed to improve the energy efficiency of the protocol, i.e. reducing the overhead of the protocol by assigning time slots in a contiguous manner. Currently, nodes are required to transmit CMs in every time slot they control, however when these time slots are contiguous, this overhead might be reduced.

7.7 Conclusion

Schedule-based MAC protocols have the advantage that each node gets its own turn in using the medium and, thus, that high peak loads can be handled without energy-wasting collisions. In this chapter, we presented a schedule-based MAC protocol for multi-hop wireless sensor networks, which targets two major causes of energy-wastage in MAC protocols: collision of messages and idle listening. It has been shown that the LMAC protocol is energy-efficient and has a good delivery ratio in [59]. It performs considerably better than when SMAC, TMAC or LPL is used to organize the access of the wireless medium.

A key issue in scheduled medium access is that neighbouring nodes in the network have to be synchronized with each other. Without precise synchronization, nodes have to use long guard intervals to ensure that receivers are ready when the transmitter starts transmitting. Valuable energy is wasted in this case. Synchronization experiments with prototype sensor nodes showed a (worst case) clock drift of 7 ppm. We presented a synchronization scheme in which nodes synchronize at least once every frame to parent nodes (i.e. nodes that are closer to the gateway; the source of timing) using the carefully timed transmission of their CM. By experimentation we showed that the timing difference falls within ±2 ($32.768kHz$) clock ticks for 99.93% of the cases. Thus, when applying a small guard time, synchronization can be ensured.

We studied the effects of different maximum waiting times (the time between the synchronization impulse and the moment the node starts choosing a time slot). We found that there is a trade-off between the number of collisions (i.e. nodes that choose the same time slot to control) at network setup and the speed at which the network is setup. A longer maximum wait interval results in lesser collisions, but also slows down the network setup.

An often referred weak point of schedule-based MAC protocols is the latency they potentially induce. This is not the case for the LMAC protocol. We showed that latency (for messages containing sensor readings travelling over multiple hops) can be easily reduced in the LMAC protocol, using different strategies for choosing a time slot to control from the set of free ones. The resulting (average) latency is $\frac{1}{6}$ MAC frame per hop (thus messages can travel 6 hops per frame). The latency for disseminating *queries* is in that case $\frac{2}{3}$ MAC frame per hop.

A fundamental problem in WSNs is that nodes closer to the gateways assist more nodes in forwarding data than nodes further away from the gateways. This has not only an impact on the battery life-time of these nodes, but also on the memory that must be available to backlog all incoming messages. We argued that congestion control is of vital importance for WSNs. Based upon the current query set and history, an estimation can be made of the volume of data that has to be transported by nodes [15, 41]. We present a framework in which LMAC is made adaptive to the expected volume of data.

Chapter 8

Attack and defence of LMAC

The LMAC protocol (Chapter 7) can be easily compromised when attackers have full knowledge of the protocol. Encrypting messages, applying source authentication and authenticating message content complicates the task of undermining a wireless sensor network, yet security keys are —for various reasons– easily compromised. Data link layers are even more easily attacked with jamming. *In this chapter, we present such (energy-efficient) jamming attacks. For this class of attacks little knowledge is required about the MAC protocol, which makes it a realistic threat. We show its effectiveness on LMAC and present countermeasures.*

8.1 Introduction

In a network where sensors communicate wirelessly, it is particularly easy to eavesdrop or inject false messages. When attackers have full knowledge of the MAC protocol in use, the WSN can easily be attacked by misusing link layer functions of well-behaved nodes. An attacker can for example deploy (colluding) malicious nodes in the network or compromise existing nodes. These nodes use an altered link layer to undermine the function of the WSN.

In the case of LMAC, the WSN can be undermined for example as follows (at link layer level):

- **Distance to gateway attack** — A malicious node can intercept messages intended

This chapter is a revision of the technical report ”Energy- Efficient Link-Layer Jamming Attacks against Wireless Sensor Network MAC Protocols” in CTIT Technical report series, TR-CTIT-06-18 [44] for genuine gateways by altering (i.e. lower) the distance to gateway field in the CM of LMAC. The messages can be discarded or misused. In addition, the malicious node can control the timing of the network and can, hence, create a fragmented network, disrupting the communication structure.

- **Bit vector of occupied slots attack** — A malicious node can alter its bit vector of occupied time slots in such a way that nodes choosing time slots observe all time slots as occupied. Consequently, those nodes are not able to use the medium. Nodes already in the continuous operation state are not affected by this attack.

- **Collision reporting attack** — A malicious node can attack its neighbours by reporting a collision in their time slots. The nodes reinitiate the time slot choosing process and are (temporary) unable to use the medium. Combined with the occupied slots vector attack, this is a strong attack to prevent well-behaved nodes from using the medium.

- **Destination address attack** — Malicious nodes can misuse the addressing mechanism in LMAC to address neighbouring nodes, but fail to send actual data packets. This attack increases the power consumption of the well-behaved nodes, since they would keep their transceiver in receive state after the received CM. The attack is especially efficient when the broadcast address is used by the attacker; it would mislead all its neighbours.

Solutions to full knowledge attacks are (1) authentication of the transmitting node, (2) authentication of the packet contents, and (3) message secrecy [54, 97]. The first two methods allow a well-behaved node to distinguish between trustworthy packets of unimpeachable nodes and packets of attackers. The secrecy of the messages —generally attained with message encryption– prevents malicious nodes to obtain important link layer information, e.g. the current time slot number.

Sensor nodes are susceptible to physical capture and tamper-resistance of the node will not win out [97]; security keys are therefore easily compromised. Yet, another threat to WSN link layers are minimal knowledge attacks, which are the topic of this chapter.

Jamming attacks on the physical and data link layer of WSNs have recently attracted attention [61, 108, 109]. In particular, Xu et al. [108] propose four generic jammer models, namely (1) the constant jammer (emits a constant noise), (2) the deceptive jammer (either fabricates or replays valid signals on the channel incessantly), (3) the random jammer (sleeps for a random time and jams for a random time) and (4) the reactive jammer (listens for activity on the channel, and in case of activity, immediately sends out a random signal to collide with the existing signal on the channel).

According to Xu et al. [108], the constant, deceptive and reactive jammers are effective jammers in that they can cause the packet delivery ratio to fall to zero or almost zero, if they are placed within a suitable distance from the victims. However, these jammers are also *energy-inefficient*, meaning they exhaust their energy earlier than their victims when given similar energy budgets. Although random jammers save energy by sleeping, they are less effective.

Our contribution is to develop jamming attacks that (1) work on encrypted packets, (2) are as effective as constant/deceptive/reactive jamming, and (3) at the same time are more energy-efficient than random jamming or reactive jamming. We implement such jamming attacks by exploiting the semantics of the data link layer and show the results quantitatively. The fact that our attacks are applicable to three representative MAC protocols suggests the same attacks are applicable to a wide range of other protocols belonging to the same categories as these protocols. Our analysis of the attacks provides new insights into the timing considerations of MAC protocols with regards to security, and provides hints on which category of protocols provides the best protection against our attacks so far, in the absence of effective countermeasures.

The motivation for this work stems from the concern that if an attacker can program and deploy a general-purpose link-layer jamming network that is able to jam any WSN effectively and energy-efficiently, and if a high entry barrier is not maintained for such a low cost attack, a WSN can never in any practical sense be secure.

A counter-argument might be that energy efficiency is no concern to powerful attackers, but even powerful jammers come with a finite energy supply and they would advertise their presence and location if they simply blast away with high power levels. This is something a sensible attacker would avoid.

This chapter is organized as follows. We start by stating the assumptions on which our attacks are based in Section 8.2. We then describe the attack algorithms in Section 8.3 with a focus on LMAC. Section 8.4 describes how the protocols and the corresponding attacks are simulated, and how the results are evaluated. The results are given in Section 8.5. The implications of our work for other protocols are discussed in Section 8.6. Section 8.7 explores some potential countermeasures.

In Section 8.8, we focus on one MAC protocol in particular; the LMAC protocol. We discuss the implementation of an efficient jammer for this protocol on prototype node hardware and we explore the effectiveness of LMAC countermeasures. Experiments are used to validate our simulation model for a small topology. Related work is discussed in Section 8.9. Finally, Section 8.10 provides conclusions.

8.2 Assumptions

We assume an attacker has two goals: the primary goal is to disrupt the network by preventing messages from arriving at the gateway node, and the secondary goal is to increase the energy wastage of the wireless sensors.

Our attacks depend on three assumptions: (1) the jammer nodes know physical layer properties used by the victim nodes, (2) the jammer nodes can measure the length of a packet, and (3) the jammer nodes know what MAC protocol the victim nodes are running. Requirements (1) and (2) should be easy to satisfy in practice. Requirement (3) is more demanding, but not impractical to satisfy. Note that the jammer nodes do not need to know the *content* of the packets, so our attacks work even if the packets are encrypted. Adding to the significance of our attacks is that the attacker does not need to capture and compromise any existing sensor nodes.

Our (general) attack scenario is applicable under the following concrete set of circumstances:

- **Unknown or no fixed sink nodes** — There are no fixed sink nodes to attack, e.g. in directed diffusion [50] (Section 2.4.2) where ID-less interests propagate through the network, there is no way for a node to tell where the real sinks are.
- **No strategic deployment of malicious nodes** — It is infeasible for the attacker to deploy its nodes strategically.
- **Link layer protection** — The target WSN is protected with a link layer authentication (and optionally encryption) scheme like TinySec [54].

In all these cases, the attacker finds it likely appealing to distribute its jammer nodes among the target WSN and to apply our jamming attacks.

Naturally, our attacks are affected by the choice of values assigned to the protocol parameters. Throughout the chapter, we choose values for the protocol parameters that are either realistic or to what the creators of the protocols recommend as possible, in the absence of a universal consensus and a scientifically rigorous way of deriving these values.

8.3 Description of attacks

How do we attack a protocol without knowing the content of exchanged messages? Since the attacker is solely interested in jamming *application packets*, our first observation is that we can focus on jamming long packets, because often application packets are longer than the control packets of the medium access control protocol (e.g. the case in LMAC). We can do this by sorting packets according to their length and predict when long packets arrive.

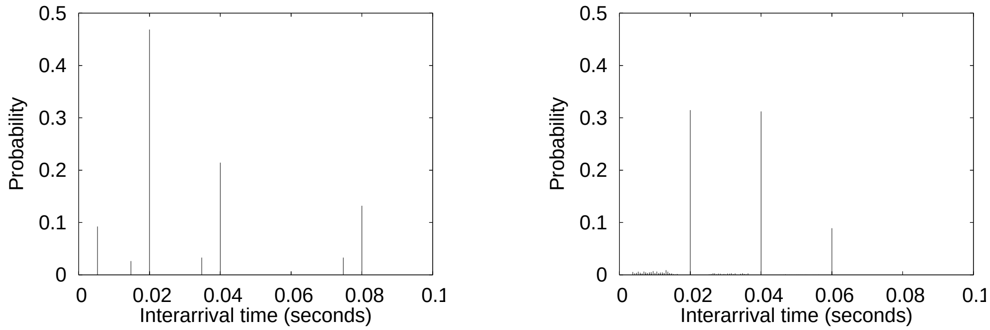
However, the above strategy might not work for the following reasons: (1) application packets might be generated randomly e.g. in the case of event detection (Section 2.3.2) and (2) application packets might be sparse e.g. one packet every 5 minutes from each node [72]. Sparse packets require us to observe for a long time before we get a working prediction model, and offer us little opportunities to re-adjust our prediction.

A more promising approach is to look at the probability distribution of the interarrival times between packets (of all types). A jammer node can use this information to exploit the natural duty cycle of MAC protocols designed for WSNs. We look at LMAC, SMAC and BMAC in turn.

8.3.1 LMAC

One of the characteristics of schedule-based MAC protocols —for example LMAC– is that all communication takes places at fixed intervals. This can clearly been seen in Figure 8.1, where at multiples of the time slot length (i.e. $T_S = 20$ ms) the spikes reside.

There might be one or more spikes before the cluster centred on the slot size, depending on the distribution of the data packet length, i.e. the interarrival time



(1) Data payload is 28 bytes

(2) Data payload ranges uniformly from 16 to 100 bytes

Figure 8.1 — Histogram of message interarrival times for LMAC with a slot size of 20 ms and 20 slots in a frame (Simulation)

between the CM and DM or between DM and the next CM. The attackers objective is to estimate the time slot size by calculating the mean of T_S (Figure 8.2), μ_s, and to jam the beginning of every slot i.e. exactly where the preamble of the CM packet starts. Consequently, the CM gets wasted and the owner of the time slot is not able to address its neighbouring nodes for data exchange.

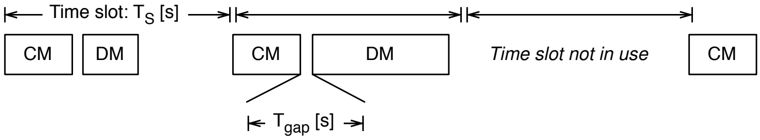


Figure 8.2 — Message interarrival times in LMAC

The estimation algorithm of μ_s is based on two assumptions:

- **Observation of the time slot interval** — Time slot size T_S can indeed be observed, i.e. at least two subsequent time slots are occupied with reasonable probability (i.e. $> \frac{1}{2}$).Using elementary combinatorics, it can be shown that [64]

$$\Pr\{\text{at least two occupied slots are consecutive}\}$$

$$= \begin{cases} 0 & k < 2 \\ 1 - \frac{\binom{n-k}{k}}{\binom{n-1}{k}} & 2 \leq k \leq \frac{k-1}{2} \\ 1 - \frac{2\binom{k-n-1}{n-1}}{\binom{k}{n}} & n = \frac{k}{2} \ (k \text{ is even}) \\ 1 & \frac{k}{2} < n \leq k \end{cases} \tag{8.1}$$

In Equation 8.1, n ($n \geq 4$) is the total number of slots in a frame, and k ($0 \leq k \leq n$) is the number of occupied slots in a frame. When n is even and $k = \frac{n}{2}$, the probability is always larger than 0.5. In other words, for a given n, a node is more likely to observe at least two consecutive occupied slots, if k, the number of occupied slots, at least satisfies Equation 8.2:

$$1 - \frac{\binom{n-k}{k}}{\binom{n-1}{k}} > 0.5 \tag{8.2}$$

For example, if $n = 20$, the smallest k that satisfies Equation 8.2 is $k = 4$. Given that most practical WSNs are dense [4], this requirement is almost certainly satisfied.

- **Distinction between CM and DM** — The shortest DM *might* be shorter than a CM, however the longest DM *must* be longer than a CM

  $$\min(L_{pkt}) \leq L_{\mathrm{CM}} < \max(L_{pkt}) \tag{8.3}$$

  Here L_{pkt} is the random variable representing packet lengths (regardless of packet type), and L_{CM} is the fixed length of a control packet. For example, a CM takes 15 bytes (Section 7.2.1.i), a data packet header takes 7 bytes (assuming minimal information like source and destination IDs of 2 bytes each, data type of 2 bytes and data size of 1 byte), a message authentication code takes 4 bytes, so if a data packet payload is less than $15 - 7 - 4 = 4$ bytes, the corresponding data packet would be shorter than a control packet. This assumption unfortunately bars us from estimating T_S by just measuring the interarrival time between two neighbouring shortest packets, since we can no longer be sure if the two neighbouring shortest packets are both CMs.

The algorithm itself consists of the following steps:

- **Gather packets** — Suppose the observed packets are P_0, P_1, ... Denote the interarrival time between packet P_i and packet P_{i+1} as t_i, and the length of packet P_i as l_i ($i = 1, 2, \ldots$).

  Store t_1, t_2, ... in the ordered set $\mathcal{T}$, and l_1, l_2, ... in the ordered set $\mathcal{L}$. Had there only been DMs and no CMs, the jammer's job would have been easier, since the slot size is then simply

  $$\mu_S = t_i - l_{i+1} + l_i \tag{8.4}$$

  But there are control packets, so the jammer has to continue as follows.

- **Estimate upper and lower bound $\mathbf{T_S}$** — This step is based on two observations. The first observation is that T_S has to be large enough to accommodate both a CM and a DM:

  $$T_S > L_{CM} + \max(L_{pkt}) \geq \min(L_{pkt}) + \max(L_{pkt}) \tag{8.5}$$

The latter inequality is the result of Equation 8.3. Equation 8.5 gives a lower bound for the slot size.

The second observation is that a slot can accommodate at most 2 packets, so it is always smaller than the sum of three contiguous interarrival times, i.e.

$$T_S < \min(t_i + t_{i+1} + t_{i+2}) \qquad \text{where } 0 \le i \le |\mathcal{T}| - 2 \tag{8.6}$$

Equation 8.6 gives an upper bound for the slot size. Denote the lower bound and the upper bound, respectively, as a_0 and a_m. Set $a_0 = \min(\mathcal{L}) + \max(\mathcal{L})$ and $a_m = \min(t_i + t_{i+1} + t_{i+2})$.

- **Estimate time slot size** — Compute the probability mass function of the interarrival times in the interval $[a_0, a_m)$. For this purpose, we can quantize the time interval $[a_0, a_m)$ into m strips, $[a_0, a_1)$, $\ldots$, $[a_{m-1}, a_m)$ and count the number of interarrival times that fall within each strip. Denote the strip with the highest count, i.e. the highest probability mass, as $[a_i, a_{i+1})$ $(0 \le i < m)$.

  Set the estimated slot size μ_S as the mean of the interarrival times that lie in $[a_i, a_{i+1})$. If t_i is the time closest to μ_S, set μ_L, the estimated control packet length, as l_i.

  Note that this estimate might be incorrect if the probability given by Equation 8.1 is not high enough. For example, when in Figure 8.1, the probability density at 40 ms (twice the slot size) is higher than the probability density at 20 ms (the slot size), i.e. two occupied slots are more likely to be separated by an unoccupied slot than to be consecutive, the estimate is double the real slot size. However, this tends to happen only at the fringe of the network, where the network density is lower. Furthermore, if two occupied slots are indeed more likely to be separated than consecutive, the jammer would not miss much by jamming every two slots instead of every slot.

- **Synchronize** — Listen for a packet that is of size μ_L. Once received, transmit a jamming packet, and sleep for $\mu_S - \mu_L$. It is true that a packet of size μ_L might or might not be a control packet, in view of Equation 8.3. If the received packet is indeed a control packet, then the jammer is able to synchronize neatly with the LMAC schedule, allowing it to jam the control packet of every slot. If the received packet is not a control packet however, the jammer's schedule is offset by at least $L_{CM} + \mu_L$, but it is still able to jam the data packet, if there is any, of every slot.

- **Jam** — Wake up, transmit a jamming packet, and sleep for μ_S seconds. Repeat this step until periodic re-estimation is required (the algorithm is executed from top).

We call this algorithm *periodic slot-based jamming* (PSJ). In the reactive version of the algorithm (RPSJ), the jammer listens for a preamble before jamming, instead of jamming proactively.

8.3.2 SMAC

In [62, 44], Law et al. describe an attack strategy for SMAC based on packet interarrival times. Once estimation of the active part of the SMAC duty cycle is established, a malicious node can carefully time its jamming in the active part of the period. In [62, 44], the following SMAC jammers are proposed:

- **Periodic clustering-based jamming (PCJ)** — The jammer estimates the period of the SMAC protocol. It notes when the active part of the period begins, and jams —repeatedly– with an interval of the average packet interarrival time (i.e. during the active period) until the active period finishes. This clustering-based jamming is again initiated in the next estimated active period of the SMAC protocol.

- **Reactive periodic clustering based jamming (RCPJ)** — This is a reactive version of the previous. The RCPJ wakes at the estimated start of the active period and listens to the wireless channel to detect a preamble. If so, it starts jamming. Otherwise, it falls back to the sleep mode in order to conserve energy.

8.3.3 BMAC

The BMAC protocol [83] uses a periodic cycle only for *receiving* and not for sending. Obviously, a jammer node would not be able to observe the periodic behaviour of BMAC. Consequently, the jamming approaches for SMAC and LMAC can not be used directly.

However, it is exactly this periodic listening that the jammer can take advantage of to save energy. Since a BMAC has to listen every, say 10 ms, for a valid preamble, the jammer can be sure that if it samples the wireless channel every 10 ms, it would be able to receive potentially transmitted preambles. Our jamming strategy is therefore to determine the listen interval the victim nodes are using. To achieve this, a jammer node simply has to measure the length of the (longest) preamble. Since the extended preamble in BMAC includes a normal preamble to ensure correct training of the transceiver to the transmitted bit stream, the jammer needs to sample the wireless channel more frequently than the estimated preamble length.

Whenever a preamble is detected, a jammer node can simply interfere the transmission. We call this approach *LPL-based jamming* (LPLJ).

In the next two sections, we will explain how the attacks are simulated before presenting the results. Then, we will explore some potential countermeasures.

8.4 Simulation and evaluation model

The protocols, attacks and countermeasures are simulated using the OMNeT++ framework (`http://www.omnetpp.org`). A simulation consists of a gateway node, $(N_r - 1)$ relay nodes, N_s source nodes and N_j jammer nodes, all capable of a transmission range of r, and located in a square $l \times l$ area. See Table 8.1 for an overview of the parameters.

Table 8.1 — Parameter description

Parameter	Description
N_s	number of source sensor nodes
N_r	number of gateways and (one-hop) neighbours of the gateway
N_j	number of jammer nodes (set to $0.75N_s$ or N_s)
D	network density (set to 15 or 20)
T_{sim}	simulated time in seconds
I	number of simulation runs (different seeds)

The gateway is positioned in the middle of the area, whereas the relay nodes are randomly placed within radio coverage of the gateway. The source nodes and the jammer nodes are pseudo randomly placed more than r away. This is to avoid the jammer nodes having a direct effect on the gateway, giving the malicious nodes an unfair advantage and to allow us to investigate the effect of jamming on the routing of information from the sources to the sink.

We require the node density to be uniform across the simulation area, i.e. $\frac{1+(N_r-1)}{\pi r^2} = \frac{N_r+N_s}{l^2}$, or $l = r\sqrt{\left(1+\frac{N_s}{N_r}\right)\pi}$, to avoid the peculiarities of any specific non-uniform topology having an effect on jamming. In the absence of jammer nodes, the network density is then $D = \frac{(N_r+N_S)\pi r^2}{l^2} = N_r$, i.e. equivalent to the number of gateway and router nodes.

To simulate attacks, the malicious jammer nodes are activated 10 seconds after the sensor network starts operating. This allows the sensor nodes to finish discovering their neighbours and settle down into a steady state before the jamming starts, thereby simulating the attack scenario described in Section 8.2. The total simulation time is T_{sim} virtual seconds. Every experiment is run I times, with different seeds each time. The network topologies are simulated as static. The values of the parameters are summarized in Table 8.2 and are chosen to satisfy the constraints of memory and time available for simulations.

Table 8.2 — Simulation settings

MAC protocol	N_s	T_{sim}	I
LMAC	20	200	10
SMAC	40	600	10
BMAC	20	200	10

8.4.1 Application and networking layers

On the application layer, the sink node broadcasts an interest once it finds a neighbour but it only broadcasts the interest once throughout the simulation. The source nodes each broadcast a matching data every 5 seconds, as an approximation of a network with moderately fast-changing data [15]. The data packet payload ranges uniformly from 16 bytes to 100 bytes.

The minimum payload corresponds to a TinyDiffision [76] payload of 2 attributes (the least number of attributes). The maximum payload is a popular choice [83, 18], and it corresponds to a TinyDiffusion payload of 23 attributes. While a uniform distribution of packet sizes is not realistic, it serves as a base case that allows us to investigate the capability of jammers in reaction to a wide range of packet lengths.

On the network layer, TinyDiffusion [76], faithfully ported from TinyOS[1], is used for SMAC and BMAC. LMAC has a simple built-in routing algorithm, and so LMAC interfaces directly with the application layer.

8.4.2 Data link layer

On the data link layer, SMAC is simulated with adaptive listening [112], with a period of 930 ms and a duty cycle of 10%. The code is also faithfully ported from TinyOS.

LMAC is simulated with a fixed slot size of 20 ms (a little more than enough to fit 100-byte payloads) and 20 time slots per frame (suitable for connectivity of 20).

The BMAC code is built on top of the LPL code from TU Delft's MAC simulator [59]. Following Polastre et al.'s choice [83], we use an RSSI sampling time of 350 μs and a check interval of 100 ms for BMAC. We also implement RTS/CTS signalling on top of the core BMAC protocol. Both SMAC and BMAC use a contention time of 41 ms (the default given by the original SMAC source code).

8.4.3 Physical layer

On the physical layer, the radio characteristics follow those of the RFM TR1001 [89]. The ratio between the power consumption in sleep, receive and transmit mode is 1:960:2400. Transceiver mode switching times are taken into account and the 8-to-12 bit data encoding scheme [88] is assumed.

RF characteristics like interference and gray area effects are not simulated; a departure from actual measurements [87]. However, these are conventionally applied in simulations, as is the case that simulated radio coverage is circular/spherical.

To simulate jamming, the jammer emits a random packet that is *at least* as long as a (normal) preamble, if the jamming starts from the start of the preamble. In reactive jamming, the jamming takes place *after* the preamble. In that case, the jammer emits a single byte (at physical layer level).

We assumed that the integrity of a packet is protected by a message authentication code, and a single corrupted bit is enough to nullify the validity of the packet, therefore corrupting one byte, or 8 bits (i.e. the minimum that can be transmitted conveniently), should be sufficient to corrupt the whole packet. Obviously, the width of the jamming

[1] `http://tinyos.sf.net`

pulse has a large impact on the simulation outcomes, since it determines the energy-consumption of the jammer.

8.4.4 Settings of the jammer nodes

There are several tuneable parameters for PCJ, RPCJ, PSJ and RPSJ. For example, all PCJ and RPCJ implementations start with a minimum sample size of 64 interarrival times, and readjust their estimation every 8 periods. Tuning these parameters allows the attacker to dynamically adjust its behaviour, but the effects of such tunings are left to future investigation. The fact that jammers have limited buffer for storing interarrival times and packet lengths is realistically reflected in the implementations.

The jamming pattern of random jammers is simulated as follows. The jammer sleeps for a random period, followed by a jamming period. Both periods are (uniformly) selected between 1 ms and 500 ms. The jamming period is repeated.

8.4.5 Metrics

Following our previous work [61], we evaluate how *effective* an attack is by measuring primarily the *censorship rate* R_c and secondarily the *attrition rate* R_a.

R_c measures the fraction of messages blocked. Let M be the number of messages arriving at the sink in the absence of attacks, and M' be the number of messages arriving at the sink in the presence of attacks, then

$$R_c = (M - M')/M \tag{8.7}$$

R_a measures the fraction of additional energy the sensor network has to spend in the presence of attacks. Let E be the amount of energy spent when there is no attack, and E' be the amount of energy spent when there is attack, then

$$R_a = (E' - E)/E \tag{8.8}$$

To evaluate how *energy-efficient* an attack is, we use the *effort ratio* R_e, defined as the ratio of the attacker's per-node energy expenditure to the sensor network's per-node energy expenditure when not under attack.

Using R_a and R_e, we can calculate the *lifetime advantage* R_l of a jammer node over a sensor node, that is how long a jammer node can live compared to a sensor node. To derive an expression for R_l, the following assumptions are used: (1) a jammer node has the same total amount of energy as a sensor node, and (2) the power usage is constant.

There are two scenarios. Equation 8.9 is for the scenario where the jammer's energy source is exhausted before the node's

$$R_l = \frac{1}{R_e - R_a} \tag{8.9}$$

Equation 8.10 is for the scenario where the sensor node's power source is exhausted first

$$R_l = \frac{1 + R_a}{R_e} \tag{8.10}$$

See [44] for the derivation of Equations 8.9 and 8.10.

8.5 Simulation results

The censorship rates and lifetime advantages of the various attacks are given in Table 8.3. For the sake of readability, the figures for the attrition rate and the effort ratio are omitted.

We start by making some general observations on the results. First, the censorship rates, both the means and the standard deviations, improve with N_j/N_s and D (defined in Table 8.2). This is in line with intuition: more jammer nodes have greater jamming effect, and the more sensor nodes a jammer nodes has as neighbours, the sooner the jammer node can synchronize with the SMAC/LMAC schedule, or determine the preamble length in BMAC's case. Note that in all simulated cases, there are less jammers than well-behaved nodes. Therefore, a further increase in censorship rate can be expected if we increase the number of jammer nodes, N_j, to the total number of sensor nodes, $N_s + N_r$.

The second general observation is that our jamming algorithms out-perform random jamming and reactive jamming in lifetime advantage R_l as expected since random jamming is not a targeted effort, and reactive jamming consumes energy constantly in listening.

Thirdly, although it appears in Table 8.3 that random jamming is not as effective against LMAC and BMAC as it is against SMAC, the results should be interpreted with caution. In LMAC's case for example, the jammer's random sleep interval, uniformly distributed between 1 ms and 500 ms (i.e. 25 times the slot size), is most of the time wide enough to allow many data packets to pass through. By reducing the jammer's random sleep interval, we can get the censorship rates equally close to 100% for SMAC, LMAC and BMAC.

We do not customize the random sleep interval for each of the protocols in our simulations. However, as the random sleep interval gets smaller, the jammer consumes more energy due to switching between the sleep mode and the transmission mode. This switching energy becomes a dominant component of the overall energy consumption, increasing the jammer's effort ratio. Although we know higher censorship rates can be achieved by customizing the random jam/sleep interval according to the target protocol, we do not do so, because the result only makes random jamming more effective than it already is, but not more efficient.

Below we compare the results in more details.

8.5.1 SMAC

We compare random jamming, reactive jamming with energy-efficient PCJ and RPCJ attacks.

Table 8.3 — Average censorship rates of jamming attacks against SMAC, LMAC and BMAC (results are rounded, standard deviations in parenthesis)

Censorship rate R_c (%)

D	N_j/N_s	SMAC				LMAC				BMAC		
		Rand.	React.	PCJ	RPCJ	Rand.	React.	PSJ	RPSJ	Rand.	React.	LPLJ
15	0.75	92 (15)	93 (22)	75 (17)	59 (17)	78 (13)	94 (5)	96 (3)	80 (9)	85 (19)	93 (9)	93 (9)
	1.00	99 (1)	100 (0)	83 (8)	71 (8)	86 (10)	96 (5)	95 (4)	88 (8)	97 (9)	99 (2)	99 (2)
20	0.75	99 (2)	95 (14)	83 (9)	69 (11)	70 (18)	97 (4)	94 (5)	82 (15)	93 (11)	97 (5)	97 (5)
	1.00	100 (0)	100 (0)	83 (7)	76 (10)	91 (7)	97 (5)	97 (1)	91 (9)	97 (5)	99 (3)	99 (3)

Lifetime advantage R_l (%)

D	N_j/N_s	SMAC				LMAC				BMAC		
		Rand.	React.	PCJ	RPCJ	Rand.	React.	PSJ	RPSJ	Rand.	React.	LPLJ
15	0.75	35 (2)	17 (1)	50 (3)	43 (4)	14 (1)	17 (1)	32 (6)	41 (7)	24 (5)	22 (4)	161 (50)
	1.00	37 (2)	17 (1)	47 (3)	43 (2)	14 (1)	17 (1)	31 (4)	35 (3)	24 (5)	22 (3)	139 (31)
20	0.75	35 (1)	17 (1)	46 (3)	42 (3)	14 (1)	16 (1)	32 (5)	37 (4)	20 (3)	20 (3)	134 (37)
	1.00	37 (1)	17 (1)	44 (3)	43 (3)	14 (1)	16 (1)	29 (5)	34 (4)	20 (3)	19 (3)	126 (29)

8.5.1.i Censorship rate

PCJ and RPCJ achieve substantial censorship rates at 83% and 76% respectively when $N_j/N_s = 1$ and $D = 20$. RPCJ is worse than PCJ because in the proactive approach of PCJ, the victim nodes go to sleep when they fail to access the jammed medium, whereas RPCJ misses more packets as a result of misalignment with the SMAC schedule.

8.5.1.ii Lifetime advantage

At high network density, the lifetime advantages of PCJ and RPCJ are comparable. The lifetime advantage of PCJ, however, decreases with network density. This is because as a jammer gets more neighbours, it fakes more transmissions, thus incurring a higher effort ratio. The lifetime advantage of RPCJ on the other hand hardly changes with network density, because an RPCJ jammer spends more time listening than transmitting, but the time spent on listening is roughly the duration of the listen interval, which does not change with network density.

8.5.2 LMAC

We compare random jamming, reactive jamming with PSJ and RPSJ.

8.5.2.i Censorship rate

The censorship rates of PSJ are impressively close to those of random and reactive jamming. RPSJ is worse than PSJ, but both PSJ and RPSJ are more effective against LMAC than PCJ and RPCJ are against SMAC, because the slot size can be estimated more accurately than the period of an SMAC schedule.

8.5.2.ii Lifetime advantage

Both PSJ and RPSJ have twice the lifetime advantages of random jamming and reactive jamming. Looking more carefully, RPSJ has a higher lifetime advantage than PSJ. This is because not all slots in a frame are necessarily occupied, and when a slot is unoccupied, the energy RPSJ spends on listening is lower than the energy PSJ spends on transmitting.

As the network becomes denser, more slots in a frame are occupied, the effort ratios of both PSJ and RPSJ become higher, and hence their lifetime advantages decrease with network density. This trend should stop when all the slots in a frame are occupied.

Random jamming does not make LMAC nodes monitor the channel more than it usually does, unlike the case with SMAC. The sensor nodes only listen for at most a small fraction of the slot size, and as packets are jammed, less energy is used on propagating the packets to the sink, resulting in a negative attrition rate. One note of caution though: had the jamming been allowed to start *before* the LMAC nodes synchronize with each other, the results would have been different. Then the LMAC nodes would have had to constantly listen for broadcast schedules, and this would have resulted in a large positive attrition rate, and, hence, lifetime advantage.

To summarize, PSJ and RPSJ have high censorship rates and lifetime advantages. Coupled with ease of implementation, they are genuine threats to LMAC even when the packets are encrypted.

8.5.3 BMAC

We compare random jamming and reactive jamming with LPLJ. LPLJ is by design reactive jamming with optimized listening, so it is only intuitive that LPLJ has similar censorship rates as reactive jamming attacks. But it has far higher lifetime advantages than reactive jamming attacks. The lifetime advantage of LPLJ however decreases with network density due to the following reason.

As the network gets denser, a jammer gets not only more sensor nodes but also more jammer nodes as its neighbours. More neighbouring sensor nodes mean more packets to jam and more neighbouring jammer nodes means staying awake more often. This is because whenever a carrier is detected on the channel, a jammer node always stays awake for a while to listen for a valid preamble. Consequently, a jammer node has a higher effort ratio in denser networks, and —according to Equations 8.9 and 8.10– the lifetime advantage becomes lower. The lifetime advantages of random jamming are comparable to reactive jamming attacks i.e. significantly lower than those of LPLJ.

To summarize, since LPLJ is trivial to implement and yet allows the jammers to live as long as, or longer than the victim nodes. Hence, it is a devastating threat to BMAC even when the packets are encrypted.

8.6 Implications to other protocols

We now look at the implications of the above findings to other protocols. As explained in Chapter 3, we concentrate on the three main classes of MAC protocols for WSNs: (1) contention-based, (2) schedule-based and (3) preamble sampling. Examples are SMAC, LMAC and BMAC, respectively. We now look at other protocols that belong to each of the three categories, starting with contention-based protocols. We pick these protocols from [59].

8.6.1 Contention-based protocols

Contention-based protocols include TMAC [18]. Since TMAC is derived from SMAC, PCJ and RPCJ are applicable to TMAC. The fact that TMAC has a dynamic duty cycle offers some relief because even though PCJ and RPCJ are able to adapt to the dynamic duty cycle through periodic readjustments, they would be less effective and less efficient against TMAC than against SMAC due to the need for more frequent readjustments.

8.6.2 Schedule-based protocols

Schedule-based medium access control protocols include e.g. TRAMA [102], μMAC [9] and SS-TDMA [58]. The primary means of jamming these protocols is by way of estimating the slot size. To estimate the slot size, the jammer needs to be able to observe two consecutive slots, that is, the number of slots and the number of occupied slots in a frame need to satisfy Equation 8.2. This requirement is typically satisfied as explained in Section 8.3.1 and will not be mentioned again in the discussion below.

SS-TDMA relies on the nodes being arranged in a rectangular or hexagonal grid. Due to the lack of control packets, SS-TDMA is easier to attack than LMAC since Equation 8.4 alone allows us to estimate the slot size.

TRAMA and μMAC divide time into alternating periods of random access and scheduled access. A slot in the random access period is called a signalling slot, while a slot in the scheduled access period is called a transmission slot. Denote the size of a signalling slot as $T_{\text{signalling}}$, and the size of a transmission slot as T_{Tx}, then for ease of synchronization, T_{Tx} is typically a multiple $T_{\text{signalling}}$, e.g. $T_{\text{Tx}} = 7T_{\text{signalling}}$ [102]. Due to the fixed length of the time slots, Equation 8.4 can be used to estimate $T_{\text{signalling}}$ and T_{Tx}.

8.6.3 Preamble sampling protocols

Preamble sampling protocols include e.g. low power listening [46] and WiseMAC [28]. Low power listening is to all our intents and purposes equivalent to BMAC, so we will not discuss it any further. WiseMAC uses the same preamble sampling scheme as BMAC, with the difference being if the sender knows the schedule of the receiver, it waits until the receiver is about to wake up and sends its packet with a normal, shorter preamble, instead of an LPL preamble. However, for broadcasting packets, the sender often has to stretch the preamble to the full length of the LPL preamble [59]. Therefore, given enough broadcast traffic, the jammer is still able to figure out the check interval and apply LPLJ.

8.6.4 Discussion

MAC protocols organize the access to the wireless medium. From attack point of view, this organization is one of the weaknesses that can be exploited.

Among the protocols, schedule-based protocols have better resistance to energy-efficient jamming because they spread out transmissions in time. Fortunately, more schedule-based protocols than any other type of protocols have been proposed in literature. In the next section, we explore some countermeasures.

8.7 Countermeasures

In this section, we discuss potential countermeasures against attacks for each of the categories of MAC protocols.

8.7.1 SMAC

Since our attacks against SMAC are based on clustering [44], a countermeasure would naturally be to prevent clustering-based analysis from being feasible. This can be done by narrowing the distance between distinctive clusters of interarrivals i.e. setting the SMAC duty cycle to 50% [44].

According to simulations with $N_j/N_s = 1$ and $D = 20$, this defence is modestly effective against PCJ if K-means is used as the clustering algorithm. The resultant censorship rate is reduced to 38% (with standard error 17%). However, when expectation maximization (EM) [20] is used as the clustering algorithm, the censorship rate can only be reduced to 75% (with standard error 4%).

Relief can be found in the fact that a jammer using EM would considerably deteriorate its own lifetime advantage because EM is a computationally expensive and therefore energy-consuming algorithm that involves multiple exponentiations.

On the other hand, a duty cycle of 50% may not be energy-efficient enough for most WSNs that are characterized by low data rate [83]. Overall, using a high duty cycle is a partial countermeasure to energy-efficient link-layer jamming.

8.7.2 LMAC

In the case of LMAC, we mentioned that it is advantageous to spread transmissions out in time. But as long as we transmit at fixed slot sizes (i.e. fixed intervals), spikes would manifest on the histogram of the packet interarrival times. The strategy of flattening the spikes to increase the difficulty in estimating the slot size can be served by changing the slot size pseudo randomly as a function of time and a hidden seed. Note that this method has a negative impact on the bandwidth of the protocol.

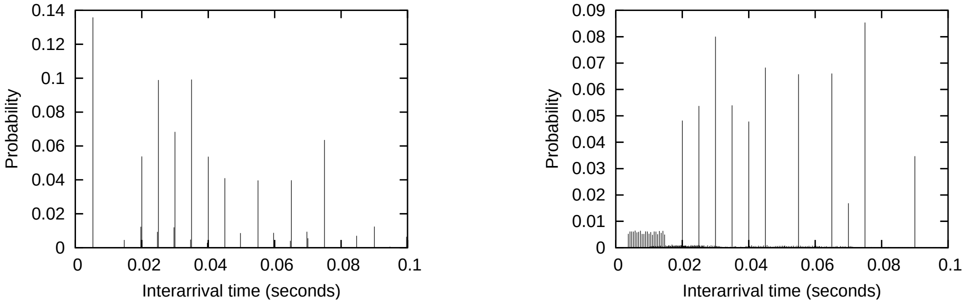
For example, if the sensor nodes change their slot size every second, by pseudo randomly picking a value from the range [20 ms, 30 ms], then according to simulations, a jammer using PSJ can still achieve a censorship rate of 80% (with standard error 12%) and a lifetime advantage of 47% (with standard error 5%) when $N_j/N_s = 1$ and $D = 20$.

When we change the slot size —on a per slot basis– by incrementing the size with 25% until two times the original size is reached and then decrement with 25% until the original size again, the censorship rate of a PSJ jammer remains 80% (with standard error 14%), but the lifetime advantage decreases to 40% (with standard error 5%). The distribution of packet interarrival times is shown in Figure 8.3.

What is remarkable about this countermeasure is the RPSJ censorship: only 35% (with standard error 15%). The reason for this effect is that the RPSJ method of attacking listens for a preamble before jamming, a jammer is only efficient when it starts to receive *just before* a packet is transmitted. When it awakens during the transmission of a packet or after —what we try to achieve with this countermeasure– it waits in receive mode until the next preamble, hence wasting energy. The PSJ attack does not suffer from this effect.

Note that the above described countermeasure has deterministic behaviour, which makes it less suitable. However, in Section 8.8 we implement it on sensor node prototypes to show its effect in practice.

The above described countermeasures are less than satisfactory. We succeed in fouling the attackers in the estimations they make of the slot size by flattening out spikes in the probability distribution of interarrival times. In the total picture, however, this misses its effect. The fact that PSJ attackers have more than one opportunity



(1) Data payload is 28 bytes

(2) Data payload ranges uniformly from 16 to 100 bytes

Figure 8.3 — Histogram of packet interarrival times for LMAC with dynamic changing of time slot size

to jam a packet —due to the multi-hop communication to the sink node– makes them robust and effective in jamming.

In the following scenario, we make the LMAC protocol more persistent in delivering messages by doubling the number of retries the LMAC protocol does before discarding packets. As a matter of fact, LMAC without countermeasure and the number of retries set to 6, has no effect on censorship rate. With countermeasure, however, the PSJ censorship rate R_c drops to 65% (with standard error 14%). When we further decrease the effect of the attackers by reducing them in number, the censorship rates drop to 51% (with standard error 9%) and $R_c = 38\%$ (with standard error 25%), for $N_j/N_s = 0.75$ and $N_j/N_s = 0.50$, respectively. The conclusion is that an attacker should not be thrifty with deploying jammer nodes.

In the above discussed results, the data packet payload ranges uniformly from 16 to 100 bytes. Large payloads give the PSJ attacker more opportunity (i.e. higher probability) to jam the packet. We simulate attacks with fixed data packet payload of 28 bytes. The censorship rates for PSJ are $R_c = 49\%$ and $R_c = 29\%$, for $N_j/N_s = 1.00$ and $N_j/N_s = 0.75$, respectively. Thus to avoid effective attacks, shorter data packets should be preferred.

8.7.3 BMAC

For BMAC, it is not clear how to prevent attacks, since BMAC relies on the preamble to be long enough for the receivers to detect. Shortening the preamble any further than what we have simulated, 10 ms (which is the minimum considered by Polastre et al. [83]) defeats the purpose of BMAC.

8.7.4 Discussion

From the above discussion, it appears that effective countermeasures are lacking. By comparing the censorship rate and lifetime advantage of the best attack on the respective protocols, LMAC emerges as a better choice than SMAC and BMAC in terms of resistance against link-layer jamming. In the LMAC countermeasure, transmission intervals are (pseudo) randomized, which hardens the jammer's task of estimating the timing of the protocol. Generalizing this observation, schedule-based TDMA protocols are potentially a better choice than other types of protocols, since randomness can easily be superimposed on the schedule.

This concludes the presentation of our simulated attacks and countermeasures for the three main classes of MAC protocols.

8.8 Validation

To validate our simulation results, we implement LMAC and its most effective jammer (PSJ) on prototype sensor node hardware.

8.8.1 Implementation

For practical reasons, the following changes with respect to our simulation model are considered. Firstly we test the effectiveness of jamming and countermeasure in a single hop network. The jammer node is in radio range of the sink node and therefore we expect the censorship rate to be 100% for an effective jammer.

Secondly, we consider a three-node network (i.e. one sink node and two well behaved nodes), which uses a frame length of four slots (i.e. the probability of two consecutive slots is 1). Each slot takes 1/16s, roughly a factor 3 larger than in simulation[2], but is more practical to implement, since it allows for sufficient time to process the packets and to prepare the transceiver to transmit the data. These steps are not considered in our simulations, but cannot be omitted on our prototype sensor nodes.

Thirdly, our prototypes are based upon the hardware described in [39], however the RFM TR1001 transceiver is exchanged for a Nordic nRF905 transceiver [81] for the following reason. The Nordic transceiver releases the processor from time critical controlling, like the generation of the preamble and the alignment of the incoming bit stream. This results in a better performance of the communication between nodes. However, the transceiver has slightly different properties as are assumed in our simulation model. The transceiver denies us the opportunity to transmit the short jamming packets as proposed in Section 8.4. The jamming is done by transmitting a packet consisting of a preamble, address, payload of 1 byte and a checksum. A smaller packet cannot be transmitted by the transceiver hardware.

Finally, we let the PSJ observe the packet interarrival times for control packets only, due to two reasons: (1) the used transceiver filters the packets on type, length and validity before handing over to the processor and (2) our countermeasure for LMAC leaves the gap T_{gap} between control and data packet untouched. Therefore, an attacker might estimate this gap accurately, since it would show up in the probability distribution of packet interarrival times and might —based on this information– distinguish between the two packet types. We let the PSJ observe 8 interarrival times before letting it estimates the slot size.

The implementation of the PSJ attacker consumes —on top of the DCOS kernel and drivers [25]– 1520 bytes of program memory including debugging code that allows us to get insight into the estimated slot interval via serial link to the jamming node.

[2] For the simulations in this section, we use timing as discussed here.

In our experiments, the sink node keeps track of correctly received data packets by recording their sequence number. This allows us to detect missing packets and thus to calculate censorship rate R_c. Both remaining well-behaved nodes transmit 16 byte data packets every second. When a data packet is not acknowledged by the sink node, it is scheduled to retry. However, when it fails to be delivered within three retries, the packet is simply discarded. In a test without an attacker, we verified that all generated data packets arrived in the sink node.

8.8.2 Experiments

First, we test the effect of jamming in on a LMAC setting without countermeasures. The experiments show that jamming is effective; no communication between nodes and sink is possible when the jammer is active (i.e. $R_c = 100\%$). Plugging in a similar setting in our simulation model, results in $R_c = 97\%$. A closer inspection shows that the 3% difference is caused by the packets that reach the sink before the simulated PSJ attacker has established an estimate of the slot size.

When the PSJ estimates the slot size incorrectly, the nodes are able to communicate again after the transmission of the jammer has drifted to the gap between data packet and the control packet of the next slot, missing its jamming effect. This effect occurred in one of ten experiments. It implies that the jammer node must resynchronize frequently, to ensure effective jamming. Our simulation model does not account for drifting.

Another observation that we make is the following. When the nodes are closely situated together and the attacker is relatively far away from the cluster, the LMAC packets are *not* disrupted by the jammer node while it is still in radio range. Apparently, the demodulation scheme of the transceiver is robust against interfering signals with lower signal strength. This makes it harder for jammers to compromise a wireless sensor network, since it requires them to transmit at a high power level or to use specialized hardware. As we already remarked in Section 8.4, we omitted interference and the gray area effect in our simulations. However they have a clear influence on the results.

In the next experiment, LMAC is extended with the countermeasure discussed in Section 8.7.2. For this purpose, we extend the control packet of LMAC with one byte used as "hidden" seed for the countermeasure. At the end of each slot, this value is increased by one, advancing to the next table entry that contains the 25% increment and decrement steps of the slot size. By receiving one control packet correctly, a node is able to calculate all future steps of the countermeasure scheme.

Compared to the simulations in Section 8.7.2, our implementation on sensor nodes uses only a small portion of the slot to actually transmit the control and data packets. This severely reduces the probability that an attacker jams a packet, especially when its estimate of the slot size does not match reality. Therefore, our countermeasure against the PSJ attack is —in this setting– quite effective. We observe a censorship rate of $R_c = 2\%$ (32 of 1532 data packets were successfully jammed). In the trace of successfully received messages in the sink, we see that the jammer is sometimes successful in disturbing a packet. However, in most cases the acknowledgement mechanism in LMAC is able to recover the packet in one of the retries. Similar effects are observed in the simulation results of this scenario. Our simulator model reported a censorship rate of $R_c = 0\%$.

8.8.3 Discussion

Our experiments validate our simulations on the following aspects: (1) packets can be nullified with short jamming packets and (2) censorship rates match with the experiments in a small network. Although the validation of the simulation model is not complete, some interesting points have been brought to our attention: (1) processing delays, (2) radio aspects (i.e. demodulation scheme, gray area and interference) and (3) timing imperfections of jammers cannot be neglected. These effects are not taken into account in our simulation model. Our future work will extend and validate the simulation model in more detail.

In addition, we showed that LMAC without countermeasure can effectively be jammed. This shows the importance of keeping potential link-layer attacks in mind when designing a MAC protocol.

8.9 Related work

Wood et al. wrote in 2002 that no effective defence was yet known against link-layer jamming [106]. Xu et al. [109] propose two evasion strategies against constant jammers: (1) channel surfing and (2) spatial retreat. Channel surfing is essentially an adaptive form of frequency hopping. Instead of continuously hopping from frequency to frequency, a node only switches to a different frequency when it discovers the current frequency is being jammed.

We mentioned the four generic jammer models by Xu et al. [108] earlier. Based on the models, Xu et al. show that either received signal strength indication (RSSI) or carrier sensing time alone is not sufficient in detecting all the 4 types of jammers. Instead, attacks can be detected by measuring (1) both the packet delivery ratio and the signal strength, or (2) both the packet delivery ratio and the location.

Instead of looking at the packet delivery ratio of individual nodes, we consider the effect of distributed jammer nodes on the WSN as a whole. In our opinion, apart from jamming effectiveness, a jammer cares about jamming efficiency. For this reason, while Xu et al. provide metrics for measuring the quality of service of individual nodes, we provide metrics for measuring the jamming effectiveness and efficiency of the jammers. Xu et al.'s and our work are complementary in the sense that Xu et al. target jammers at the physical layer, while we target jammers that aim to gain more edge by exploiting the data link layer. This work extends previous work [61] to more protocols and in a more general setting (the latest attacks work even on encrypted traffic, while the earlier attacks do not).

8.10 Conclusion

In this chapter, we propose new link-layer jamming algorithms that are based on *minimal* knowledge of the target protocols. The effectiveness and energy-efficiency of the minimal knowledge attacks is almost as good as the effectiveness and energy-efficiency of the detailed knowledge attacks. In addition, the minimal knowledge attacks are effective when data packets are encrypted. We used metrics proposed by Law et al. (censorship rate, attrition rate, effort ratio and lifetime advantage [61]) and a method to quantify the effect of link-layer jamming.

We implemented LMAC and its most effective jammer on sensor node prototype hardware to validate our simulation model. The censorship rates matched our simulations in a four-node topology. With our experiments we pinpointed that (1) timing aspects, like drift in jammer nodes, and (2) RF aspects (i.e. demodulation scheme, gray area and interference) deserve more attention in our model.

With typical WSN systems in use today no effective measures against link-layer jamming are possible. For WSNs that require high security against link-layer jamming we recommend (1) encrypting link-layer packets to ensure a high entry barrier for jammers, (2) the use of spread spectrum hardware, (3) the use of a schedule-based protocol, and (4) the use of randomized transmission intervals.

Chapter 9

Conclusions

The technology that lets tiny and smart devices create their own network, allowing them to transport sensor data while requiring little power and transmission range is potentially the next big thing to happen [45]. Recent advances in sensor technology, low-power analogue and digital electronics, and low-power radio frequency designs have enabled the development of these cheap, small, low-power sensor nodes, integrating sensing, processing and wireless communication capabilities.

The objective of wireless sensor nodes is twofold: (1) obtain a description of the physical surroundings by means of sensors, and (2) wirelessly communicate this description and assist other nodes to deliver descriptions. To carry out these two functions, a wireless sensor node is typically equipped with the following (functional) components: sensor(s), transceiver, power source, processor, memory and debugging/testing interfaces. Following this concept, we design two prototype hardware platforms that have been successfully used in several research projects.

In the vision of wireless sensor networks, sensors collaborate to be able to cope with the environment: they operate completely wireless, and are able to spontaneously create an ad hoc network, assemble the network themselves, dynamically adapt to device failure and degradation, manage movement of sensor nodes, and react to changes in task and network requirements. In other words, wireless sensor networks must be self-organizing.

Common characteristics of WSN applications include long lifetime requirements, dynamics in network topology, high peak loads and in-network intelligence. Due to the many envisioned applications for WSNs and their broad spectrum of requirements, wireless sensor networks are application specific networks, i.e. the network is optimised —for example by cross-layer optimisation– to improve its efficiency. Inherently, wireless sensor networks differ significantly from traditional computer networks. The required level of efficiency in wireless sensor networks justifies the design of specific protocols for this class of networks.

In this thesis, we discuss medium access control (MAC) for wireless sensor networks. The MAC protocol is a set of communication rules, which determines when a node is allowed to transmit and when it should receive. As such, the protocol is in direct control of the communication modality, i.e. RF transceiver of the wireless sensor. Our research is motivated by the observation that the MAC protocol has a strong influence on energy consumption of —typically battery operated– wireless sensors. This observation is augmented with current consumption measurements using the designed prototype platforms. In wireless sensor networks, the MAC protocol is required to be energy-efficient in order to extend the node lifetime.

In the massive networks, the wireless sensors are not only responsible for propagating their own sensor readings, but they are also required to assist other nodes in forwarding data. This is necessary, since not all nodes in the network are likely to be within communication range of the point where the interest in the sensorial information is. Thus, even though nodes might not require advertising data generated by themselves, they are still needed to create a connected structure in order to enable information flow from the far end of the network to where this information is required. Consequently, the wireless communication must be well organized; a certain challenge in large-scale multi-hop networks such as wireless sensor networks. Nodes must not interfere with other transmissions and, at the same, time must utilize the shared wireless channel optimally by allowing spatial reuse of the medium. The energy problem and communication challenges congregate in the data link layer, making it a relevant research topic. In this thesis, we present medium access solutions that match wireless sensor network requirements.

Channel sharing method — We select schedule-based medium access —a sub-class of time division multiple access (TDMA)– as the most appropriate channel sharing method for wireless sensor networks. In schedule-based medium access, each node uses its receiving and transmitting functionality according to a schedule. As such, the medium can be shared conflict-free and high peak loads can be sustained without inducing energy-wasting collisions. These are requirements for wireless sensor networks; although overall data rates are low in wireless sensor networks, high peak loads occur due to the following reasons: (1) simultaneous reporting of events, and (2) broadcast storms of route discovery. Therefore, we conclude that schedule-based access is a viable method of channel sharing for wireless sensor networks.

Self-organization, message delay, scalability and synchronization requirements are often referred to as weak points of schedule-based medium access. We develop a robust scheduled medium access approach, which addresses these weak points.

Scheduling principle — With an eye to limited hardware capabilities of wireless sensors, the proposed medium access scheme is kept simple. Time is organized into time slots, which are grouped into frames. Each frame has a fixed length of a (integer) number of time slots. Each node takes control of (at least) one time slot and a node is allowed to transmit packets during this time slot. During time slots of other nodes, a node receives packets when it is addressed. Otherwise, the node switches its transceiver to low-power mode in order to conserve energy.

Self-organization — To ensure scalability, self-organization is key in wireless sensor networks. Additionally, nodes lack sufficient memory to obtain and maintain a global view of the network. Thus, operation is required to be localized.

We develop a mechanism that allows nodes to decide which time slots can be used without causing interference to other nodes, based upon local information only. Nodes autonomously select (randomly) a time slot when it is not used in a two hop radius. A similar distance between concurrent transmissions is applied in the handshaking mechanisms of contention-based medium access. Consequently, our MAC protocols do not suffer from the hidden terminal problem.

Implicitly, our mechanism allows the medium to be spatially reused, which is beneficial for transport capacity of the network. Experiments show that the wireless medium can indeed be simultaneously used if co-channel interference conditions are met.

From a graph theoretical perspective, assigning time slots to nodes is comparable to the E^2 graph colouring problem. The number of required time slots is dependant on maximum network connectivity Δ. We show that the required (optimal) number of time slots is bounded between $\Delta + 1$ and $\Delta^2 + 1$ to give each node in the network opportunity to control at least one time slot. We verify that our local time slot assigning algorithm meets the factor 3 approximation from the optimal number of required time slots, as is suggested in literature.

A novel approach in the presented work is self-organizing reuse of the wireless medium in combination with scheduled medium access.

Robustness — During initiation of the network, many nodes potentially enter the time slot choosing process. Since communication between these nodes is not yet possible, nodes might choose identical time slots. This obviously leads to schedule conflicts. Additionally, conflicts can occur in dynamic network topologies, e.g. due to node mobility.

We propose and verify a mechanism to resolve these schedule conflicts. Our solution depends on the detection of collisions. Experiments showed that indeed the cause of lost packets can indeed be determined. Once collisions are detected, nodes report the collision to the nodes that have a schedule conflict. Next, the nodes with schedule conflict reconsider their time slot choice. With formal methods, we demonstrate that conflicts are resolved when they are detected. We conclude that our schedule-based medium access targets conflict-free operation and is self-healing in dynamic topologies.

Message delay — When the number of time slots per frame is large, nodes have to wait a long time before their time slot comes up for the opportunity to transmit. This increases the reaction time of the network considerably: an undesired effect. To limit latency, it is key to keep the number of time slots in a frame to a minimum. However, we mention that in practical networks a thrifty number of time slots should not be applied, e.g. for mobility or iterative deployment arguments. A balance should be established between the mobility/iterative deployment requirements and latency requirements.

We present latency-reducing strategies for LMAC based upon alteration of the time slot selection mechanism. We observe that latency can be reduced when a source node transmits just before a sink is going to transmit. The sink node receives the message and is able to immediately forward it to the next node. This reduces the time that the message resides in the parent, and results in better message delay performance. Simulations showed that messages on average travel six hops per MAC frame –a reduction by factor 3.

Scalability — We combine schedule-based medium access control with backbone creation that identifies redundant wireless sensors. These redundant sensor nodes (i.e. passive nodes) are not required in the multi-hop forwarding and can thus save energy by following a sleep pattern. This backbone maintains a connected and dominating structure that keeps the overall network operational.

Besides its attractiveness for energy consumption considerations, the backbone creation has an important advantage in densely deployed networks. At initial deployment of the network, all nodes try to obtain a time slot. Recall that the frame size needs to be tuned to the maximum connectivity in the network. As a result, many time slots per frame are required in dense networks and consequently message delay is significant. However, when using the backbone creation, the number of required time slots can inherently be reduced, because many nodes realize their redundancy, give up their time slot and become passive.

Furthermore, we design the EMACs protocol in such a way that passive nodes can communicate —although not guaranteed collision-free– with active nodes creating the backbone. The autonomous backbone creation solves the scalability drawback of schedule-based medium access.

The communication protocol (EMACs) is implemented and evaluated both in a simulation environment and in a real-world testbed. It significantly increases the network lifetime compared to an existing communication approach.

Synchronization — We introduce the concept of gateway nodes. Gateway nodes fulfil a bridging function between the wireless sensor network and the user. As advocators of interest for sensor readings, we give these nodes an initiating role in setting up the WSN. Additionally, timing in the network is relative to the timing of gateways.

A key issue in scheduled medium access is that it needs a common sense of timing in order to create a long-lived network. Without precise (local) synchronization, nodes have to use long guard intervals and time outs to ensure that receivers are ready when transmitters start transmitting, wasting valuable energy. The more accurate the synchronization is, the smaller tolerated timing differences can be and the lesser the energy-wastage.

From real-life experiments, we conclude that nodes can maintain relative synchronization when timing is updated regularly. Consequently, guard intervals can be kept small. The updating of timing is in the discussed prototype wireless sensor demanding minimal energy consumption (i.e. energy costs of a few microcontroller cycles) and is not crucial in timing itself.

Concrete protocol designs — Based upon our general scheduling approach, we have designed two concrete MAC protocols: EMACs and LMAC.

EMACs includes autonomous backbone creation and solves the scalability drawback of schedule-based medium access. In this thesis, we have motivated cross-layer optimization as viable approach to extend the lifetime of the wireless sensor network. We demonstrated this by tight integration of EMACs and an existing message routing protocol.

LMAC characterises itself by inclusion of routing to gateways, i.e. the preferred destination of messages in WSN.

Existing MAC protocols for WSNs often operate independently of the application e.g. the injected queries or the kind of data that flows through the network. Additionally, the closer nodes are to a gateway, the more data they have to forward. In this section, we present mechanisms to make LMAC adaptive to the volume requirements of the actual sensor application.

We adapt the LMAC protocol by tuning the number of time slots a node controls i.e. the more time slots a node controls, the greater its share in the channel capacity and the more data it can transport. This scheme is particularly useful when the amount of transported data can be predicted. The prediction is translated into a (discrete) number of time slots that would be required to transport the data and, together with fairness considerations, this is used to obtain more or release time slots. The rules for obtaining time slot(s) are not altered.

We conclude that there are two important parameters which determine whether EMACs or LMAC has to be used: (1) network connectivity, and (2) payload. EMACs with active/passive roles is more efficient concerning (average) power consumption of the transceiver than LMAC in highly connective networks, however, a prerequisite is that the payload of (passive) nodes should be very low. EMACs is not resilient against high (peak) loads generated by passive nodes.

We have showed that both protocols are implemented on prototype wireless sensor platforms using small program memory footprints. We conclude that both protocols have practical value for WSNs.

In simulation, both EMACs and LMAC perform better —in considered scenarios– on energy-efficiency and delivery ratio than state of the art MAC protocols designed for WSNs.

Attack and defence — We demonstrate that MAC protocols in use today in WSNs typically do not have any protection against attacks. The WSN can easily be compromised. We showed that energy-efficient attacks are possible with minimal knowledge on the MAC protocol in use. Security needs more attention. For the LMAC protocol, we have proposed an effective, energy-efficient jamming attack strategy. Its effectiveness has been demonstrated by simulation and in implementation on prototype wireless sensors.

Ultimately, wireless sensor networks help us complete our daily tasks or even make our lives easier and safer. Nowadays, the vision of wireless sensor networks appears in a variety of often small application trials. However, the technology gets more and more accepted by industry, e.g. [57, 93]. In this thesis, we have contributed to the realization of the wireless sensor network vision by studying efficient medium access control.

Appendix A

Prototype wireless sensor designs

We present two prototype wireless sensor platforms: (1) the μNode and (2) the SmartTag. The μNode design is well apt for interfacing with many sensors, either digital or analogue, testing communication protocols and debugging WSN applications. The SmartTag design is attractive due to its small size, but is less suitable for debugging. Typically, only one (digital) sensor can be connected to this design.

A.1 Motivation

In Chapter 2, we focused on the functional requirements of wireless sensor networks. We have described several applications and the demands they impose. In this appendix, we describe practical aspects of wireless sensor networks. We design two hardware platforms as prototypes for wireless sensor nodes. The designed platforms are successfully used in several research projects, like CoBIs (EU IST-2003-4270) and Smart Surroundings (Dutch BSIK program).

One of the main reasons for the growing interest in wireless sensor networks nowadays, is the burgeoning of low-cost, low-power and reliable components that provide the envisioned functionality. In our hardware designs, we benefit from this trend by combining several components to versatile platforms. Based on the designed platforms, we establish an energy consumption model representative for wireless sensor networks.

The reasons for designing prototype WSN platforms are manifold. Firstly, the gap between theory and practice needs to be closed. Prototype platforms are a convenient means of testing and debugging new networking protocols, system software or drivers for sensors, because these platforms are typically not optimized with respect to resources and size. Secondly, prototype platforms can assist in quickly comparing the performance of protocols in certain applications or even give feedback on the application specifications. And thirdly, once protocols and sensors have been selected for an application, prototype hardware provides insight into required resources, which is valuable information for designing an application-specific (and cost effective) wireless sensor. We already stated in Section 2.3.1 that cost effectiveness and node lifetime are challenges.

In Section A.2, we discuss a general architecture for wireless sensor nodes. In this thesis, we used the developed hardware platforms for testing and debugging networking protocols, especially the MAC protocol (Sections 6.7, 7.2.4 and 8.8).

A.2 Functional architecture of wireless sensor nodes

The objective of wireless sensor nodes is twofold: (1) obtain a description of the physical surroundings by means of sensors, and (2) wirelessly communicate this description and assist other nodes to deliver descriptions. To carry out these two functions, a wireless sensor node is typically equipped with the following (functional) components (Figure A.1): sensor(s), transceiver, power source, processor, memory and debugging/testing interfaces.

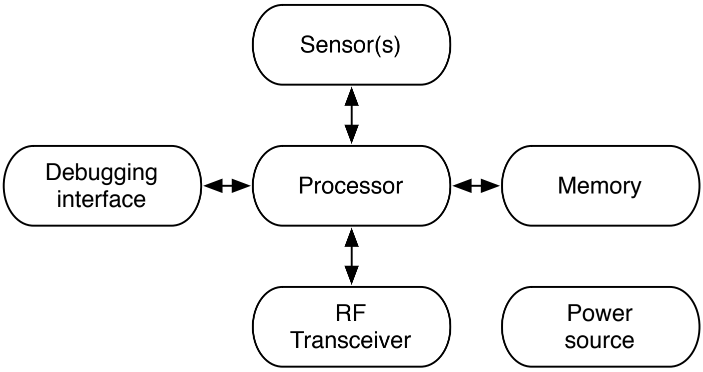


Figure A.1 — Functional architecture of wireless sensor nodes

Nowadays, we find these components separately on the available WSN platforms (discussed in Section A.3). However, integration of functional components is an often seen tendency, as will be the case with sensor nodes. In [52], Kahn et al. even present ideas on a ”grain” sized sensor node, which has dimensions in the order of a few cubic millimetres, including the power source. Many of the wireless sensor functional components have already been miniaturized, yet a truly integrated device needs still be awaited.

A.2.1 Sensors

Sensors form the "ears and eyes" to the outside world of the node. They monitor physical phenomena and give the node contexts about its surroundings. In this section, we discuss shortly what interfacing a node should be capable of, in order to obtain sensor readings.

First required capability is the provisioning of a *power source* for the sensor(s). Most sensors require energy to perform their measurements. Depending on the interval between measurements and the idle current of the sensor, we require nodes to be able to shut down the power to the sensor completely, certainly when the power consumption is a significant part of the total energy consumption. For example, the ECHO-TE sensor [26] discussed in Section 2.2.1.iii to measure soil moisture, has a current consumption of 9 mA. If this sensor would be continuously powered, it would drain two AA batteries, with an estimated capacity of 2 Ah, in roughly nine days. This is far from the envisioned lifetime of several years on a single set of batteries and sketches the need for power control of sensors.

Sensors provide their readings either analogue (i.e. current or voltage) or digitally. Sensors with analogue output require the node to be equipped with an analogue to digital converter in order to be able to further process the sensor input. Most low power processors nowadays have built in analogue to digital converters (ADC). Examples are Texas Instruments' MSP430-series (`http://www.msp430.com`) and Atmel's ATMega-series (`http://www.atmel.com`) processors. With few additional electronics, these processors can interface with a broad range of analogue sensors.

However, more and more sensors are becoming available with built-in ADCs. These sensors are typically controlled by a (bi-directional) digital interface like SPI or I^{2}C and are often pre-programmed with calibration data. In addition, most digital sensors apply averaging of samples to provide a reliable sensor output. This makes the use of this type of sensors very attractive. For example, the Sensirion SHT71 relative humidity and temperature sensor (`http://www.sensirion.com`) waits in the low-power standby mode a command of the host it is attached to. When it receives the command to sample, the digital circuit in the sensor powers on the analogue part of the sensor, obtains a sample, applies calibration factors and signals the host that the sensor value is ready. The energy-consuming analogue part of the sensor is automatically switched off after the sample has been obtained.

A.2.2 Transceiver

To communicate wirelessly with its counterparts, a node is equipped with a RF transceiver. Since WSNs are typically composed of many devices [4], the cost of an individual sensor node must be kept to a minimum. Thus, simple and inexpensive transceivers are envisioned. In general, half-duplex transceivers are assumed which have three operational states: *transmit*, *receive* and *standby*. Typically, transmitting consumes as much power as receiving and standby lies beneath the power consumption of receiving by a factor 1,000 or more. In 2.4 GHz transceivers, we see a tendency that receiving is the most energy consuming mode. Table A.1 gives an overview of commercially available low-power transceivers.

The transceivers need to be "trained" to the incoming RF signal before an acceptable bit error rate (BER) can be established. This training is often done by transmitting a known sequence (i.e. a preamble) to the receiver, allowing it to adjust its input sensitivity and adopt its timing synchronization to that of the transmitter. Obviously, the preamble adds to the energy costs of transmitting and receiving a message.

Additionally, transceivers suffer from start-up effects. Transceivers that use crystals to derive their mixer frequencies, typically switch off their oscillators when they are put in low power state. To re-enable transmit or receive function, the crystal oscillator has to be restarted, which takes time and consumes a (usually undefined) amount of energy. Frequent transceiver state switches can drastically reduce the lifetime of the network.

Most of the commercially available transceivers operate in licence-free ISM bands, which are reserved for industrial, science or medical applications. In Europe, channels are defined around 868 MHz and stringent regulations exist [29]. For example, the maximum radiated power must be below a certain dBm's and devices are allowed to transmit only during a certain amount of seconds per hour. In most research in the area of WSNs, the last regulation is often neglected or at least not mentioned.

In literature the following bands are used for WSN applications: 433 MHz, 868 MHz, 915 MHz and 2.4 GHz. Maximum transmit powers range from 0 dBm to 13 dBm, receive sensitivity from -109 dBm to -93 dBm and bit rates from 40 kbaud to 1152 kbaud.

A.2.3 Power source

The power source of a wireless sensor determines how much energy can be spent during the lifetime of the node. We recognize two types of energy providers:

- **Batteries** — Energy for operation of the wireless node is extracted from energy stored in chemical compounds or radioactive materials. The latter are not widely used, due to the (potential) hazards of the radioactive material. However, they have very high energy density.

  Fuel and flow cells are more efficient than traditional chemical battery cells. However, these technologies need improvement before they are applicable. These types of batteries are mainly being developed for applications where quick recharging is needed. Fuel and flow cells differ from traditional batteries by their constant replenishment of reactants. In Section 2.3.1, we argued that replacing or refreshing of batteries is not likely in WSN applications, due to their large scale.

  Battery technologies are only progressing slowly: the energy density of batteries is doubling roughly every 9 years. Nowadays, the miniaturization of semiconductors causes major part of the increase in battery lifetime, not the battery itself. To give an impression of the current energy density of chemical batteries, widely used $LiSO_2$ batteries can hold roughly 3.6 kJ per cm^3.

Table A.1 — Commercially avaiable low-power transceivers (by April 2006)

Device		Freq. [MHz]	Modulation	Bitrate [kbits/s]	Tx power (max.) [dBm]	Rx sens. [dBm]	Tx current [mA]	Rx current [mA]	Standby current [μA]
Analog Devices	ADF7025	431-928	GFSK	384	13	-104	28	19	1
Atmel	ATA5428	431-872	FSK/ASK	40	10	-107	17	10	<1
Atmel	ATR2406	2400-2483	GFSK	1152	4	-93	42	57	1
Ember	EM2420	2400-2483	O-QPSK	250	0	-94	17	20	1
Infineon Techn.	TDA5250	868-870	FSK/ASK	64	13	-109	9	12	<1
Maxim	MAX2460	800-1000			2		26	23	0.5
Melexis	TH7122	27-930	FSK/ASK	115	10	-105	24	11	0.1
Nordic Semi.	nRF905	430-928	GFSK	100	10	-100	30	12.5	2.5
Nordic Semi.	nRF2401	2400-2524	GFSK	1000	0	-93	13	18	0.4
RF Monolithics	TRC101	300-1000	FSK	256	5	-100	22	9.5	0.25
Texas Instruments	CC1010	300-1000	FSK/ASK	76.8	4	-106	17	12	1
Texas Instruments	CC1110	300-1000	FSK/ASK	500	10	-109	31	22	0.6
Texas Instruments	CC2430	2400-2483	O-QPSK	250	0	-94	25	27	0.6

- **Environment** — The wireless sensor node extracts energy from its *environment*. Heat, light or vibrations are converted to electrical currents, which power the node and optionally charge backup batteries. Energy scavenging is the most preferred solution to the energy problem. However, the efficiency of most methods is still doubtful. Energy scavenging requires much more attention from the research community.

The actual choice of a power source is —again– application specific, e.g. in certain environments energy might be plenty available whereas in others energy is truly scarce.

One of the design goals of wireless sensors is to be energy-efficient. Consequently, the average current consumption of the nodes is minimised. Yet during high activity of the nodes, e.g. during transmitting or processing, the current consumption can be high as we demonstrate with measurements (Appendix B). The power source must be able to deliver these peak currents, even under harsh environmental conditions, such as low temperatures.

A problem —which has not been solved yet– is how to retrieve/recycle wireless sensors which ran out of energy. These devices cannot be located easily. Since wireless sensor networks potentially consist of many devices and are preferably unobtrusive, it is a difficult task to ensure that all devices are properly disposed off without polluting the environment.

A.2.4 Processor and memory

In the WSN research community, it is generally assumed that calculation intensive functions, like pre-processing of sensor readings, in-network processing, data aggregation, de-/encryption and other networking tasks, are carried out by processing unit(s). Although many of these functions can efficiently be implemented in ASICs, the flexibility and ability to be reprogrammed make general processing architectures an attractive choice. Obviously, memory is required in the wireless sensor node to hold the program of the processor. Nowadays, many low-power microcontrollers include FLASH memory, which can be rewritten many times, yet has excellent retention properties. Often, memory is assumed to be present on the wireless sensor node to hold (temporary) variables. Wireless sensor nodes require memory to reside message queues for networking.

A.2.5 Debugging and testing interface

Most microcontrollers have debugging provisions (for example via JTAG interface) that allow instruction-by-instruction stepping through program code and even reading of memory. However, this covers not all debugging and testing that is necessary. For example, time critical routines cannot be tested in this way and additional debugging means are required.

A.3 Prototype wireless sensor node designs

Table A.2 lists some well-known wireless sensor platforms on aspects related to used microcontroller and transceiver. Beutel argues that it is difficult to compare wireless sensor platforms due to the broad range of WSN applications and their different characteristics [11]. In Chapter 2, we have argued that WSNs are application specific networks. Consequently, this also holds for the hardware designs of wireless sensors.

Mostly, 8-bit or 16-bit RISC cores are used in the low-power microcontrollers. An exception is the uPart (`http://www.particle-computer.net/`), which uses an integrated 8051 core and ChipCon transceiver.

In general, the presented node designs use microcontrollers with on-chip RAM and program memory. The non-volatile program memory ranges between 32 kB and 128 kB, while the volatile memory is considerably smaller, ranging from 2 kB to 10 kB. Usually, the nodes can access external non-volatile memory to store arbitrary data.

Transceivers of RF Monolitics and ChipCon are popular choices, however, in newer platforms we see a tendency to ChipCon's 2.4 GHz transceiver. The burgeoning of low-cost and low-power wireless applications has caused rapid advancements in transceiver technology. The IEEE 802.15.4 physical layer standard begins to be widely accepted and this is reflected in nowadays transceivers. Most of them now operate in the 2.4 GHz band and are compatible with the standard, as is the ChipCon CC2420.

Before we present designed prototypes, we discuss the lessons learned from other prototypes.

A.3.1 Lessons from other prototype platforms

The important lessons we have learned from previous prototypes are related to *timing*:

- **Time critical decoding** — In the EU project EYES (IST-2001-34734) project we gained experience with the RFM TR1001 transceiver [89]. This transceiver was used on the EYES sensor node [39] designed by Nedap N.V. (top entry in Table A.2). It has excellent range, but it requires the microcontroller to be very precise in timing to generate and decode transceiver bit streams. We experimented with using the hardware UART in the MSP430 microcontroller for this task, but this often resulted in unreliable wireless communication. Similar problems —however with the MICA platform– are discussed by Hill et al. in [38]. Preferably, this (computation intensive) task should not be assigned to the MCU, but instead it should be carried out by dedicated hardware.

- **Timing during MCU low-power mode** — In the early stages of development of the μNode, we considered the use of an integrated microcontroller and RF frontend: the ChipCon CC1010. We envisioned a clear division between networking protocols, which could reside in the ChipCon, and the wireless sensor application that could independently run in a separate processor. The ChipCon was, at that stage, the first transceiver with integrated microcontroller.

  In early prototypes of a such platform, it turned out that the 8051 core in the ChipCon is indeed power-hungry, with its 13 mA's running at 16 MHz, as the datasheets indicated. The datasheets indicate that the microcontroller can be driven by a secondary 32 kHz crystal in low-power mode. However, once the microcontroller is put in low-power mode (and the 16 MHz crystal is stopped), it can only be awakened using inaccurate timing (best resolution 1/8s).Thus,

Table A.2 — Comparison of popular wireless sensor platforms

Name	Microcontroller			Transceiver		
	Type	RAM	Program memory	Type	Band	Data rate
Nedap EYES node	TI MSP430F149	2 kB	60 kB	RFM TR1001	868.4 MHz	115.2 kbps
Infineon EYES node	TI MSP430F149	2 kB	60 kB	Infineon TDA5250	868.4 MHz	64 kbps
See http://eyes.eu.org/						
Moteiv Telos Rev. A	TI MSP430F149	2 kB	60 kB	ChipCon CC2420	2400 MHz	250 kbps
Moteiv Tmote Sky	TI MSP430F1611	10 kB	48 kB	ChipCon CC2420	2400 MHz	250 kbps
See http://www.moteiv.com/						
ETH BTNode Rev. 3	Atmel ATmega128L	4 kB	128 kB	ChipCon 1000	868-915 MHz	38.4 kbps
				Zeevo Bluetooth	2400 MHz	600kbps
See http://www.btnode.ethz.ch/						
Crossbow MICA	Atmel ATmega128L	4 kB	128 kB	RFM TR1000	915 MHz	40 kbps
Crossbow MICA2	Atmel ATmega128L	4 kB	128 kB	ChipCon CC1000	868-915 MHz	38.4 kbps
Crossbow MICADOT	Atmel ATmega128L	4 kB	128 kB	ChipCon CC1000	868-915 MHz	38.4 kbps
Crossbow MICAz	Atmel ATmega128L	4 kB	128 kB	ChipCon CC2420	2400 MHz	250 kbps
See http://www.xbow.com/						
pPart Particle	Motorola PIC18F6720	4 kB	128 kB	RFM TR1001	868 MHz	125 kbps
uPart Particle	8051 core	4 kB	32 kB	ChipCon CC1010	868 MHz	19.2 kbps
See http://www.particle-computer.net/						

for accurate timing, the energy-consuming CPU must be kept running, draining batteries within days.

- **Quiescent current** — It is interesting to point out that quiescent current of the components in wireless sensor nodes can be a major factor in the lifetime of the device and its battery. Consider a component, which is used actively for 0.1% of the time consuming 0.9 mA and 99.9% 1.2 μA. Clearly, its average current consumption is 2.1 μA. However, an equivalent component, consuming 1.0 mA when active and 1.0 μA when idle, should be favoured since its average consumption is 2.0 μA. Thus, the component with the lowest active current is not necessarily the best choice for lifetime considerations. Naturally, different duty cycles impose different trade-offs.

A.3.2 μNode

The following components are selected for the μNode design: (1) Texas Instruments MSP430F1611 low-power mixed signal processor, (2) Nordic Semiconductor nRF905 433-915 MHz transceiver, (3) STMicroelectronics M25P40 low-voltage 4 Mbit serial flash memory and (4) MAXIM 3319 RS232 level converter with automatic shutdown. These components have been selected for their low-power characteristics and their short lead times and availability.

For the placement of components, the recommendations of the manufacturers are followed as closely as possible. The transceiver is decoupled with capacitors on all power lines and —as can be seen on the printed circuit board (PCB) design in Figure A.2– the footprint of the transceiver is surrounded by a ground plane. The RF input and output are connected to an impedance-matched circuit and feed a 50Ω whip antenna.

The digital control signals of the transceiver are connected directly to the processor. We chose to connect the signals of the transceiver that indicate (1) carrier detect, (2) address match and (3) data ready, to input pins of processor that can generate interrupts and are linked to timing functionality i.e. when one of the signals becomes active, timing hardware in the processor records —without interference of the CPU– the exact moment when the event occurred. This makes synchronization to incoming radio packets less time critical and gets rid of latency that interrupts generally induce. The SPI interface of the transceiver is connected to I/O pins, which make use of the hardware SPI of the MSP430. Thus, data to and from the transceiver can be efficiently exchanged, even using DMA functionality of the processor.

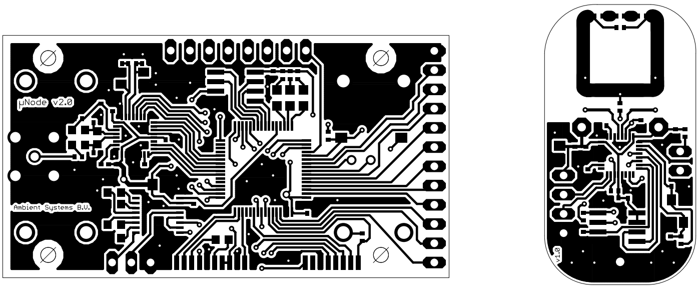
For debugging and programming of the node, we added (1) a JTAG interface, (2) three LEDs (red, green and blue), (3) a push button connected to an interrupt capable I/O pin and (4) a connector for a low-power graphical LCD (102 by 80 pixels). The node can communicate with a host PC using an RS232 connection. For that purpose a three-pin (RX, TX and ground) connector is added to the design.

The μNode by itself is not equipped with sensors as such. The wireless node can be extended with sensors or actuators using the following processor I/O that is available through connectors:

- **General purpose inputs/outputs** — Each of the lines can be used either as input to the processor, as output or can have alternate functionality. Each pin can be configured individuality and independently. When used as (digital) input, most lines are interrupt-capable. Outputs can sink and source currents up to a few mA's. This makes them suitable to provide switched power for sensors (Section A.2.1) and actuators.

- **ADC inputs** — Eight of the pins can be configured in software to be connected to the internal 12-bit ADC of the MSP430 processor. The microcontroller has a built-in voltage reference. However an external reference can also be connected.
- **Digital to analogue converter (DAC)** — The processor has a built-in DAC. Two lines can be configured to provide DAC output functionality.
- **I^{2}C** — Many digital sensors communicate with a host microcontroller through the I^{2}C protocol. The μNode design has (hardware) I^{2}C available on a connector to the exterior.

The μNode is powered by two 1.5V AA alkaline batteries, which can be attached to battery clips at the bottom side of the PCB. However, all components in the design can be fed with a supply voltage in the range of 2.7 to 3.6 Volts. Current consumption details are presented in Section B. The dimensions of the design are determined by the size of the batteries, resulting in a 32 by 60mm PCB area (Figure A.2(1)).



(1) (2)

Figure A.2 — Printed circuit board (PCB) design of (1) μNode, and (2) SmartTag

A.3.3 SmartTag

The SmartTag design is a simplified version of the μNode. It is based on a similar RF frontend, however the Nordic Semiconductor nRF9E5 transceiver includes a low-power 8051 CPU core. This 8051 CPU core consumes 2.6 mA while running at 16 MHz. When in deep-power down mode it consumes 2.5 μA and it can be awakened by its low-frequency clock. This integrated microcontroller and transceiver is therefore an attractive choice.

Furthermore, the SmartTag has considerably less memory than the μNode. The SmartTag has been designed for different goals. Although it can be connected to one digital sensor, or to a host PC using UART communication, its main purpose is to tag objects or persons (e.g. the chemical drums as presented in Section 2.2.2.iii). Its small form factor allows embedding in key chains or other personal items. The design has been used in several research projects, mainly as an experimental platform for localization applications (Section 2.3.4).

The SmartTag design has a loop antenna integrated on the PCB. Again, we followed application notes of the transceiver manufacturer closely regarding the design of the antenna. The platform is capable of driving one LED and reacting on a push button. This platform is powered by a coin cell. The antenna and coin cell determine the size of the SmartTag: 20 by 36 mm, Figure A.2(2).

When the CPU core —integrated with the transceiver– is powered, it enters its bootloader program. It copies the content of the EEPROM to its 4 kB RAM memory and executes the program code. Thus, to the user 4 kB minus the size of the program is left to store variables etc.

Appendix B

Current consumption of prototype wireless sensor nodes

In many WSN applications, node lifetime is an important issue. The large scale of those applications makes it virtually impossible to replace or refresh power sources on the individual nodes, as we argued in Section A.2.3. Obviously, the lifetime is dependant on the capacity of the power source, and on the consumption of the device. The latter is the focus of this appendix. First, DC measurements are presented and then the current consumption during transceiver activity. We conclude that the RF transceiver is the most energy consuming component of the wireless sensor node (excluding sensors).

B.1 DC current consumption

We compare the current consumption of the μNode with the expected current consumption in datasheets of the components. In Table B.1, *typical* supply currents are summarized from the datasheets of the components used in the μNode design. Note that these are given for a supply voltage of 3.0 Volts.

Since the μNode is an integrated design, the current consumption of individual components cannot be measured separately. Therefore, we create a set of programs that configure components to certain power modes, and we measure current consumption of the resulting configuration. The following modes are used:

- **LPMx** — Low power modes of the *microcontroller*, where $x = 0 \ldots 4$ denotes the mode. Relevant modes are LPM1, LPM3 and LPM4. In LPM1 the CPU is

Table B.1 — Typical supply currents for the μNode as defined in datasheets of the components. The datasheet of the EEPROM does not provide typical supply current, hence *maximum* supply current is given

Component	Mode	Supply current	
Transceiver	Power down	2.5	μA
	Standby	135	μA
	Receive	12.5	mA
	Transmit (-10 dBm)	9.0	mA
	Transmit (10 dBm)	30.0	mA
Microcontroller	RAM retention	0.2	μA
	Low power mode 4	0.2	μA
	Low power mode 3	2.6	μA
	Low power mode 1	75	μA
	Active (1 MHz)	500	μA
EEPROM	Power down	10	μA
	Standby	50	μA
	Read	6	mA
	Write	15	mA
RS 232	Power down	1.0	μA
	Active	0.3	mA

disabled, however all clock sources in the processor remain active (i.e. the processor remains ready for incoming serial communication and other peripherals in the processor can be active). In LPM3, the high speed (internal) oscillator is switched off, saving more power than in LPM1. When an interrupt occurs, the high speed oscillator is switched back on in $6\mu s$. In the LPM3 mode, the $32.768kHz$ crystal oscillator remains active and is source for all timing when the CPU is idle. In LPM4 all oscillators are switched off and the processor can only be woken through an external event on one of its input lines.

For other components, we use

- **PD** — Power down mode. This denotes the most energy saving state of a device.
- **S** — Standby mode. The device is ready to accept commands of the microcontroller. The RS 232 level converter MAX 3319 has a built-in automatic shutdown function. When no serial communication is detected, it switches off after 30 seconds.

Current consumption results are listed in Table B.2. In general, the current consumptions are slightly lower than we would expect from their datasheets, which are summarized in Table B.1. The MSP430 microcontroller consumes 2.1 mA when active at 4.6 MHz. The standby EEPROM consumes less than the maximum supply current given in its datasheet. The RS 232 level converter adds to the current consumption as expected. The μNode design consumes 10.9 μA when all components are in power down mode. Interestingly, the LEDs on the design drain a considerable current compared to the other components. Therefore, they should only be used when debugging is required.

Table B.2 — Measured supply currents for various settings of the μNode. Measured at 3.24 V supply voltage

Mode of component					Measured	Expected
Processor	EEPROM	Transceiver	RS 232	LEDs	supply current	supply current
LPM4	PD	PD	PD	Off	2.1 μA	13.7 μA
LPM3	PD	PD	PD	Off	10.9 μA	16.1 μA
LPM1	PD	PD	PD	Off	238 μA	μA
Active ($4.6 MHz$)	PD	PD	PD	Off	2.1 mA	2.3 mA
LPM3	S	PD	PD	Off	20.1 μA	56.1 μA
LPM3	PD	S	PD	Off	143.5 μA	148.6 μA
LPM3	PD	PD	S ($<$ 30s)	Off	273.0 μA	315.1 μA
LPM3	PD	PD	S ($>$ 30s)	Off	10.9 μA	16.1 μA
LPM3	S	S	S ($>$ 30s)	Off	152.5 μA	188.6 μA
LPM3	PD	PD	PD	Red	4.02 mA	–
LPM3	PD	PD	PD	Green	1.29 mA	–
LPM3	PD	PD	PD	Blue	1.36 mA	–

B.2 Transceiver current consumption

In this section, the current consumption of the transceiver is determined. For this purpose, we programmed the μNode with a simple periodic task using its real-time system software [25]. The task's single purpose is to let the node transmit a 32 byte payload packet.

Figure B.1 shows the measured current consumption of the μNode when transmitting three 32-byte packets at -10 dBm and 10 dBm, respectively. The current consumption during transmit (11.6 mA) is more than expected from the Nordic Semiconductor nRF905 transceiver datasheet (i.e. typically 9 mA, [81]). This is mainly due to the fact that we implemented the transmit function as a blocking function (i.e. the processor polls the data ready flag of the transceiver, before it continues). This incurs an additional current of 2.1 mA (Table B.2).

We conclude that the measured current consumption matches the expected consumption closely in this case. The current consumption of the μNode —when transmitting at 10 dBm– is slightly lower than expected (Figures B.1(2)). We measured 28 mA including 2.1 mA polling current of the processor, however, the datasheet indicates (typically) 30 mA.

Figure B.1 shows some artefacts at the beginning of the transmissions. The exact moment when the transceiver is put in transmit mode is annotated as *TX mode* in the figures. We recognize three phases:

- **Scheduler overhead and SPI programming** — At the beginning of this phase, the DCOS scheduler becomes active and our periodic task is released for execution. Before releasing the task, the scheduler takes a (measured) overhead of 120 μs.

  Once our task is released for execution, it configures and programs the transceiver using its SPI interface. The transceiver requires the microcontroller to write the size of the payload, which takes two bytes. Additionally, one command byte and the 32 payload bytes are written. Thus in total 35 bytes (i.e. 280 bits) are programmed into the transceiver, measured to take 648 μs.

  The scheduler overhead and SPI programming interval, which comes down to roughly 0.75 ms, can clearly be distinguished in Figure B.1. The current consumption during this phase is determined by the microcontroller (i.e. 2.1 mA). The transceiver, which is in standby mode during this phase, consumes no additional current while being programmed [81].

- **Transceiver switching** — Immediately after programming the 32 byte payload, the transceiver is put into transmitting mode (*TX mode* in the figures). The datasheet specifies a turn-on time of 650 μs, but during the switching, the current consumption is not defined. From Figures B.1(1) and (2), we conclude that the consumption during this interval is 6.7 mA, after subtracting the polling current of the microcontroller.

  Measurements show that the state switch time of the transceiver takes 390 μs instead of the expected 650 μs from transceiver datasheet [81].

- **Transmitting** — In the last phase, the message is broadcasted. In the transceiver datasheet [81] the on-air time is defined as follows

$$T_{on\ air} = T_{Start\,up} + T_{Preamble} + \frac{N_{Address} + N_{Data\,payload} + N_{CRC}}{R} \quad \text{(B.1)}$$

The parameters used in these experiments are summarized in Table B.3. On-air time evaluates according the datasheet to 6.93 ms. This is confirmed by our measurements.

Table B.3 — Transceiver parameters and settings

Parameter/Setting	Description	Value
$T_{Start\,up}$	Start up time from standby to transmit/receive	650 μs
$T_{Preamble}$	Duration of automatically added preamble	200 μs
$N_{Address}$	Length of transmitted address	32 bits
$N_{Data\,payload}$	Length of payload	256 bits
N_{CRC}	Length of automatically added CRC	16 bits
R	Bit rate (after Manchester encoding)	50 kbps

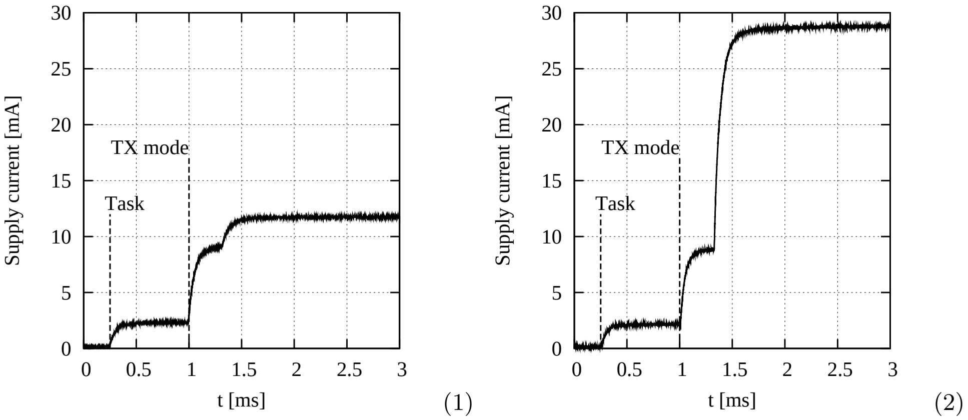


Figure B.1 — Current consumption when transmitting at (1) −10 dBm and (2) 10 dBm. Shown here are averaged results of ten runs

Next, the current consumption during receive mode is investigated. The Nordic datasheet specifies 12.5 mA in this case and our measurements reveal 12.3 mA (Figure B.2). Note that the current consumption is not affected by the actual receiving of a packet or idle listening. We conclude that the current consumption of the transceiver matches the datasheet [81] closely. Interestingly, Figure B.2 shows current drops, e.g. at $t = 5$ ms. Apparently, the receiver is switched off when no correct address match has been found [81].

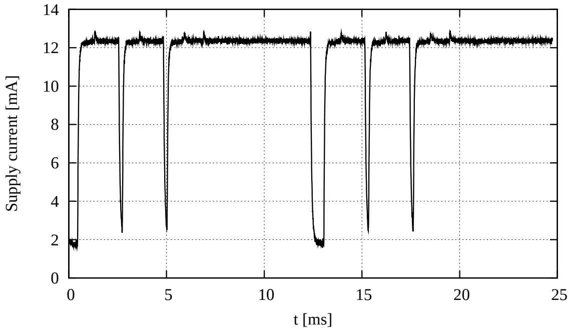


Figure B.2 — Current consumption of μNode when receiving 32 byte packets. Note that the transceiver remains in receive mode

Consider the energy cost of transmitting one single byte by the Nordic nRF905 transceiver, i.e. 9.5 mA during (rounded) 0.25 ms at 3 V equals 7.1 μJ/byte. The energy consumption of the microcontroller (at 4.6 MHz) is 2.1 mA times 3 V equals 6.3 mW. Thus, the energy budget of transmitting one byte is equivalent to roughly $\frac{1}{880}$ s of microcontroller processing i.e. $2,600$ instructions (assuming two-cycle instructions on average). Whenever less than $2,600$ instructions can be used to reduce the size of a transmitted message, this is worth the effort. We conclude that an energy consumption trade off should be made between the processing effort of compressing and transmitting uncompressed messages.

B.3 Conclusion

In this appendix, we considered current consumption of the μNode, a representative wireless sensor platform. The design of the μNode prototype wireless sensor has been discussed in Appendix A. Sensors have not been considered in our current consumption measurements.

We conclude that measured current consumption is lower than would be expected from typical supply current information in component datasheets. The EEPROM datasheet provides only maximum supply current. In practice, the component consumes less than the indicated maximum. This explains the difference in measured supply current versus the expected current consumption.

The transceiver is the most energy consuming component in the wireless sensor node hardware. In receive and transmit modes, supply currents are as indicated in its datasheet. However, also during mode switches, the transceiver drains supply current. Measurements revealed that during state switching the current consumption is 6.7 mA for a duration of 390 μs. The state switching interval is considerably shorter than stated in the transceiver datasheet. Other timing aspects match the datasheet.

It is interesting to note that the energy costs of transmitting one byte are roughly equivalent to $2,600$ microcontroller instructions. We conclude that compressing data, e.g. by aggregation, is valuable when the total message size is reduced and the processing requirements are kept below the aforementioned number of instructions per byte in reduction.

Appendix C

Spatial medium reuse and collision detection

In Chapter 4, we have presented a schedule-based medium access concept, which depends on the following assumptions: (1) the wireless medium can be spatially reused, and (2) the cause of an erroneous packet can be determined. In this appendix, we study in particular the physical layer effects of collisions, i.e. concurrent transmissions in range of a receiver.

C.1 Preliminaries

To validate the correctness of received packets, nodes carry out a cyclic redundancy code (CRC) check. Before a packet is transmitted, it is augmented with a CRC, which is computed by the transmitting node. The receiving node(s) calculate the CRC of the received packet and compares the result with the redundancy code in the message. When the two values match, the packet is accepted, otherwise, the node(s) decide that the packet contains at least one bit error and discards it.

For the following reasons bit errors can occur during the receiving of a packet: (1) noise in the receiver, and (2) simultaneous or in time overlapping transmissions. For our schedule-based medium scheduling algorithm (Chapter 4), nodes must be able to distinguish between these two causes. Namely, nodes are required to notify when two or more nodes are transmitting in the same time slot to maintain a connected network. However, when a bit error occurs due to e.g. noise, the erroneous packet must not result in the notification of a collision.

Low-power transceivers, which are suitable for use in wireless sensor networks, typically have a maximum output power of 10 dBm and a receive sensitivity of -100 dBm (Section A.2.2). When the received signal power is higher than the defined received sensitivity, the RF link is reliable. For example, in the datasheet of the transceiver used in the μNode design (Appendix A) is the receive sensitivity -100 dBm for $< 0.1\%$ bit error rate (BER). Thus, when a node measures a higher channel power, the probability is smaller that a bit error is caused by noise.

In this appendix, we show that nodes can determine the cause of erroneous packets by measuring channel power, i.e. when the power in the channel is high, a packet error is most likely caused by a collision. However, a low received signal strength can be caused due to collision *and* noise. In practice, a sharp threshold is set in measured received signal strength to decide the cause of packet errors.

In Section C.2, we focus on channel power versus distance (excluding interference). We discuss two propagation models and compare these with channel power measurements. In Section C.3, we add interfering nodes to our experiments to mimic nodes transmitting during identical time slots. In Section C.3.4, we draw conclusions.

C.2 Channel power measurements

Channel power and receiver input sensitivity are important determinants in the transmission range of RF systems. In this section, we compare two attenuation models with channel power measurements.

C.2.1 Free space and earth propagation models

Figure C.1 shows a typical transmitter/receiver pair. The power of the received signal —after the antenna of the receiver– is given by the Friis equation [17]:

$$P_{rx}(r) = P_{tx}G_{tx}G_{rx}\left(\frac{\lambda}{4\pi r}\right)^2 \tag{C.1}$$

where r is the distance between transmitter and receiver, P_{tx} the output power of the transceiver, G_{tx}, G_{rx} the respective antenna gains of the transmitter and receiver and λ the wavelength of the transmitted signal. This equation only captures the effect of *distance* between transmitter and receiver and is generally know as the free-space model [17]. The model assumes that energy radiated by the transmitter is isotropic spread and hence the channel power decays with 20 dB per decade. Antenna's have a length of $\frac{\lambda}{4}$.

In an outdoor environment, the slightly conductive (moist) ground reflects the radio waves partially (Figure C.2). At some distances there is more energy arriving at the receiver, but deep fades also exist when line-of-sight waves are extinguished by the (phase shifted) reflected part. These effects are captured in the earth propagation model [17]:

$$P_{rx}(r) = P_{tx}G_{tx}G_{rx}\left(\frac{\lambda}{4\pi r}\right)^2 \left|1+\Gamma_{||}e^{\imath\frac{kh_{tx}h_{rx}}{r}}\right|^2 \tag{C.2}$$

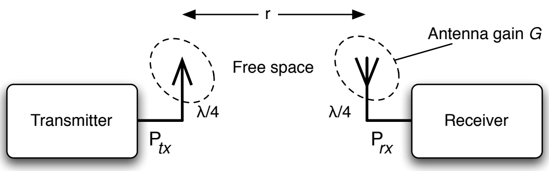


Figure C.1 — A typical transmitter/receiver pair

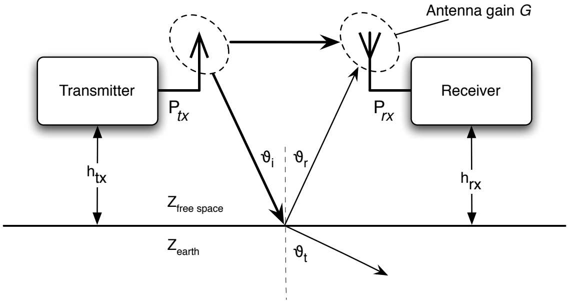


Figure C.2 — Reflecting ground wave in plane earth/two-ray propagation model (Equation C.2)

with wave number $k = \frac{2\pi}{\lambda}$. This model is valid when the earth's curvature can be neglected, which is the case in most WSN applications, where the distance r between nodes is small compared to the earths radius. Antenna heights are denoted by h_{tx}, h_{rx}. We assume monopole antennas perpendicular to the earth's surface, hence the polarisation of the RF waves is *parallel* to the surface [17]. $\Gamma_{||}$ represents the (parallel) reflection coefficient and is given by:

$$\Gamma_{||} = \frac{Z_{earth}\cos(\theta_t) - Z_{free\ space}\cos(\theta_i)}{Z_{earth}\cos(\theta_t) + Z_{free\ space}\cos(\theta_i)} \tag{C.3}$$

with wave impedances

$$Z = \sqrt{\frac{\imath\omega\mu_0\mu_r}{\sigma + \imath\omega\epsilon_0\epsilon_r}} \tag{C.4}$$

For free-space, the conductivity $\sigma = 0$ S/m, thus $Z_{free\ space} = \sqrt{\mu_0/\epsilon_0} \approx 377\ \Omega$.

When we assume that transmitter and receiver antennas are at equal height i.e. $h_{tx} = h_{rx} = h$, then angles θ_i, θ_r and θ_t are given by

$$\theta_i = \theta_r = \arctan\left(\frac{r}{2h}\right) \tag{C.5}$$

$$\theta_t = \arcsin\left(\frac{1}{\sqrt{\mu\epsilon}} \sin(\theta_i)\right) \tag{C.6}$$

Table C.1 lists relative dielectric constants and conductivities for relevant materials [17].

Table C.1 — Conductivities and relative dielectric constants of several materials [17]. Actual values depend on temperature and frequency. The listed constants are average, low-frequency values at room temperature

Surface	Conductivity $\sigma[S/m]$	Relative dielectric constant ϵ_r
Dry ground	0.001	4—7
Average ground	0.005	15
Wet ground	0.02	25—30
Sea water	5	81
Fresh water	0.01	81
Dry concrete	0.7	5
Gypsum board	0.15	2.8
Plywood	0.21	2.88
Brick wall	0.11	3.3
Glass	10^{-12}	4—10

C.2.2 Antenna gains and losses

Before a comparison between the two models and measurements can be made, antenna gains and losses need to be estimated.

In the antenna simulation program EZNEC (`http://www.eznec.com`), models of the transmitting and receiving antennas are created and gains are calculated to be 1.39 dBi and 2.15 dBi respectively for the μNode and the antenna connected to the spectrum analyzer [24].

The following power losses can be identified: (1) duty cycle loss $L_{duty\ cycle}$ and (2) cable loss L_{cable}. Because the transmitter —the μNode– is not transmitting continuously, but has a (measured) duty cycle of 84 %, the effective power level at the receiver is reduced by $L_{duty\ cycle} = 0.75$ dB. The analyzer averages over 3 seconds.

The antenna is not directly fit to the spectrum analyzer, but a coax cable connects the two. The cable has a length of 2 meters. This induces some loss of power $L_{cable} = 1.4$ dB [24].

We adopt our models accordingly:

$$P_{rx\ with\ losses} = P_{rx} - L_{duty\ cycle} - L_{cable} \quad \text{in dB's} \tag{C.7}$$

Other losses than the above are ignored.

C.2.3 Outdoor channel power measurements

In our experiments, we placed a μNode in an (moist) open field and let it transmit packets at given times (i.e. a duty cycle of 84 %). The transceiver is configured for an output power of -10 dBm. The channel power is measured at several distances using a spectrum analyzer (FSH-6, Rohde & Schwarz). Both antennas are placed at $h_{tx} = h_{rx} = 4\lambda \approx 1.38$ m above the surface. Our experiments are carried out on a (very) wet pasture, therefore, we choose $\epsilon_r = 30$ to account for the high water content in the ground and $\sigma = 0.02$ for the conductive salt content in the vegetation (Table C.1).

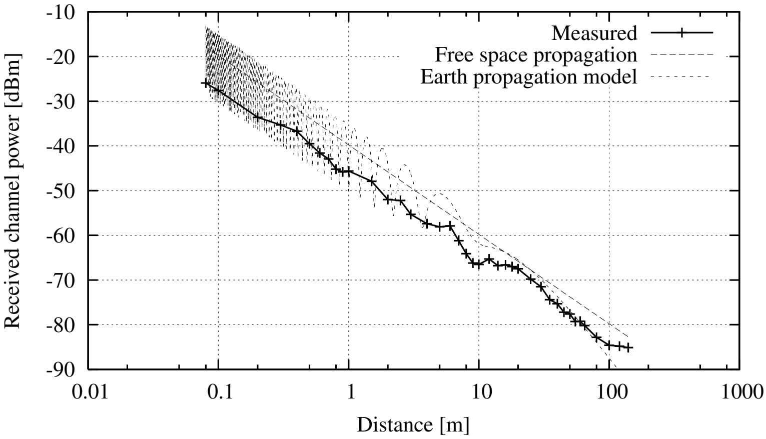


Figure C.3 — Measured channel power (outdoor scenario) compared to the free space model, Equation C.1, and the earth propagation model, Equation C.2. Compensation for losses during measurement is applied

Figure C.3 compares the measurements with the above discussed models. The measured channel power is within 8 dBm (below) the expected channel power of the free space propagation model. Interestingly, at a transmitter/receiver distance of $r = 9$ m, the measured channel power fades similarly as predicted by the earth propagation model.

C.2.4 Indoor channel power measurements

Well known effects, which have influence on the attenuation in indoor environments, are:

- **Reflections** — Radio waves are (partially) reflected by conducting material which is large compared to the wavelength. The material absorbs a part of the signal energy, so both reflected and transmitted signals are attenuated.
- **Scattering** — Scattering occurs when the radio waves incident on an object whose size is about or smaller than the signal wavelength. The energy of the

radio wave scatters in many directions. For example metal studs or cabinets in indoor environments case this phenomenon.

- **Diffraction** — A part of the radio wave bends into the shaded area behind large objects, even if the object is impenetrable to the radio wave. This effect is caused by the edges of walls, windows and other large objects.

In this section, we only consider reflection. We compare indoor channel power measurements with the two-ray model to account for reflection on the floor of the building and we neglect reflection of other objects —a departure from reality. Our experiments are carried out in a concrete building, therefore, we faithfully chose $\epsilon_r = 5$ and $\sigma = 0.7$ (Table C.1) in the two-ray earth propagation model.

Figure C.4 shows measured channel power in an indoor environment (similar transmitter/receiver configuration as in Section C.2.3). The receiver and transmitter (i.e. a μNode) are at all times in line of sight. The measurements indicate deep fades in the received channel power. Similarly, as in the outdoor case, the channel power fades around 9 m due to reflections.

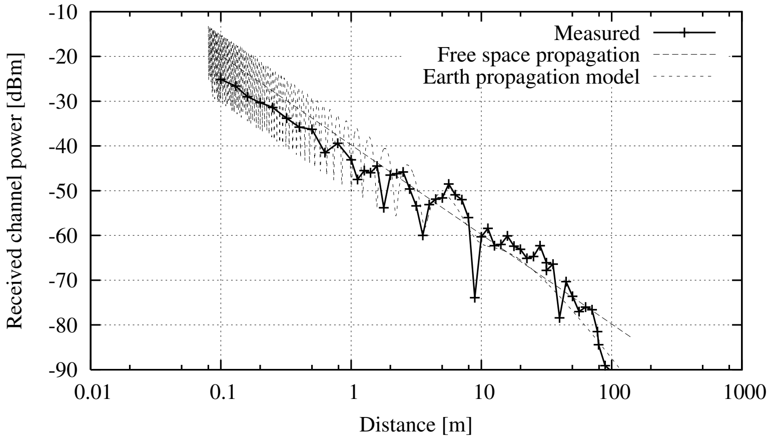


Figure C.4 — Measured channel power (indoor scenario) compared to the free space model, Equation C.1, and the earth propagation model, Equation C.2. Compensation for losses during measurement is applied

C.2.5 Conclusions

In this section, we have discussed two propagation models and compared them with measurements in outdoor and indoor environments. In general, the measured channel power is below the expected channel power of the propagation models, which are corrected for estimated losses. This is, for example, due to the variability of the output power of the transmitting μNode, which is defined to be within the range -6 to -14 dBm, instead of the assumed -10 dBm in our propagation models. Additionally, we ignored certain losses, e.g. feeder loss (circuitry that feeds the antenna), etc.

We conclude that channel power reduces with increasing distance between transmitter and receiver. Due to reflections, sudden fades or even peaks occur in channel power. During our experiments, we noted that the influence of reflections strongly depends on the height of the antennas, i.e. h_{tx} and h_{rx}.

C.3 Experiments with concurrent transmissions

To receive packets correctly, the signal strength of the desired signal must be higher than undesirable signals:

$$CCI < \frac{P_{TX}}{\sum P_{IX}} \tag{C.8}$$

with P_{TX} the signal strength (at receiver) of the desired signal, $\sum P_{IX}$ the summed signal strength's of undesirable signals and CCI, the co-channel interference accepted by the receiver. Typically, CCI depends on the receiver used (e.g. the modulation scheme). For example, the datasheet of the transceiver used in the μNode design defines $CCI = 13$ dB [81], meaning that as long as interfering signals in total remain 13 dB's below the desired signal, the nRF905 is able to demodulate the signals correctly.

When we use the free space propagation model i.e. Equation C.1, $CCI = 13$ dB translates into a factor 4.5 distance difference between a transmitter and an interferer node (assuming both transmit at similar output power). Obviously, the interferer node is furthest away from the receiving node. Consequently, the wireless medium can be spatially reused, as long as Equation C.8 holds for all receiving nodes.

C.3.1 Experiment setup

To validate Equation C.8, we carry out the following experiment. One μNode is assigned to be a desired transmitter, which is placed at a known distance from a receiving node. For each of the distances, we verified that the receiving node has perfect reception of the transmitting node.

Next, we add one interferer node, which transmits during the transmission of the wanted μNode. The position of the interferer is varied during the experiment and the receiving node tracks the number of correctly received packets. The experiment setup is illustrated in Figure C.5.

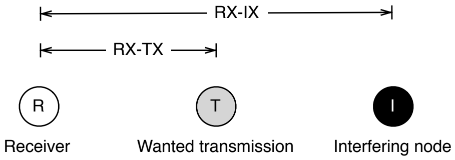


Figure C.5 — Experiment setup with one interfering node

C.3.2 Results

Measurements show that the effect of interference is dependant on the distance TX-RX, the height and height difference between transmitter, receiver and interferer. Table C.2 summarizes results of our experiments. In general, the required ratio between TX-RX and IX-RX distance is smaller than we expect from the free space propagation model (i.e. factor 4.5). Apparently, the transceiver is less sensitive to interference, as described in its datasheet.

Table C.2 — Summary of interference results. When the distance between receiver and interferer is larger than the indicated RX-IX distance, the receiver correctly receives all wanted transmissions of TX

Scenario	Height			Distance	Min. distance	Ratio
	TX	RX	IX	TX-RX	RX-IX	
	[m]	[m]	[m]	[m]	[m]	
Outdoor	1.03	1.03	1.03	10	30	3
Outdoor	1.38	1.38	1.38	10	50	5
Outdoor	1.38	1.38	1.72	10	> 50	> 5
Outdoor	1.38	1.38	1.38	175	250	1.5
Indoor	1.38	1.38	1.38	1	2.5	2.5
Indoor	1.38	1.38	1.38	5	7	1.4
Indoor	1.38	1.38	1.38	10	40	4
Indoor	1.38	1.38	1.38	20	60	3

Closing in on the maximum transmission range TX-RX of 175 m, the receiver becomes less sensitive to interference. Apparently, the interference disappears into the noise level of the receiver. Reflection has a clear influence on the experiments in this section. In Figure C.6, this is shown for two indoor cases.

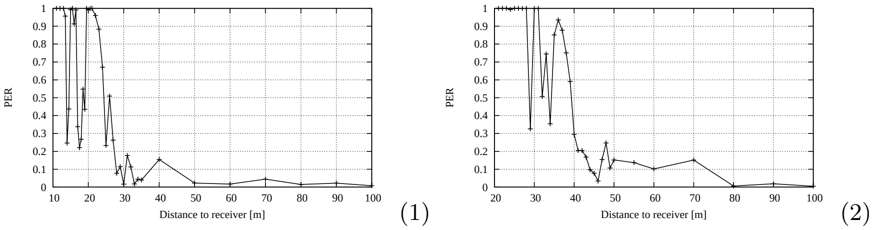


Figure C.6 — Effect of an interfering node on a transmitter/receiver pair spaced (1) 10m and (2) 20m (indoor environment). Fading of interference signal due to refection makes the desired transmissions prevail, while the interferer is close to receiver

Although, one of our objectives is to show that nodes can detect the cause of faulty packets by looking at the received signal strength, the nRF905 transceiver used in the μNode design does not provide such feedback. The lacking of this facility makes the transceiver less suited for the medium scheduling mechanisms as discussed in Chapter 4. In all tested cases, the channel power is well above the receive sensitivity of the receiver. This indicates that we expect a BER $<< 0.1\%$ and, therefore, that a packet error is more likely to be caused by a collision.

C.3.3 Related work: transport capacity in multi-hop network

In [36] and [3], Gupta and Agarwal present a model for successful communication, i.e. communication that is not disturbed by other transmissions. The authors use the notion of transmission/reception range r and interference range r_{int} to calculate the transport capacity of a wireless multi-hop network. The authors assume that the interference range r_{int} is typically larger than the transmission/reception range r. Communication is said to be successful if there is only one active transmitter in range of a receiver and there is no other active transmitter within interference range. Gupta et al. create virtual *exclusion* regions around transmitter/receiver pairs. When considering the surface of exclusion regions, the transport capacity of the network can be calculated. Gupta and Agarwal show the upper and lower bounds on transport capacity in [3, 36].

C.3.4 Conclusions

The wireless medium can be spatially reused when Equation C.8 is met. We have shown with our experiments that an interfering transmission loses its effect when its signal strength is a factor below that of the desired signal. Additionally, the cause of packet loss can be determined by measuring signal strength. A high channel power in combination with a lost packet is likely to be caused by a collision. In that case, the collision handle of our schedule-based medium access approach needs to be used to solve the conflict.

To our knowledge, there is only one medium access control protocol designed for wireless sensor network, which exploits the fact that transmissions can be interfered with: the bitMAC protocol [90]. The protocol assumes on-off keying as modulation scheme and consequently, interference acts as a logical OR between simultaneous transmitted bits.

Bibliography

[1] N. Abramson. The Aloha System– Another Alternative for Computer Communications. *Proc. 1970 Fall Joint Computer Conf.*, vol. 37:281–285, 1970.

[2] N. Abramson. Development of the AlohaNet. *IEEE Transactions on information theory*, vol. 31 no. 2:119–123, March 1985.

[3] A. Agarwal and P.R. Kumar. Improved capacity bounds for wireless networks. *Wireless Communication and Mobile Computing*, vol. 4:251–261, July 2004.

[4] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Elsevier Computer Networks*, 38(4):393-422, 2002.

[5] I. Akyildiz and I. Kasimoglu. Wireless sensor and actor networks: Research challenges. *Elsevier Ad Hoc Networks*, 2(4):351–367, Oct. 2004.

[6] Kennisinstituut Alterra. http://www.alterra.wur.nl.

[7] A. D. Amis, R. Prakash, D. Huynh and T. Vuong. Max-Min D-Cluster Formation in Wireless Ad Hoc Networks. *Proceedings of INFOCOM*, pages 32–41, 2000.

[8] S. Antifakos, F. Michahelles and B. Schiele. Proactive Instructions for Furniture Assembly. In *Proc. Ubicomp 2002*, Gothenburg, Sweden, 2002. `http://www.viktoria.se/fal/exhibitions/smart-its-s2003/furniture-video.mov`

[9] A. Barroso, U. Roedig, and C.J. Sreenan. μ-MAC: An Energy-Efficient Medium Access Control for Wireless Sensor Networks. *Proc. of the 2nd IEEE European Workshop on Wireless Sensor Networks (EWSN2005)*, IEEE Computer Society Press, 2005.

[10] S. Basagni. Finding a maximal weighted independent set in wireless networks. *Telecommunication Systems, Special Issue on Mobile Computing and Wireless Networks*, 18(1/3):155–168, 2001.

[11] J. Beutel. Metrics for Snesor Network Platforms. *In: REALWSN'06*, Uppsala, Sweden, 2006.

[12] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: A media access protocol for wireless LANs. *In Conf. on Communications Architectures, Protocols and Applications*, pages 212–225, Augustus 1994.

[13] A. Boulis, S. Ganeriwal and M.B. Srivastava. Aggregation in sensor networks: an energy-accuracy trade-off. *In Elsevier journal of Ad Hoc Networks*, Volume 1, Issues 2-3, pages 317–331, September 2003.

[14] E. Brinksma. Verification is experimentation! *In Int. Journal on Software Tools for Technology Transfer*, Volume 3(2), pages 107–111, May 2001.

[15] S. Chatterjea, L. van Hoesel, and P. Havinga. AI-LMAC: An Adaptive, Information-centric and Lightweight MAC Protocol for Wireless Sensor Networks. In *Proceedings of the Intelligent Sensors, Sensor Networks and Information Processing Conference*, pages 381–388, IEEE Computer Society Press, 2004.

[16] S. Chatterjea, L.F.W. van Hoesel and P.J.M. Havinga. A Framework for a Distributed and Adaptive Query Processing Engine for Wireless Sensor Networks. *Transactions of the Society of Instrument and Control Engineers*, Vol. E-S-1 (1), pp. 58–67, January 2006. ISSN 0453-4654

[17] D.K. Cheng. Field and Wave Electromagnetics. Second edition. Addison-Wesley publishing company, 1989, ISBN 0-201-52820-7.

[18] T. van Dam and K. Langendoen. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. *In 1st ACM Conf. on Embedded Networked Sensor Systems (SenSys 2003)*, pages 171–180, November 2003.

[19] S. De, C. Qiao, D. A. Pados, M. Chatterjee and S.J. Philip. An integrated cross-layer study of wireless CDMA sensor networks. *IEEE Journal on Selected Areas in Communications*, 22(7):1271–1285, September 2004.

[20] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38, 1977.

[21] P. Diaconis and F. Mosteller. Methods for Studying Coincidences. *Journal of the American Statistical Association*, Vol. 84, No. 408, pages 853–861, December 1989.

[22] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, Next century challenges: Scalable coordination in sensor networks. *In: Proc. 5th ACM/IEEE Ann. Int'l Conf. Mobile Computing and Networking (MobiCom'99)*, ACM Press, New York, pp. 263-270, August 1999.

[23] L. Evers, M.J.J. Bijl, M. Marin-Perianu, R.S. Marin-Perianu and P.J.M. Havinga. Wireless Sensor Networks and Beyond: A Case Study on Transport and Logistics. *Centre for Telematics and Information Technology, University of Twente, TR-CTIT-05-26*, June 2005, ISSN 1381-3625.

[24] J. Geerlings, L.F.W. van Hoesel, F.W. Hoeksema, P.J.M. Havinga and C.H. Slump. Spatial medium reuse in wireless sensor networks. *To be published in: SPSDARTS'07*, March 2007.

[25] T.J. Hofmeijer, S.O. Dulman, P.G. Jansen, and P.J.M. Havinga. DCOS, A Real Time, Leight-Weight Data Centric Operating System. In *IASTED Int. Conf. on Advances in Computer Science and Technology (ACST)*, pages 259–264, 2004.

[26] Decagon Devices Inc. Manual ECHO-TE soilmoisture sensor http://www.decagon.com/manuals/ECHOTEman.pdf, 2006

[27] A. El-Hoiydi. ALOHA with preamble sampling for sporadic traffic in ad hoc wireless sensor networks. *In: IEEE International Conference on Communications.* IEEE Press, pages 3418–3423, Apr. 2002.

[28] A. El-Hoiydi, J.-D. Decotignie, C. Enz, and E. L. Roux. Poster abstract: WiseMAC, an ultra low power MAC protocol for the WiseNET wireless sensor network. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 302–303, New York, NY, USA, 2003. ACM Press.

[29] European Telecommunications Standards Institute. ETSI EN 300 220-1.

[30] EYES project (EU IST 2001-34734). Project deliverables. http://eyes.eu.org

[31] A. Fehnker, L.F.W. van Hoesel and A.H. Mader. Modelling and Verification of the LMAC Protocol for Wireless Sensor Networks. *Technical Report TR-CTIT-07-09 Centre for Telematics and Information Technology*, University of Twente, Enschede, Februari 2007, ISSN 1381-3625.

[32] A. Fehnker, L.F.W. van Hoesel and A.H. Mader. Modelling and Verification of the LMAC Protocol for Wireless Sensor Networks. Accepted for publication in: *Integrated Formal Methods conference (IFM 2007)*, Oxford, UK, June 2007.

[33] Ji Fiala, Aleksei V. Fishkin, and Fedor V. Fomin. On distance constrained labeling of disk graphs. *Theoretical Computer Science*, 326(1-3):261–292, 2004.

[34] M. Gerla and T.J. Tsai. Multicluster, mobile, multimedia radio network. In: *ACM/Baltzer Journal of Wireless Networks*, Vol. 1(3), pages 255–265, 1995.

[35] R.P. Grimaldi. Discrete and Combinatorial Mathematics, Chapter 8: The Principle of Inclusion and Exclusion. Fourth edition. Addison–Wesley, 1998, ISBN 0-201-19912-2

[36] P. Gupta and P.R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, vol. IT-46 no. 2:388–404, March 2000.

[37] J. Hightower and G. Borriello. SPOTON: An indoor 3D location sensing technology based on RF signal strength. *Technical Report University of Washington*, February 2000.

[38] J. Hill and D. Culler. MICA: a wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, November 2002.

[39] L.F.W. van Hoesel, S.O. Dulman, P.J.M. Havinga, and H.J. Kip. Design of a low-power testbed for wireless sensor networks and verification. *CTIT Technical report*, September 2003. http://www.ctit.utwente.nl/library/techreports/tr03.doc/index.html

[40] L. van Hoesel and P. Havinga. A Lightweight Medium Access Protocol (LMAC) for Wireless Sensor Networks: Reducing Preamble Transmissions and Transceiver State Switches. In *In 1st International Workshop on Networked Sensing Systems (INSS 2004)*,pages 205–208, June 2004.

[41] L.F.W. van Hoesel, S. Chatterjea and P.J.M. Havinga. A Low-Latency, Information-Centric Medium Access Protocol for Wireless Sensor Networks. *In Proceedings of PRORISC.* The Netherlands, November 2004.

[42] L.F.W. van Hoesel, T. Nieberg, J. Wu, and P.J.M. Havinga. Prolonging the lifetime of wireless sensor networks by cross-layer interaction. *IEEE Wireless Communications*, 11(6)78–86, December 2004.

[43] L.F.W. van Hoesel and P.J.M. Havinga. Design Aspects of An Energy-Efficient, Lightweight Medium Access Control Protocol for Wireless Sensor Networks. *CTIT Technical report series, TR-CTIT-06-18*, June 2006. ISSN 1381-3625.

[44] L. van Hoesel, Y.W. Law, J. Doumen, P. Hartel and P. Havinga. Energy-Efficient Link-Layer Jamming Attacks against Wireless Sensor Network MAC Protocols. *CTIT Technical report series, TR-CTIT-06-18*, June 2006. ISSN 1381-3625.

[45] R.D. Hof. The quest for the next big thing. In: *BusinessWeek*, pages 91–94, August 25th 2003.

[46] J. Hill and D. Culler. Mica: a wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, Nov. 2002.

[47] J. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. *In 2nd ACM conf. on Embedded Networked Sensor Systems*, pages 81–94, November 2004.

[48] O. Durmaz Incel and S.O. Dulman and P.G. Jansen. Multi-channel Support for Dense Wireless Sensor Networking. *Proceedings of the First European Conference on Smart Sensing and Context, EuroSSC 2006*, The Netherlands, October 2006.

[49] IEEE standard 802.11. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. 1999.

[50] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. on Netw.*, 11(1):2–16, 2003.

[51] D. Johnson, Y. Hu and D. Maltz. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. *IETF Internet Draft*, http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-09.txt, April 2003.

[52] J.M. Kahn, R.H. Katz, and K.S.J. Pister. Next Century Challenges: Mobile Networking for "Smart Dust". *In Proc. 5th Ann. Intl. Conf. on Mobile Computing and Networking*, pages 271-278, U.S.A., August 1999

[53] J.N. Al-Karaki and A.E. Kamal. Routing Techniques in Wireless Sensor Networks: A Survey. *IEEE Wireless Communication Magazine*, 11(6)6–28, December 2004.

[54] C. Karlof, N. Sastry, and D. Wagner. TinySec: a link layer security architecture for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 162–175, New York, NY, USA, 2004. ACM Press.

[55] P. Karn. MACA - a new channel access method for packet radio. *In 9th ARRL Computing Networking Conference*, pages 134–140, September 1990.

[56] L. Kleinrock and F.A. Tobagi. Packet switching in radio channels: Part 1–carrier sense multiple-access modes and their throughput-delay characteristics. *IEEE Transactions on Communications*, Vol. 23, No. 12, December 1975.

[57] R. Knoppers. Sensor wijst naar vrije parkeerplek. *in Technisch Weekblad*, April 15th 2006.

[58] S. S. Kulkarni and U. Arumugam. TDMA Service for Sensor Networks. *Proceedings of the Third International Workshop on Assurance in Distributed Systems and Networks (ADSN)*, pages 604–609, Mar. 2004.

[59] K. Langendoen and G. Halkes. In *Embedded Systems Handbook*, Chapter 34: Energy-Efficient Medium Access Control. Editor: R. Zurawski, CRC Press, 2005, ISBN 0-8493-2824-1.

[60] K. Langendoen, A. Baggio and O. Visser. Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture. *Proceedings of 20th IEEE International Parallel and Distributed Processing Symposium*, page 174, ISBN 1-4244-0054-6, April 2006.

[61] Y. Law, P. Hartel, J. den Hartog, and P. Havinga. Link-layer jamming attacks on S-MAC. In *2nd European Workshop on Wireless Sensor Networks (EWSN 2005)*, pages 217–225. IEEE, 2005.

[62] Y.W. Law, L.F.W. van Hoesel, J.M. and Doumen, P.H. Hartel and P.J.M. Havinga. Energy-Efficient Link-Layer Jamming Attacks against Three Wireless Sensor Network MAC Protocols. *In: 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, Alexandria, Virginia. pp. 76-88. ACM Press. December 2005. ISBN 1-59593-227-5

[63] P. Levis, N. Patel, D. Culler, and S. Shenker Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks *In: Proceedings of the First Symposium on Networked Systems Design and Implementation* March 29–31, 2004.

[64] J. van Lint and R. Wilson. *A course in combinatorics.* Cambridge University Press, 2nd edition, 2001.

[65] G. Lu, B. Krishnamachari, and C.S. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering sensor networks. *4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks*, 2004.

[66] G. Lu, B. Krishnamachari, and C.S. Raghavendra. An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks. *18th International Parallel and Distributed Processing Symposium (IPDPS'04) - Workshop 12*, 13(13), 2004.

[67] S. Madden, M.J. Franklin, J.M. Hellerstein, and W.Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *5th Annual Symposium on Operating Systems Design and Implementation*, December 2002.

[68] S.R. Madden. *The Design and Evaluation of a Query Processing Architecture for Sensor Networks.* PhD thesis, University Of California, Berkeley, 2003.

[69] S. Mahlknecht and M. Bock. CSMA-MPS: a minimum preamble sampling MAC protocol for low power wireless sensor networks. *IEEE Workshop on Factory Communication Systems*, pp. 73–80, 2004, ISBN 0-7803-8734-1.

[70] M. Marin-Perianu, T.J. Hofmeijer and P.J.M. Havinga. Implementing Business Rules on Sensor Nodes. *In: 11th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2006)*, pages 292–299, September, 2006, ISBN 1-4244-0681-1.

[71] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. *Proceedings of the Second International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 39–49, November 2004.

[72] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97. ACM Press, 2002.

[73] M.J. Miller and N.H. Vaidya. A MAC Protocol to Reduce Sensor Network Energy Consumption Using a Wakeup Radio. *IEEE Transactions on Mobile Computing*, Vol. 4(3), pp. 228–242, 2005.

[74] T. Moscibroda and R. Wattenhofer. Coloring unstructured radio networks. *17th Symposium on Parallelism in Algorithms and Architectures*, July 2005.

[75] S. Muthukrishnan and G. Pandurangan. The bin-covering technique for thresholding random geometric graph properties. *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 989–998, February 2005.

[76] M. Mysore, M. Golan, E. Osterweil, D. Estrin, and M. Rahimi. TinyDiffusion in the Extensible Sensing System at the James Reserve. `http://www.cens.ucla.edu/~mmysore/Design/OPP`, University of California, Los Angeles, U.S.A., May 2003.

[77] T. Nadeem and A. Agrawala. Performance of IEEE 802.11 based Wireless Sensor Networks in Noisy Environments. *IEEE Workshop on Information Assurance in Wireless Sensor Networks (WSNIA'05)*, U.S.A., April 2005.

[78] D. Niculescu and B. Nath. Ad hoc positioning system (APS). *IEEE Global Telecommunications Conference (GLOBECOM '01)*, pp. (5)2926–2931, 2001.

[79] T. Nieberg. Independent and Dominating Sets in Wireless Communication Graphs. *Ph.D. thesis*, University of Twente, The Netherlands, ISBN 90-265-231-1, April 2006.

[80] T. Nieberg, S. Dulman, P. Havinga, L. van Hoesel and J. Wu. Collaborative Algorithms for Communication in Wireless Sensor Networks. *Ambient Intelligence: Impact on Embedded Systems*, Kluwer Academic Publishers, November 2003, ISBN 1-4020-7668-1.

[81] Nordic Semiconductor ASA. Single Chip 433/868/915 MHz Transceiver nRF905. Datasheet, V1.2, 2005.

[82] C. Perkins, E. Royer and S. Das. Ad Hoc On Demand Distance Vector (AODV) Routing. *IETF Internet Draft*, http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-13.txt, Februari 2003.

[83] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. *Proceedings of the Second International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 95–107, ACM Press, November 2004.

[84] J. Rabaey, E. Arens, C. Federspiel, A. Gadgil, D. Messerschmitt, W. Nazaroff, K. Pister, S. Oren and P. Varaiya. Smart energy distribution and consumption: Information technology as an enabling force. *White paper* CITRIS, 2001.

[85] V. Rajendran, K. Obrazka, and J. Garcia-Luna-Aceves. Energy-effcient, collision-free medium access control for wireless sensor netwoks. *Conference on Embedded Networked Sensor System*, pages 181–192, 2003.

[86] N. Reijers and K. Langendoen. Efficient Code Distribution in Wireless Sensor Networks. *In Proc. 2nd ACM Intl. Workshop on Wireless Sensor Networks and Applications (WSNA)*, San Diego, CA, September 2003.

[87] N. Reijers, G. Halkes, and K. Langendoen. Link layer measurements in sensor networks. In *1st IEEE Int. Conf. on Mobile Ad hoc and Sensor Systems (MASS '04)*, pages 224–234, IEEE Computer Society Press, 2004.

[88] RF Monolithics, Inc. *ASH Transceiver Software Designer's Guide*, 2002.

[89] RF Monolithics, Inc. TR1001: 868.35 MHz Hybrid Transceiver. Datasheet, 2002.

[90] M. Ringwald, K. Römer. BitMAC: A Deterministic, Collision-Free, and Robust MAC Protocol for Sensor Networks. *Proceedings of 2nd European Workshop on Wireless Sensor Networks (EWSN 2005)*, pages 57-69, Istanbul, Turkey, January 2005.

[91] U. Roedig, A. Barroso and C.J. Sreenan. f-MAC: A Deterministic Media Access Contrl Protocol Without Time Snchronization. *In Proc. of Third European Workshop on Wireless Sensor Network*, pages 276–291, 2006.

[92] K. Römer and F. Mattern. The Design Space Wireless Sensor Networks. *IEEE Wireless Communication Magazine*, 11(6)54–61, December 2004.

[93] J. de Rooij. Zelfhelend mesh-netwerk. *Computable magazine*, May 12th 2006.

[94] A.G. Ruzzelli, R. Tynan and G.M.P. O'Hare. An Energy-Efficient and Low-Latency Routing Protocol for Wireless Sensor Networks. *Proceedings of the 2005 Systems Communications (ICW'05)*, 2005.

[95] Y.E. Sagduyu and A. Ephremides. The Problem of Medium Access Control in Wireless Sensor Networks. *IEEE Wireless Communication Magazine*, 11(6)44–53, December 2004.

[96] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Optimizing sensor networks in the energy-latency-density design space. *IEEE transactions on Mobile Computing*, Vol. 1 No. 1:70–80, March 2002.

[97] E. Shi and A. Perrig. Designing Secure Sensor Networks. *IEEE Wireless Communication Magazine*, 11(6)38–43, December 2004.

[98] A. Sridharan and B. Krishnamachari. Max-min fair collision-free scheduling for wireless sensor networks. *Workshop on multi-hop wireless networks*, 2004.

[99] M. Srivastava, R. Muntz, and M. Potkonjak. Smart Kindergarten: Sensor-based Wireless Networks for Smart Developmental Problem-solving Environments (Challenge Paper). *In (ACM) Proc. 7th Ann. Intl. Conf. on Mobile Computing and Networking*, pages 132–138, Italy, July 2001.

[100] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. *Second international conference on embedded networked sensor systems (SENSYS)*, pages 214–226, November 2004.

[101] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a sensor network expedition. *First European workshop on wireless sensor networks (EWSN)*, pages 307–322, January 2004.

[102] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 181–192. ACM Press, 2003.

[103] S. Sakai, M. Togasaki, and K. Yamazaki. A note on greedy algorithms for maximum weighted independent set problem. *In Elsevier Science, Discrete Applied Mathematics*, 126:2-3, March 2003.

[104] J. Walrand. *Communication networks.* McGraw-Hill, second edition.

[105] A. Woo and D.E. Culler. A transmission control scheme for media access in sensor networks. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking*, Rome, Italy, 2001.

[106] A. Wood and J. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, Oct. 2002.

[107] J. Wu, S. Dulman, T. Nieberg and P. Havinga. EYES Source Routing Protocol for Wireless Sensor networks. *In proceedings of: European Workshop on Wireless Sensor Networks (EWSN'04)*, January 2004.

[108] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of 6th ACM Inter. Symp. on Mobile Ad Hoc Networking and Computing (ACM MOBIHOC'05)*, pages 46–57. ACM Press, 2005.

[109] W. Xu, T. Wood, W. Trappe, and Y. Zhang. Channel surfing and spatial retreats: defenses against wireless denial of service. In *WiSe '04: Proceedings of the 2004 ACM workshop on Wireless security*, pages 80–89, New York, NY, USA, 2004. ACM Press.

[110] X. Yang and N. Vaidya. A Wakeup Scheme for Sensor Networks: Achieving Balance between Energy Saving and End-to-end Delay. *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2004).* 2004

[111] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Vol. 3:1567–1576, June 2002.

[112] W. Ye, J. Heidemann, and D. Estrin. Medium Access Control with Coordinated, Adaptive Sleeping for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506, 2003.

[113] V.I. Zadorozhny, P.K. Chrysanthis and P. Krishnamurthy. Frame-work for Extending the Synergy between MAC Layer and Query Optimization in Sensor Networks. *In Proc. of First International Workshop on Data Management for Sensor Networks*, Canada, 2004.

[114] ZigBee and IEEE 802.15.4 standard.

[115] H. Zimmermann. OSI Reference Model – The ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, Vol. 28, No. 4, April 1980.

[116] M. Zorzi and R.R. Rao. Geographic random forwarding (GERAF) for ad hoc and sensor networks: Multihop performance. *IEEE Transactions on Mobile Computing*, (Vol. 2 No. 4):337–348, 2003.

[117] M. Zorzi and R.R. Rao. Geographic random forwarding (GERAF) for ad hoc and sensor networks: Energy and Latency Performance. *IEEE Transactions on Mobile Computing*, (Vol. 2 No. 4):337–348, 2003.

Journal papers/book chapters:

- A Framework for a Distributed and Adaptive Query Processing Engine for Wireless Sensor Networks. With S. Chatterjea and P.J.M. Havinga. In *Transactions of the Society of Instrument and Control Engineers*, Vol. E-S-1 (1), pp. 58–67, January 2006. ISSN 0453-4654
- Prolonging the lifetime of wireless sensor networks by cross-layer interaction. With T. Nieberg, J. Wu, and P.J.M. Havinga. In *IEEE Wireless Communications*, 11(6)78–86, December 2004, ISSN 1536-1284.
- Collaborative Algorithms for Communication in Wireless Sensor Networks. With T. Nieberg, S. Dulman, P. Havinga and J. Wu. In *Ambient Intelligence: Impact on Embedded Systems*, Kluwer Academic Publishers, November 2003, ISBN 1-4020-7668-1.

Conference papers:

- Modelling and Verification of the LMAC Protocol for Wireless Sensor Networks. With A. Fehnker and A.H. Mader. *To be published in: Integrated Formal Methods conference (IFM 2007)*, Oxford, UK, June 2007.
- Spatial medium reuse in wireless sensor networks. With J. Geerlings, F.W. Hoeksema, P.J.M. Havinga and C.H. Slump. *To be published in: SPSDARTS'07*, March 2007.
- Collision-free time slot reuse in multi-hop wireless sensor networks. With P.J.M. Havinga. In *Intelligent sensors, sensor networks and information processing (ISSNIP'05) conference*, Australia, 2005, ISBN 0-7803-940
- An Application-Tailored MAC Protocol for Wireless Sensor Networks. With S. Chatterjea and P.J.M. Havinga. In *Proceedings of the 2005 International Workshop on Wireless Ad-hoc Networks*, London, UK, 2005.
- Energy-Efficient Link-Layer Jamming Attacks against Three Wireless Sensor Network MAC Protocols. With Y.W. Law, J.M. Doumen, P.H. Hartel and P.J.M. Havinga. In *3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'05)*, pp. 76–88, 2005, ISBN 1-59593-227-5
- AI-LMAC: An Adaptive, Information-centric and Lightweight MAC Protocol for Wireless Sensor Networks. With S. Chatterjea and P. Havinga. In *Proceedings of the Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP) Conference*, pages 381–388, IEEE Computer Society Press, 2004, ISBN 0-7803-8894-1
- An Energy-efficient Medium Access Protocol for Wireless Sensor Networks. With P.J.M. Havinga. In *Proceedings of SenSys*, Baltimore, USA, 2004, ISBN 1-58113-879-2

- A Low-Latency, Information-Centric Medium Access Protocol for Wireless Sensor Networks. With S. Chatterjea and P.J.M. Havinga. *In Proceedings of PRORISC.* The Netherlands, November 2004.
- A Lightweight Medium Access Protocol (LMAC) for Wireless Sensor Networks: Reducing Preamble Transmissions and Transceiver State Switches. With P. Havinga. In *In 1st International Workshop on Networked Sensing Systems (INSS 2004)*,pages 205–208, June 2004.
- Advantages of a TDMA based, energy-efficient, self-organizing MAC protocol for WSNs. With T. Nieberg, H.J. Kip, P.J.M. Havinga. In *IEEE VTC spring*, Italy, May 2004.
- On the design of an energy-efficient low-latency integrated protocol for distributed mobile sensor networks. With A.G. Ruzelli, L. Evers, S.O. Dulman and P.J.M. Havinga. *Proceedings of the International Workshop on Wireless Ad hoc Networks*, Finland, 2004, ISBN 951-42-7373-7
- Design of an autonomous decentralized MAC protocol for wireless sensor networks. With L. Dal Pont and P.J.M. Havinga. In *Proceedings of the 6th International Symposium on Autonomous Decentralized Systems (2003)*, Italy, 2003, ISBN 0-7695-1876-1

Technical reports:

- Modelling and Verification of the LMAC Protocol for Wireless Sensor Networks. With A. Fehnker and A.H. Mader. *Technical Report TR-CTIT-07-09 Centre for Telematics and Information Technology*, University of Twente, Enschede, Februari 2007, ISSN 1381-3625.
- Design Aspects of An Energy-Efficient, Lightweight Medium Access Control Protocol for Wireless Sensor Networks. With P.J.M. Havinga. *CTIT Technical report series, TR-CTIT-06-18*, June 2006. ISSN 1381-3625.
- Energy-Efficient Link-Layer Jamming Attacks against Wireless Sensor Network MAC Protocols. With Y.W. Law, J. Doumen, P. Hartel and P. Havinga. *CTIT Technical report series, TR-CTIT-06-18*, June 2006, ISSN 1381-3625.
- Design of a low-power testbed for wireless sensor networks and verification. With S.O. Dulman, P.J.M. Havinga, and H.J. Kip. *CTIT Technical report*, September 2003, ISSN 1381-3625.

Patents:

- Distributed precision based localization algorithm for ad-hoc wireless networks. With P.J.M. Havinga, L. Evers and S. Octavian Dulman. EP 1 617 601 A2. April 18, 2005.