Safety-critical wireless sensor networks

Diss. ETH No. 18451

Safety-Critical Wireless Sensor Networks

A dissertation submitted to the

SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZURICH

for the degree of

Doctor of Sciences

presented by

Andreas Meier
Msc ETH in Electrical Engineering

born November 10, 1978
citizen of Luzern

accepted on the recommendation of

Prof. Dr. Lothar Thiele, examiner
Prof. Dr. Koen Langendoen, co-examiner

2009

Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

TIK-SCHRIFTENREIHE NR. 106

Andreas Meier

Safety-Critical Wireless Sensor Networks

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

A dissertation submitted to the
Swiss Federal Institute of Technology (ETH) Zürich
for the degree of Doctor of Sciences

Diss. ETH No. 18451

Prof. Dr. Lothar Thiele, examiner
Prof. Dr. Koen Langendoen, co-examiner

Examination date: June 12, 2009

Abstract

Safety-critical sensor networks are pervasively embedded in our surroundings. Such networks impose strong requirements in terms of reliability and latency of sensor readings and do for instance allow to monitor buildings for detecting fires and intrusions. The network requires a costly and cumbersome installation of wires for connecting the distributed sensors, which is sometimes not even possible. This suggests adopting the emerging technology of wireless sensor networks (WSN) to be used in a safety-critical context. With this technology, the wires connecting the sensors can be replaced by a radio and a battery pack.

A WSN is a collection of embedded sensor nodes with wireless networking capabilities. Collectively the sensor nodes establish a wireless ad-hoc network for transferring, processing and monitoring the sensed data. In order to ensure a small form factor, the sensor nodes are highly integrated and provide minor processing capabilities and limited memory. More stringent, the battery-powered nodes have to carefully orchestrate the power-hungry radio device if a yearlong independent operation is targeted. To make matters even worse, wireless communication is inherently unreliable and limited in range. Altogether this makes it a very demanding task to ensure a reliable, timely and energy efficient transport of the sensed data over possibly multiple hops.

Reliability is of utmost importance in a safety-critical environment. Additionally, there are often regulations imposing strong demands in terms of message latency and the availability of the sensor nodes. In particular, this thesis refers to the exemplary case of a wireless fire-alarm application, in which an alarm must be reported to a control station within 10 seconds, and a failed node has to be detected by the network within 5 minutes. These requirements are exacerbated by the fact that the nodes have to power off the radio more than 99% of the time, in order to enable an independent operation for several years with a small battery.

This thesis contributes towards adopting WSN technology for safety-critical applications. It focuses on communication aspects, and makes the following major contributions:

- The novel communication strategy, Dwarf, ensures a robust and timely forwarding of alarm messages, despite having the sensor nodes powered off most of the time. The maintenance protocol DiMo allows the monitoring of the nodes and the network topology

with minimal communication overhead. In conjunction, Dwarf and DiMo enable safety-critical networking.

- This thesis contributes an analytical framework for analyzing and comparing WSN MAC protocols. The framework provides deep insight into the behavior of WSN MAC protocols and provides the first available solution for benchmarking. This provides the means for selecting the most suitable MAC protocol for an application at hand.

- This thesis contributes the NoSE protocol enhancement. NoSE allows for considerable energy savings while maintaining the sensor network and allows for a swift and dependable initialization. NoSE is beneficial for specialized applications and protocols like Dwarf and DiMo that are optimized for yearlong operation, but exhibit a reduced energy efficiency and responsiveness during maintenance and initialization.

Zusammenfassung

Sicherheitskritische Sensornetzwerke sind allgegenwärtig und werden beispielsweise eingesetzt um Brände oder Einbrüche zu detektieren. Solche Netzwerke erfordern eine zuverlässige Übermittlung der Sensormessungen mit einer minimalen zeitlichen Verzögerung. Bisher erforderte das eine aufwändige und kostspielige Verdrahtung der verteilten Sensoren. Es liegt daher auf der Hand, die neu entstehende Technologie der drahtlosen Sensornetzwerke auch für sicherheitskritische Anwendungen zu nutzen. Mit dieser Technologie können die Verbindungskabel der Sensorenknoten durch ein Funkmodul und eine Batterie ersetzt werden.

In einem drahtlosen Sensornetzwerk wird im Verbund ein Netzwerk aufgebaut, damit die Sensordaten transferiert und verarbeitet werden können. Um die Grösse und den Energieverbrauch zu minimieren sind die Sensorknoten hochintegrierte Systeme und verfügen daher nur über eingeschränkte Rechenleistung und Speicher. Zusätzlich ist es zwingend notwendig das Funkmodul als Hauptenergieverbraucher nur zu gezielten Zeitpunkten einzuschalten, um einen jahrelangen und unabhängigen Betrieb zu ermöglichen. Erschwerend kommt die Unzuverlässigkeit und limitierte Reichweite der drahtlosen Kommunikation dazu. Es ist daher eine äusserst schwierige Aufgabe einen zuverlässigen und energieeffizienten Datentransport innerhalb der vorgegebenen Zeit zu gewährleisten.

Zuverlässigkeit ist für sicherheitskritische Anwendungen von höchster Bedeutung. Zudem gibt es Vorschriften die Mindestanforderungen an die Latenz und Verfügbarkeit des Netzwerkes stellen. Diese Arbeit bezieht sich konkret auf das Beispiel eines Feuermeldenetzwerks mit der Vorgabe, ein detektiertes Feuer innert 10 Sekunden und einen beschädigten Sensorknoten innerhalb von 5 Minuten bei einer Kontrollstation zu melden. Diese Anforderungen werden verschärft, da das Funkmodul mehr als 99% der Zeit ausgeschaltet sein muss, um einen jahrelangen Betrieb ohne Batteriewechsel sicherzustellen.

Diese Dissertation befasst sich mit offenen Problemen, die es für die Anwendung drahtloser Technologie auf dem Gebiet von sicherheitskritischen Sensornetzwerke zu lösen gilt. Die Arbeit fokussiert sich auf das Themengebiet der Kommunikation und leistet folgende Hauptbeiträge:

- Mit der Kommunikationsstrategie Dwarf kann ein robustes und rechtzeitiges zustellen von Alarmnachrichten sichergestellt werden, auch wenn die Sensorknoten überwiegend ausgeschaltet sind. Das Überwachungsprotokoll DiMo stellt die kontinuierliche Überwachung der Knoten und der Netzwerktopologie sicher. Im Zusammenspiel ermöglichen Dwarf und DiMo sicherheitskritische drahtlose Sensornetzwerke.

- Diese Arbeit stellt ein analytisches Instrument für die Analyse und den Vergleich von MAC Protokollen für drahtlose Sensornetzwerke vor. Es ermöglicht einen tiefen Einblick in das Verhalten der Protokolle und bietet die erste verfügbare Lösung für Benchmarks. Damit ist es möglich das passende MAC Protokoll für eine bestimmte Anwendung auszuwählen.

- Mit der NoSE Protokollerweiterung kann das Sensornetzwerk energieeffizient unterhalten und zudem schnell und zuverlässig in Betrieb genommen werden. NoSE ist von besonderem Vorteil für spezialisierte Applikationen und Protokolle wie Dwarf und DiMo, die für einen langjährigen Betrieb optimiert sind, aber während dem Unterhalt und der Inbetriebnahme eine reduzierte Energieeffizienz und Ansprechverhalten zeigen.

Acknowledgement

First and foremost I would like to thank Prof. Dr. Lothar Thiele for supporting my thesis and for convincing me to pursue a PhD in the first place. It has been a great experience I would not have wanted to miss. I really appreciated his trust he put in me for doing my research independently, yet always being available for discussions when necessary.

I would like to express my gratitude to Prof. Dr. Koen Langendoen for co-examining my thesis and for the fruitful collaboration. His sabbatical at ETH Zurich was perfectly timed for giving my research a jump start. It was also Koen who made me appreciate the value of academic research.

Many thanks to my fellow colleagues here at the Computer Engineering Group. Thanks to Matthias Woehrle for the fruitful discussions and his valuable input when revising many of my papers and texts. I would also like to thank Dr. Jan Beutel for all the valuable insights on WSNs he provided me with. It was great sharing an office with the both of you. Furthermore I would like to thank Mario Strasser from the Communication Systems Group for the valuable collaboration.

My gratitude is given to the whole team from Siemens Building Technologies for enabling the joint research on safety-critical WSNs. In particular I would like to thank Dr. Simon Künzli and Dr. Philipp Blum for all the inspiring discussions and the data from the system implementation.

My gratitude is also given to Prof. Dr. Mehul Motani and Prof. Dr. Samarjit Chakraborty from the National University of Singapore for enabling the five-month research visit in Singapore and for the rewarding collaboration.

Many thanks as well to all my master- and semester-thesis students, in particular to Roman Lim, Tobias Rein and Mischa Weise. Your results provided me with a lot of valuable insight in various important aspects for safety-critical WSNs.

Finally I would like to express my deepest gratitude to my dear family and friends. Thanks a lot for your long-lasting support.

The work presented in this thesis was supported by CTI grant number 8222.1 and the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322. This support is gratefully acknowledged.

Contents

1

Introduction

Safety-critical sensor networks are deeply embedded in our environment. Such networks allow airplanes to fly, nuclear power plants to produce energy and smoke detectors to raise alarms. These networks have in common that a so called control station is regulating the system by observing a network of attached sensors. The distributed sensors are connected to the control station by a wire that powers the sensors and allows for a reliable and immediate communication.

In a wireless sensor network (WSN) the wire connecting the sensor with the control station is replaced by a radio device and a set of batteries. Consequently the sensor becomes an autonomous unit (mote) that has to forward its data by means of radio communication. Without the necessity of the (costly) wiring, the WSN can be deeply embedded in our surroundings and for instance remotely gather environmental data such as temperature and humidity. If a wide sensor range needs to be covered, the motes build a multi-hop topology and collectively forward the data. The motes run on batteries and hence the power consumption needs to be minimized for a prolonged lifetime. With the radio device as the mote's major energy consumer, the radio has to be switched off (duty cycled) most of the time. This however greatly increases the delay of the sensor readings, in particular in a multi-hop environment.

Adopting WSN technology for implementing safety-critical applications such as fire- and burglar-alarm systems is a big challenge because of the real-time constraints imposed by their users. Typically, alarms detected by sensor nodes have to be reported reliably, and within a few seconds to at least one sink node, even in case that some of the nodes and communication links fail. Additionally, safety-critical applications are re-

quired to observe the status of the network and report node failures within a specified time. A complicating factor is that maintenance costs have to be very low to make an application economically feasible. This requires energy-efficient operation, since batteries should not be replaced more often than once every couple of years. With current-generation sensor node hardware built out of commercial off-the-shelf (COTS) components, the radio consumes the most power, and the above lifetime requirement translates into a radio duty cycle of well below 1%.

Mandatory requirements for a wireless fire-alarm system [Eur08]:

1. Report a detected fire within 10 s at the sink.
2. Report a failed node within 5 min at the sink.
3. A network lifetime of 3-5 years.

Section 1.1 provides a broad overview of today's WSNs technology, highlighting in Section 1.2 that the state-of-the-art in WSN is not suited for safety-critical operation. The thesis' main contributions are presented in Section 1.3, providing the fundamentals for enabling safety-critical WSNs. The thesis is outlined in Section 1.4.

1.1 WSN – State of the Art

Kahn et al.'s vision of *Smart Dust* [KKP99] is commonly seen as the origin of wireless sensor networks (WSNs). According to this vision, minuscule sensor motes, consisting of a small circuitry for data processing, a sensor and means for wireless communication are deployed in our surroundings. The motes would then build an ad-hoc network and monitor the environment and for instance report the sensed data regularly to a central station. Realizing this vision requires, among others, solving the most crucial factors of energy-efficient operation for longevity and the networking on a distributed, resource limited and deeply embedded system.

Pilot deployments of wireless sensor networks, such as the *Great Duck Island* [MPS+02] or *ZebraNet* [JOW+02], proofed the feasibility of such systems, even though their performance still left ample room for improvement. Especially the energy consumption did not scale well with the network size and the message delivery rate was far from being satisfying. For instance the Great Duck Island gathered 58% of the data, a deployment in a redwood tree [TPS+05] less than 50%, or a deployment in a potato field [LBV06] only resulted in a data recovery of as little as 2%.

Although the application domains vary, these deployments typically fall into the class of remote data gathering, where rather soft constraints on performance (e.g., latency, throughput, and lifetime) allow for straightforward engineering solutions. The experience gained with these pilots is being incorporated into second-generation software that is better tuned, more robust, and offers the potential for enlarging the scope to more demanding applications such as for permafrost measurements in exposed alpine conditions [BGH+09].

1.1.1 WSN Hardware

State of the art WSN hardware is still following the line of Smart Dust's original proposal, i.e., having means for communication, a sensing device, a data processing unit and a battery for power supply. Despite its superior energy efficiency, the original idea of having optical means for communication is generally discarded due to its very limiting line of sight requirement and is replaced with radio frequency (RF) technology. Nowadays, the motes are assembled with COTS components and consist of a low-power microprocessor (μC), a RF radio and antenna, a sensing unit and a power supply. However it is only a matter of time, before system-on-a-chip (SoC) designs, such as the WiseNET [EEHDP04] prototype platform, will emerge on the market allowing for an increased performance for a reduced price.

Various mote like platforms were developed in the last years, e.g., BTnode [BKM+04], Imote [KAH+04], Mica family [HC02, HSW+00], Smart-Its [BG03], Sun SPOT [Sun], TinyNode [DFFMM06], Tmote family [PSC05] etc. Even though being assembled of similar components, the difference in terms of computing power, available memory, and radio bandwidth can be quite substantial, which in turns allows selecting the platform that suits the requirements [Beu06].

Energy

Energy is commonly provided by a battery and is often the most limiting factor for sensor networks. This makes it very attractive to use *energy scavenging* technologies for harvesting environmental energy such as solar power, wind or vibration [RWR03, RSF+04]. Scavenging technology is still in its beginnings, yet is very promising for some WSN applications. Many deployments however cannot benefit from these technologies, as such environmental energy is not always available in a sufficient amount (e.g., a node placed in a dark closet), or the costs for equipping the node with a scavenging device are not affordable.

The sink is commonly less resource constraint than the source nodes. In particular the sink is often line powered or has a battery with a largely increased capacity attached. This can be exploited in protocol design, increasing the sink's energy drain for a reduced energy consumption of the (resource limited) nodes neighboring the sink.

Microcontroller

The sensor data is processed with a (microcontroller) μC, such as the ATmega128L or MSP430C111, integrating a data processing unit, memory, timers, analog-to-digital converters (ADC). For selecting a suitable μC, the energy consumption is most crucial. Besides a minimized power consumption while processing or being idle, it is essential that so called sleep states are supported. During sleep, most modules of the μC are powered off (set to sleep), which greatly reduces the energy consumption. Depending on the sleep state, the system can only be woken up by a timer or an external interrupt (e.g., clock).

The μCs provide limited processing power and memory in turn for low fabrication costs and the low-power operation. For instance the ATmega128L (MSP430) provides 8 MHz (8 MHz) clock speed, 64 kB (10 kB) of RAM and 128 kB (48 kB) of ROM. These resource constraints require that no processing and memory intensive calculations are performed.

Radio

The radio transceiver is by far the greatest energy consumer of the system, independent of whether the radio is sending, receiving or just idle. Hence minimizing the payload does not necessarily translate in a minimized energy consumption. Duty cycling the radio, that is, repeatedly switching it off for some time, is the only way to achieve the required two orders of magnitude reduction in energy consumption for extending lifetime from days to years. By putting the radio into (deep) sleep, the energy consumption reduces to zero (i.e., from mW to μW range), which is the task of the Medium Access Control (MAC) layer driving the radio hardware. Often neglected but essential are the radio's switching times, essentially increasing the energy drain by prolonging the radio's on time. The duty cycling effectively reduces the available bandwidth. This however is a minor concern due to the commonly low data rates in the WSN.

Most transceivers operate in the license free ISM bands, e.g., 433 MHz, 868 MHz and 2.4 GHz, and usually provide a communication range in the order of ten to hundreds of meters if the line of sight is not blocked. WSNs are usually spread over a larger area, requiring a multi-hop network architecture. Furthermore, the node's sensing range is commonly much

smaller than its communication range, resulting in dense deployments. Hence there are many possible neighbors available of which a substantial part provide a poor and unpredictable link quality [ZG03, RHL04, SKH06, SL06] and therefore should not be used for routing. This effect is especially severe in indoor settings yet most essential when doing safety-critical communication and is further detailed in Chapter 2.

Two kinds of radios are typically being used and are distinguished by their level of abstraction, in particular whether the access is stream (byte) or packet based. Stream-based radios, such as the TI CC1000 and the RFM TR1001 allow to control every single bit that is put on the channel allowing to greatly optimize the MAC's performance, however introducing substantial overhead for the μC. On the other hand, a packet based radio, like the TI CC2420 or TI CC2500, provides a packet interface for transceiving messages. Hence it is the radio that deals with the particulars of handling the single bits, which trades off flexibility for a greatly minimized overhead of the μC.

Sensors

There are two types of sensors: analog and digital ones. For converting analog sensor signals, the analog-to-digital converter (ADC) of the μC is commonly used. This conversion needs to be performed very carefully, as the ADC is very susceptible to noise due to the low-power voltage reference. In particular the node should turn the radio off and suspend processing if highly accurate measurements are required.

The sensing can account for a substantial part of the motes overall energy budget and price [BGH+09]. However, emerging technologies like micro-electro-mechanical systems (MEMS) might reduce both the cost and the energy consumption for sensing in the near future [RSF+04].

1.1.2 WSN Software

The WSN software is running on the mote's μC and arbitrates the system. In particular the software ensures regular sensor readings, processes data and communicates with neighboring motes. All this needs to be performed on a very resource limited, deeply embedded and distributed system. This makes the following paradigm of utmost importance when designing WSN software:

> The limitations of the motes processing capabilities and the distributed nature of the network suggest minimizing the complexity, in particular in the protocol design.

Operating System

WSN applications usually run on top of a lightweight operating system (OS), including a communication stack especially optimized for the application at hand. The OS is trimmed to its fundamentals for running a sensor node and many functionalities of a general purpose OS such as multi-user operation, file system, graphical user interface and memory swapping are not required. On the other hand services like *concurrent tasks*, energy management, interrupts, a small memory footprint, peripheral access and modularity are most essential.

A very lightweight operating system is the most widely used TinyOS [HSW+00, LMP+05] framework providing an event-driven model for handling concurrency. Internal and external events, such as a timer or an incoming packet, are sent to the appropriate event handler that can post a task to a queue for being scheduled some time later. Contiki [DGV04] enhances this event handler with so called *protothreads*. For a minimized overhead, these lightweight threads without stack requirements provide most of the advantages of a threaded architecture. Preemption however is not possible and local variables are not supported due to the missing stack. BTnut [BBDM05] and *t-kernel* [GS06] on the other hand provide a truly threaded architecture with all its flexibility, yet imposing quite some processing and memory overhead for the system. *t-kernel* is even going a step closer towards a more general OS design, providing OS protection, which allows accessing the node even if the application crashes.

Communication Stack

For communication purpose the OS usually incorporates a small communication stack, similar to the ISO/OSI reference model, however without the layers 4–6. Basically there is the radio, the medium access control (MAC), a network layer and the application. Traditional communication protocols such as IEEE 802.11 or TCP/IP provide a lot of flexibility and maximized performance (e.g., throughput), but are not well suited for low-power operation of a deeply embedded wireless systems due to the imposed overhead. Should a TCP/IP stack be a necessity for the application, the so called uIP stack [Dun03] minimizes the overhead while maintaining interoperability and RFC standards compliance with TCP/IP.

In traditional internetworks, such as TCP/IP in the heterogeneous Internet, issues like latency, fairness, congestion control and maximized throughput are integral parts. WSNs on the other hand focus primarily on maximizing the network's lifetime, having these "traditional" issues as a secondary goal, e.g., maximize the throughput for a certain energy budget. This unique requirement lead to a large set of MAC and routing protocols specifically designed for being used in WSNs. These protocols

on the different layers are not necessarily strongly detached from each other (as proposed by the ISO/OSI model) and cross-layer functionalities are often introduced for an improved performance [HC02]. For instance, Dozer [BvRW07] combines the MAC and routing into one rather complex building block, resulting in a very energy-efficient operation in return for a reduced flexibility.

Medium Access Control

It is the MAC protocol that arbiters the radio state and in particular duty cycles the radio for an largely minimized energy consumption, trading off bandwidth and latency [LH05]. In the ideal case, the sender switches the radio on, quickly sends the packet(s) and switches the radio off immediately, whereas the receiver's radio is only switched on to receive the packet. Otherwise, energy is wasted by either one of the following five reasons: (1) *idle listening* occurs if the node is listening but not receiving data, (2) *missing* results if a node is sending but the intended recipient is not listening, (3) *overhearing* happens if a node receives a message meant for another node, (4) *collisions* usually result in multiple messages being lost, or (5) *protocol overhead* such as for medium reservation or synchronization.

A multitude of different MAC protocols especially designed for WSNs emerged in recent years. Many general purpose ones, such as S-MAC [YHE02], B-MAC [PHC04], WiseMAC [EHD04], yet most of them are designed for specific application scenarios such as Crankshaft's [HL07] dense networks, LMAC's [HH04] minimized interference or D-MAC's [LKR04] optimized latency. The imposing question of how to select the most suitable protocol for a particular application is discussed in detail in Chapter 3.

Networking

Generally, it cannot be assumed that all source nodes are in communication range with a sink, requiring a multi-hop operation for forwarding the data. This multi-hop architecture commonly results in an increased load and energy consumption for nodes close to the sink. If latency is not an issue, *data aggregation* [KEW02] techniques can be introduced for minimizing this effect.

Routing in wireless network can often benefit of assumption given by the application at hand:

- The traffic pattern is well defined, e.g., every node samples data every 10 min and sends this data immediately.

- Peer-to-peer communication is rarely required. Instead, the communication strategies are many-to-one (nodes report to the sink) and one-to-many (sink notifying nodes).

- The packet size is usually very small, i.e., in the order of tens of bytes compared to the Internet's thousands of bytes. This makes the protocol header a substantial part of the overall message size and suggests minimization.

- The nodes are resource limited and favor simple protocols with little processing overhead and memory requirements, e.g., no huge routing tables.

- Due to the low data rates, the protocol overhead, e.g., routing beacons, can account for a substantial part of the overall message count and should be sent cautiously.

These special characteristics resulted in various different routing strategies for WSNs. Some of the most prominent examples are *Mintroute* [WTC03], which builds a tree with a cost metric based on the link quality, *Rumor Routing* [BE02], which uses random gossiping, and *Directed Diffusion* [IGE00] being based on a publish/subscribe scheme. These routing strategies however do not provide a reliable data transfer to the sink, which requires more elaborate approaches as discussed subsequently.

1.2 Reliable Data Transfer in WSNs

If reliability is essential for the application, the network needs to handle errors in the wireless channel (short-term packet loss, long-term link failures) and the possibilities of node failures. Similar to traditional internetworks, this can be achieved on different layers, distinguishing between single-hop reliability and end-to-end reliability.

Single hop reliability is a common feature of WSN MAC protocols. This single-hop reliability is usually achieved by sending an immediate *acknowledgement packet* (ACK) after a successful reception of a data packet. Hence the sending node will immediately switch the radio to reception mode, waiting awhile for the ACK. Is the ACK missing, the message is either retransmitted at a later point, or the message is dropped and a signal is sent to the network layer. However, the missing of the ACK does not necessarily indicate the failure of the data packet, since the ACK message can get lost as well. Rerouting in case of a link failure is not

possible for the MAC protocol since the network topology is unknown in this layer.

This task of rerouting data packets in case of a failure is part of the network layer. However, surprisingly little research has been done on providing reliable end-to-end message delivery. The transport protocols that do address link failures (e.g., ESRT [SAA03], RMST [SH03], and PSFQ [WCK02]) include techniques like retransmissions and path diversity to overcome the errors on individual links, but do not provide end-to-end reliability. Advanced protocols like MMSPEED [FLE06] provide probabilistic bounds on end-to-end delivery ratios, i.e., offering QoS guarantees on reliability and latency, but ignores energy consumption.

If the amount of data is known at the receiver, e.g., periodic traffic, a complementary end-to-end acknowledgement scheme with retransmission can increase the delivery quality of the transport protocols as implemented in [DHS+07]. Specialized protocols like Flush [KFD+07] for bulky data or RCRT [PG07] for concurrent streams already incorporate such a retransmission scheme. While the end-to-end acknowledgement scheme increases the reliability, it suffers greatly from high costs and long (multi-hop) latencies, making it unsuitable for time-critical wireless sensor networks.

An effective, but expensive, approach to handle communication errors is to use flooding [TG01]. Quite often network-wide redundancy is not needed and partial flooding suffices. For example, the GRAB protocol [YZLZ05] uses a credit mechanism to specify how many additional hops may be made to reach the destination, effectively creating a "wide-path". GRAB requires the set-up and maintenance of a gradient field towards the destination, hence, is only applicable to a few, popular destinations like the sink(s) in an alarm system. The DFRF framework [Mar04] generalizes this idea and allows for easy creation of tailor-made partial-flooding protocols. Unfortunately, DFRF and GRAB do not integrate well with the underlying MAC layer making it difficult to control latency and energy consumption.

Multi-path routing [LG01, Ban96], that is having k node or link distinct routes to the sink increases the robustness by sending the messages over multiple routes. However, the distributed setup of these routes is expensive, in particular since the scheme has to adapt to changes in the topology.

An issue of particular importance for safety-critical systems is the detection of failed nodes, which compromise the integrity of the system. A straightforward solution is to make use of heart-beat style failure detectors where nodes periodically send out a message notifying neighbors of their status. Wang and Kuo extend this idea to a two-phase gossiping protocol suited for ad-hoc networks [WK03]. Although very robust,

information propagates slowly and at high cost. Recent work by Rost and Balakrishnan shows that more-advanced failure detectors help in reducing the message overhead [RB06], but latency remains an issue.

Ensuring a reliable detection of failed nodes (or permanent link failures), which compromise the integrity of the system, is an issue of particular importance for safety-critical applications. A straightforward solution is to make use of heart-beat style failure detectors where nodes periodically send out a message notifying neighbors of their status [WK03]. Although very robust, information propagates slowly towards the sink and at high cost. Advanced failure detectors and monitoring tools like Memento [RB06] and NUCLEUS [TC05] do address these issues, but in return raise the number of false alarms. This is acceptable for debugging purposes, but not for safety-critical systems as it significantly raises the operational costs.

1.3 Contributions

Safety-critical networking demands a timely and robust message delivery despite the limited energy resources. However, these strong demands cannot be met with current state-of-the-art communication schemes. This thesis provides the necessary fundamentals for enabling safety-critical WSNs. In particular, this thesis makes the following main contributions ordered by their importance:

1. **Routing Strategy for Safety-Critical WSNs**
   The routing scheme *Dwarf* provides a delay aware, energy efficient and robust forwarding for safety-critical messages. The energy consumption is greatly reduced by a minimized message overhead and running the nodes with a very low duty-cycle. Despite this low duty cycle. The timeliness and robustness is achieved by a delay aware forwarding of the message over multiple paths. The topology is maintained with the node and topology monitoring scheme *DiMo* . This scheme ensures with marginal overhead that all nodes provides multiple paths to the sink and that all these paths are observed continuously and are being replaced if necessary. Hence short-time drops in the link quality are absorbed having multiple paths and messages, whereas a failed link is replaced by DiMo's topology maintenance. DiMo further combines the topology monitoring with the node monitoring and detects a failed node within a required detection time. In conjunction, Dwarf and DiMo is the first solution that meets the demanding requirements of safety-critical event monitoring.

2. **MAC Framework**
   The MAC protocol allows saving by far the largest amount of energy by duty cycling the radio. A multitude of MAC protocols especially designed for WSNs are available, all of them allowing parameterization for an optimized performance. This possibility for parameterizing the protocols makes it infeasible to benchmark them exhaustively by neither simulation nor implementation. With a model-based approach, this thesis presents the first extensive study for comparing WSN MAC protocols, which allows selecting the MAC protocol that suits the best for the application at hand. In particular the study highlights that WiseMAC shows the best performance for a safety-critical system and is an integral part for Dwarf's and DiMo's energy efficient operation.

3. **Maintenance and Initialization**
   Protocols that require longevity need to be tuned for an optimized operational phase. The maintenance and in particular the most critical initial installation of the network is often greatly neglected. This part of the network's life cycle can drain a substantial part of the available energy budget when the network is not (yet) operational. This thesis proposes the NoSE protocol-stack enhancement, which allows switching the network between the operational state and a deep sleep state. The deep-sleep state allows for energy savings while performing maintenance. The network can be woken up any given time. During NoSE's swift and time-bound start up, a comprehensive neighbor assessment provides a solid basis for the subsequent network topology set up and operation.

4. **Channel Characterization**
   Safety-critical wireless systems are often deployed in indoor settings with moving people and many sources of interference. The imposing question is, what the impact of such an adversarial setting on the wireless channel is. This thesis includes an extensive study of the wireless channel for an indoor, office-like scenario. This is in a strong contrast to most available studies, assuming artificial node layouts. In particular it is shown that many links suffer from unstable behavior and that a link's total loss of communication in the order of a few seconds is not uncommon. These observations suggest a high degree of redundancy and resulted in Dwarf's multiple messages and DiMo's redundant paths being continuously monitored.

1.4 Thesis Outline

This thesis provides the foundation for safety-critical event monitoring in WSNs and is structured as follows:

Chapter 2 details the wireless medium that has to be coped with. Extensive measurements show that wireless communication always has to account for random packet losses. In particular, wireless communication links can show sudden changes in the link quality or even complete drop outs in the order of a few seconds. It is further discussed how wireless links can be assessed efficiently.

Chapter 3 provides insight into the multitude of MAC protocols especially designed for WSNs. All these MAC protocols have in common that they duty cycle the radio for a minimized energy consumption but differ in their approach. It is pointed out that a MAC protocol based on a regular channel polling are very well suited for low-power operation and that WiseMAC strikes out with the best latency vs. energy-consumption trade-off.

Chapter 4 introduces DiMo, a monitoring scheme that checks the availability of the nodes while observing the links and the topology at the same time. It accentuate that its distributed approach for monitoring is very well suited for a low latency, energy efficient operation while highlighting a very small false positive rate.

Chapter 5 introduces Dwarf, an energy-efficient, robust and dependable forwarding algorithm. The core idea of Dwarf is to use unicast-based partial flooding along with a delay-aware node selection strategy. Furthermore an integrated protocol suite is introduced that combines Dwarf and DiMo and enables safety-critical networking. It is shown in simulation and implementation that more than 99.9 % of the alarm messages are delivered in time, failing nodes are promptly reported, and the (extrapolated) network lifetime exceeds 3 years.

Chapter 6 proposes NoSE for an energy efficient maintenance and initialization of the network. NoSE uses a reduced signaling scheme to achieve an energy efficient, yet highly reactive wake-up scheme. NoSE features a synchronous and exhaustive neighbor search with integrated link assessment. Measurements on a real-world testbed and in simulation show that the network can be woken up from its deep sleep and a complete neighbor list with corresponding link qualities can be assessed within a few minutes.

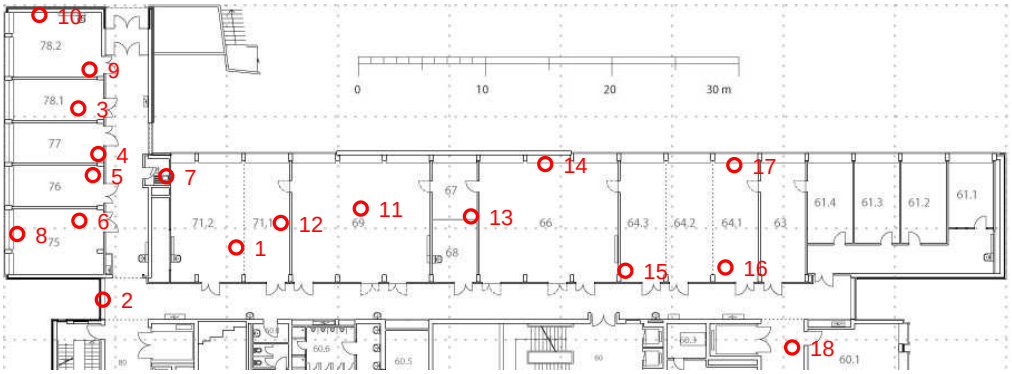
Chapter 7 concludes the thesis with an outlook for future research and a summary of the contributions.

2

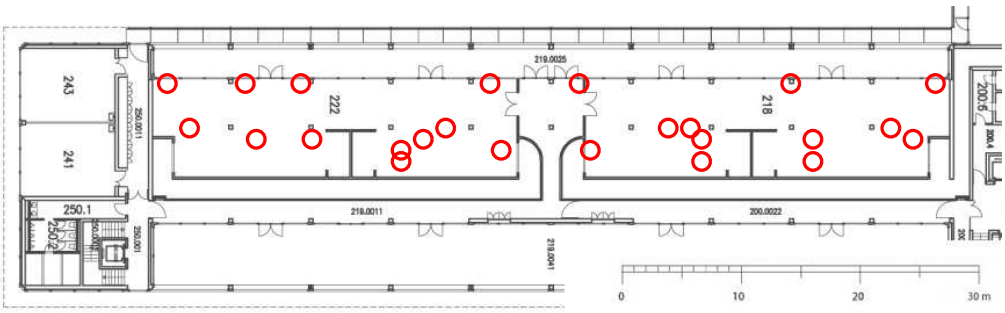
Radio Communication

Safety-critical WSNs are frequently deployed indoors. When it breaks down to the physical level of the transmission, these indoor deployments seem to be the hardest to work with [ZG03, RHL04]. This is due to the high degree of multi-path fading and interference induced by cell phones or WLANs, which results in unreliable and hardly predictable communication links. This chapter analyzes in detail the wireless characteristics for indoor deployments. It highlights the aspects affecting reliable communication, in particular the links' temporal behavior. Furthermore it investigates different link-estimation strategies. This allows selecting high-quality links and provides a solid basis for safety-critical networking.

Section 2.1 introduces the procedure for the measurements and presents the two sites they are conducted in. Section 2.2 analyzes the overall link performance, discussing whether there is a correlation between the link quality and the distance or the radio's signal-strength indicators that can be exploited. Section 2.3 analyzes the temporal behavior of the links, showing that sudden changes in the link quality are not uncommon and that bursts of packet failures have to be accounted for. In particular the novel σ_m metric is introduced, which represents the degree of a link's stability and conveniently allows detecting unstable links. Section 2.4 analyzes link-assessment strategies for an optimized link-quality estimation. Section 2.5 presents related work and Section 2.6 concludes this chapter, highlighting the essentials for safety-critical communication.



(a) Network $\mathcal{A}$ (Adversarial): 18 nodes are deployed in an office with various sources of interference (e.g., WLAN, Bluetooth, moving people etc.).



(b) Network $\mathcal{B}$ (Benign): 24 nodes are deployed in an infrastructural part of a building, in particular having hardly people around.

Fig. 1: The links are analyzed in a "adversarial" office scenario and in a rather "benign" infrastructural part of a building.

2.1 Experimental Setup

The link measurements are performed in two different networks $\mathcal{A}$ and $\mathcal{B}$, located in different buildings. The positions of the nodes represent a real-world topology of a possible fire-alarm network and are not placed in an artificial setup like a line or a grid. Network $\mathcal{A}$ (cf. Figure 1(a)) is deployed in an $\mathcal{A}$dversarial environment. The nodes are spread over multiple offices, i.e., there are many static obstacles such as walls, but also dynamic ones such as moving people. Furthermore there are various sources of interference due to 802.11g WLAN access points, a Bluetooth network and devices/machines like elevators, a microwave and a fridge. Network $\mathcal{B}$ (cf. Figure 1(b)) on the other hand is deployed in a rather $\mathcal{B}$enign, infrastructural part, of a building. Located on the top floor, there is a lot of infrastructure, such as machinery (220/380V), pumps and pipes. In contrast to Network $\mathcal{A}$, there are rarely people around, which greatly reduces the sources for interference.

Network	$\mathcal{A}$ (Adversarial)	$\mathcal{B}$ (Benign)
Number of nodes	18	24
Number of links	172	390
Received packets	7,936,690	7,171,839
Traced packets	10,430,000	8,280,000
Average packet-reception rate (PRR)	76.1%	86.6%

Tab. 1: Measurement overview: nearly 20 million packets are traced in two distinct deployment sites as further characterized in Figure 1.

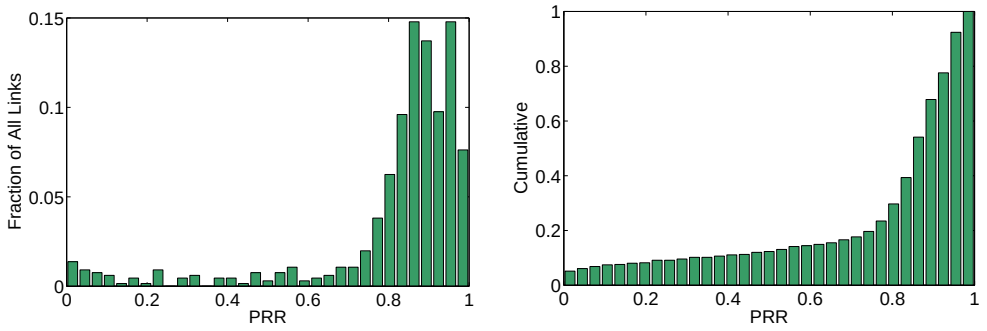
The measurements are taken on the Tmote Sky platform [PSC05], featuring an MSP430 microcontroller and the ZigBee compatible (2.4 GHz) CC2420 radio. For the link measurements one single node is selected to transmit 10,000 packets with a 200 ms interval (33.3 min). All other nodes in the network are in reception mode and trace the received packets with the corresponding sequence number, received signal strength indicator (RSSI) and link-quality indicator (LQI). The role of the transmitting node is assigned to all others in a round robin fashion. This results in a test run with every node in the network being the transmitting node once, observing all possible links in the network.

In order to capture rare effects in the communication patterns, extensive measurement are performed, tracing nearly 20 Millions packets (cf. Table 1). The measurements are taken with two distinct transmission powers (0 dBm and -5 dBm), which basically doubles the number of analyzed network topologies and links. Furthermore, the tests are carried out during workdays and nights in order to see the impact of people moving and the increased communication of the WLAN hotspots. The measurements are all performed on one single channel (26), in order to not further increase the number of parameters.

2.2 Static Link Analysis

The most fundamental quantifier for the link quality is the average packet reception rate (PRR), which represents the average quality of a link over the measured period. For instance, if 100 packets are sent, yet only 95 of them received, a PRR of 0.95 results.

The PRR provides a first insight on the overall link performance in the network. This is detailed in Figure 2 showing the distribution of the PRR for all links in Network $\mathcal{A}$. Of particular interest is the PRR range of 0.8 to 1 where most links are assigned to. This area is very sensitive when it comes down to selecting high quality links only. For instance



(a) Link-quality distribution (b) Cumulative link-quality distribution

Fig. 2: PRR Distribution in Network $\mathcal{A}$: A large fraction of the links shows a rather poor performance. For instance, 56% of the links suffer from a PRR of less than 0.9.

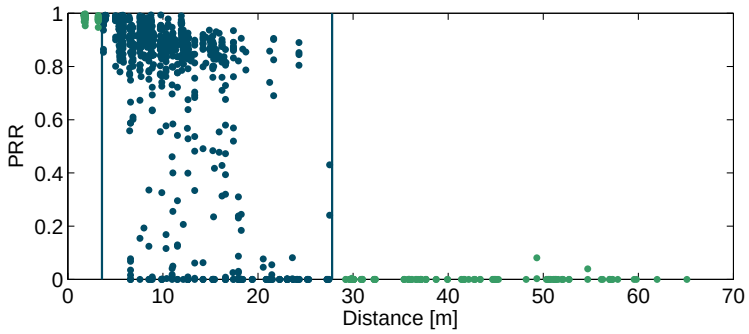
77% of all links have a PRR > 0.8, 44% a PRR > 0.9 but only 24% of the links highlight a PRR > 0.95. Hence the links need to be carefully selected when routing in a wireless sensor network, if the reliability is essential and energy should not be wasted with numerous retries.

In the following it is detailed, whether there is a dependency between the PRR and the distance or the signal-quality indicators of the radio that can be exploited for routing in WSNs.

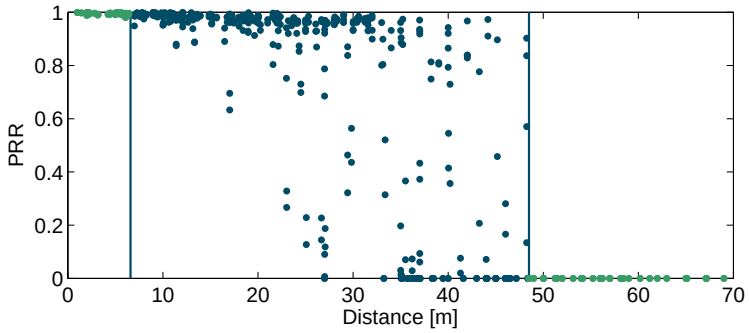
2.2.1 Spatial Behavior

Especially in safety-critical networks the exact position of the nodes is usually available. So it is of particular interest for routing protocols, whether the link quality exhibits a correlation with the distance between the nodes that can be taken advantage of. For instance if the link quality follows the widely used *unit disk graph* (UDG) model, i.e., all packets are received within a given radius and no packets beyond, a centralized algorithm at the sink can set up a sophisticated network topology only considering the provided positions of the nodes.

Previous studies however indicated that there is no clear correlation between the node distance and the PRR. Based on artificial layouts, the prevalence of the so called *gray area* was first highlighted by Zhao et al. [ZG03] and confirmed in various studies [WTC03, RHL04, ZK04]. That is, the reception range can be divided into three distinct regions: Nodes up to a certain distance d_1 from the sender yield almost 100% of the packets whereas nodes beyond a distance $d_2 > d_1$ do not receive any packets. The gray area in between exhibits a rather unpredictable link quality and nodes closer to the sink can show a worse link quality than nodes much farther away, i.e., a monotonic correlation between distance and



(a) Network $\mathcal{A}$ (Adversarial)



(b) Network $\mathcal{B}$ (Benign)

Fig. 3: PPR vs. distance: Every dot represents the PRR of one distinct link in the network. The PRR does not follow a deterministic behavior and shows a wide range, referred to as gray area (indicated with the two vertical lines), in which the reception rate varies a lot.

link quality does not exist. Depending on the environment, the gray area has shown to have a substantial extent. While the gray area measured on a parking lot covers 10% of the total communication range, this range is increased to 30% in a habitat and to 50% in an office building [ZG03]. These findings are explained by the increase in the multi-path signal delivery for the different surroundings.

The imposing question is, how this gray area is affected by real (not artificial) indoor deployments where for instance additional walls further influence the signal quality. This is detailed in Figure 3 and highlights for both Networks $\mathcal{A}$ and $\mathcal{B}$ a largely increased size of the gray area, having almost all the links being attributed to it. Network $\mathcal{B}$ shows a much wider communication range, and therefore also a larger extent of the gray area, than $\mathcal{A}$. The most evident reason for this is the increased signal attenuation in Network $\mathcal{A}$ due to the numerous walls in this site.

The core message considering the spatiality of the links is that it is hardly of use for setting up a topology in an indoor environment.

2.2.2 RSSI and LQI

Having a large percentage of poorly connected neighbors suggests to communicate with high-quality links only, which requires to estimate the link qualities. For this purpose the radio provides the two additional parameters of the *received signal strength indicator* (RSSI) and the *link-quality indicator* (LQI). Both indicators are provided with every received packet. The RSSI value represents the RF signal *strength*, yet is often claimed to be a bad indicator for the signal *quality* [ZG03]. However this claim is based on measurements with earlier radio hardware and does not necessarily hold for newer radios like the CC2420 as suggested by Srinivasan et al. [SDTL06, SL06]. The LQI correlation value is a virtual measurement of the chip error rate and is based on the first 8 symbols following the start frame delimiter.

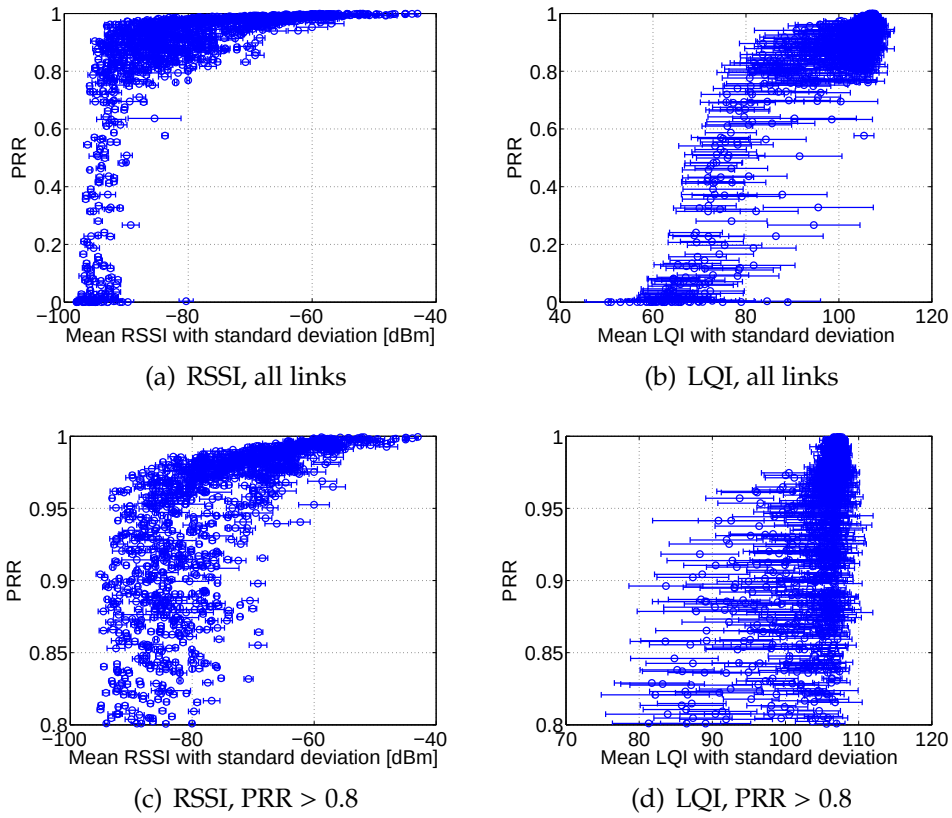
The correlation between the RSSI and the PRR is depicted in Figure 4(a) and in a close-up in Figure 4(c). It can be seen that an $RSSI > -80\,\text{dBm}$ indicates that at least 80% of the packets are received, whereas a $RSSI > -68\,\text{dBm}$ indicates high quality links with a PRR > 0.95 (except two outliers). The inverse statement on the other hand is not possible: a low signal strength does not necessarily indicate a bad link. Srinivasan et al. had similar findings, however much clearer results; e.g., an RSSI value greater than -87 dBm indicating a $PRR > 0.85$. This clearly indicates that the deployment site has a large influence on the relation between RSSI and the PRR. As a result, a parameterization independently of the actual deployment situation is not possible.

The mean standard deviation of the RSSI range is about 1 dBm in both Networks $\mathcal{A}$ and $\mathcal{B}$. This small standard deviation makes the RSSI a strong indicator for determining high-quality links. It is analyzed in Section 2.4 how this can be exploited for estimating the link quality.

The LQI on the other hand is a rather bad indicator for high-quality links as depicted in Figure 4(b) and 4(d). Especially the large standard deviation (represented by the bar) of the different packets received from one particular sender, makes the LQI a rather unattractive indicator. On the other hand the LQI indicates bad quality links, e.g., a LQI < 80 usually indicates a link with a PRR < 0.8.

2.3 Temporal Link Analysis

Characterizing the link quality with the average PRR provides a first impression on the link quality, but lacks the information whether the link quality varies over time. This temporal behavior can be depicted by applying a sliding window over the trace for calculating a temporal



(a) RSSI, all links (b) LQI, all links

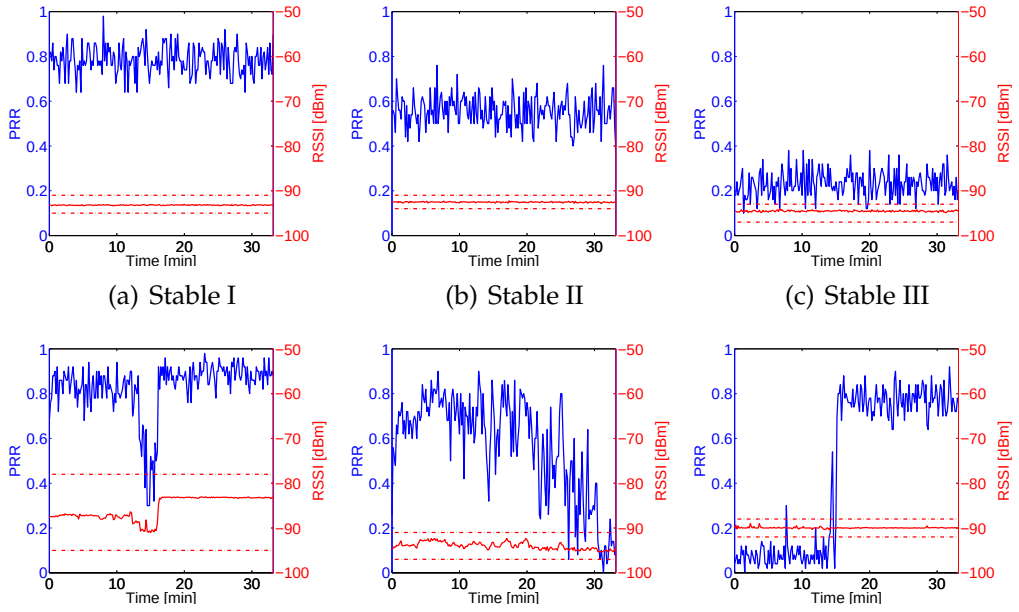
(c) RSSI, PRR > 0.8 (d) LQI, PRR > 0.8

Fig. 4: Correlation between the link-quality indicators RSSI and LQI and the corresponding PRR: The RSSI is better suited to indicate high-quality links.

PRR. Figure 5 shows sample traces of the temporal behavior using a window size of $m = 100$: the upper plots (a-c) highlight links that are stable over time, i.e., the PRR varies slightly only. This is different for the lower plots (d-f). The first plot (d) shows a distinct dropout lasting for 3.5 minutes, the second (e) a degrading behavior and the third (f) a suddenly increased link performance after showing a bad link quality for 15 minutes.

An interesting observation is made if the RSSI value is taken into account. The stable links go along with the expected stable RSSI value. However the same holds for the unstable links (f) and (g), which also exhibit a stable RSSI. Only the dropout observed in link (d) is reflected by a distinct change of the RSSI.

If the link stability needs to be characterized by a single value, neither the average PRR nor the RSSI link metric can reflect them. This finding has already been reported for the PRR [CWPE05] value, yet is new for the RSSI value.



(a) Stable I (b) Stable II (c) Stable III

(d) Drop (e) Degrading (f) Step

Fig. 5: Trace of the PRR (upper curve) and the according (mean, min, max) RSSI (lower curve) of four sample links: Most links in the network show a stable link quality over time as shown in the plot (a-c). There are also links showing unstable behavior, such as dropouts (d), degradation (e) or even step behavior (f). Only plot (d) shows a distinct change of the RSSI value.

2.3.1 Metrics

In order to quantify the temporal behavior of a link, Cerpa et al. [CWPE05] proposed to use the so called *required number of packet* (RNP) metric, representing the number of required (re)transmissions upon a successful reception. While the RNP metric is very useful for predicting the retransmission overhead for a distinct link of the routing protocol, it is not well suited to reflect the link stability. Instead this thesis proposes to use the standard deviation σ_m of the *temporal* PRR. The degree of the link stability is reflected in the comparison of the link's standard deviation with the one of a (stochastic) Bernoulli Process and is further detailed and analyzed in Section 2.3.2. Furthermore, it is highlighted that the σ_m metric can conveniently be used to distinguish between stable and unstable links.

All the previously introduced metrics, namely the PRR, RNP and the new σ_m, represent a link with a single number only, enabling a direct comparison of links. However, rare corner cases of a link's behavior will evidently average out. When designing a network protocol, it is of particular interest whether burst of packet losses occur, which will make communication on a particular link temporarily impossible. For

instance the retransmission strategy of an unacknowledged packet can be designed more effectively if the behavior of such bursts is known. Willig et al. [WM06] suggest to analyze packet-burst failures based of the relative frequency of a certain burst size. That is, compared to all bursts, how many consist of a single packet failure, how many consist of two consecutive failed packets and so on. This aspect is further detailed in Section 2.3.3.

2.3.2 Link Stability

The stable links in Figure 5 (a)-(c) exhibit a rather constant packet reception rate, whereas the unstable links (d)-(f) show a changing rate over time. In particular, the link failures of these unstable links have time dependencies and hence the individual link failures have dependencies. For a stable link on the other hand it is assumed that link failures are independent and provide a constant packet reception rate for every transmitted packet. The novel σ_m metric introduced in the following measures the degree of volatility. This is achieved by comparing the standard deviation of the link σ_m^{trace} with the one of a stochastic process σ_m^{rand} with independent link failures but the same average PRR. The quotient $\gamma_m = \sigma_m^{\text{trace}}/\sigma_m^{\text{rand}}$ conveniently represents the level of the link stability.

2.3.2.1 Standard Deviation

The trace of a link measurement can conveniently be represented by a series of '1's (packet received) and '0's (packet not received). In order to calculate the standard deviation σ_m^{trace}, the trace is divided into blocks of m packets. For each of these blocks the PRR PRR_m^i is calculated, which results in a sequence of temporary PRR values $(PRR_m^1, PRR_m^2, \ldots, PRR_m^k)$. The standard deviation is then defined as the root-mean-square deviation of the temporary PRRs from their mean P_r:

$$\sigma_m^{\text{trace}} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(PRR_m^i - P_r)^2}. \tag{2.1}$$

For a perfectly stable link on the other hand, the packet failures are independent. Such a sequence is a Bernoulli process, i.e., a sequence with independent random variables having each the probability P_r (equal the PRR of a stable link) of being one. In order to map this stochastic sequence to a temporal PRR, m packets are aggregated to a temporary PRR which leads to a binomial distribution $B(m, P_r)$. The standard deviation σ_m^{rand} of a stochastic and perfectly stable link can then be expressed as

$$\sigma_m^{\text{rand}}(P_r) = \sqrt{\frac{1}{m} P_r(1 - P_r)}. \tag{2.2}$$

Figure 6 shows the standard deviation of all the traces measured in Network $\mathcal{A}$ and $\mathcal{B}$. The standard deviation of the Bernoulli process (stable link) is plotted with a straight line, whereas all the different traces taken in the two networks are represented by a circle. The difference between the two networks is very distinct, showing a lot of links with a largely increased instability in Network $\mathcal{A}$ but only a few in Network $\mathcal{B}$.

2.3.2.2 Stability Criteria

A stable link should have a similar standard deviation as the Bernoulli process with the same average PRR has. An unstable link on the other hand is expected to show an increased standard deviation. The *stability factor* γ_m can be used to determine the degree of the link's stability:

$$\gamma_m = \frac{\sigma_m^{\text{trace}}}{\sigma_m^{\text{rand}}(P_r)} = \sqrt{\frac{m}{k(1 - P_r)P_r} \sum_{i=1}^{k} (PRR_m^i - P_r)^2} \tag{2.3}$$

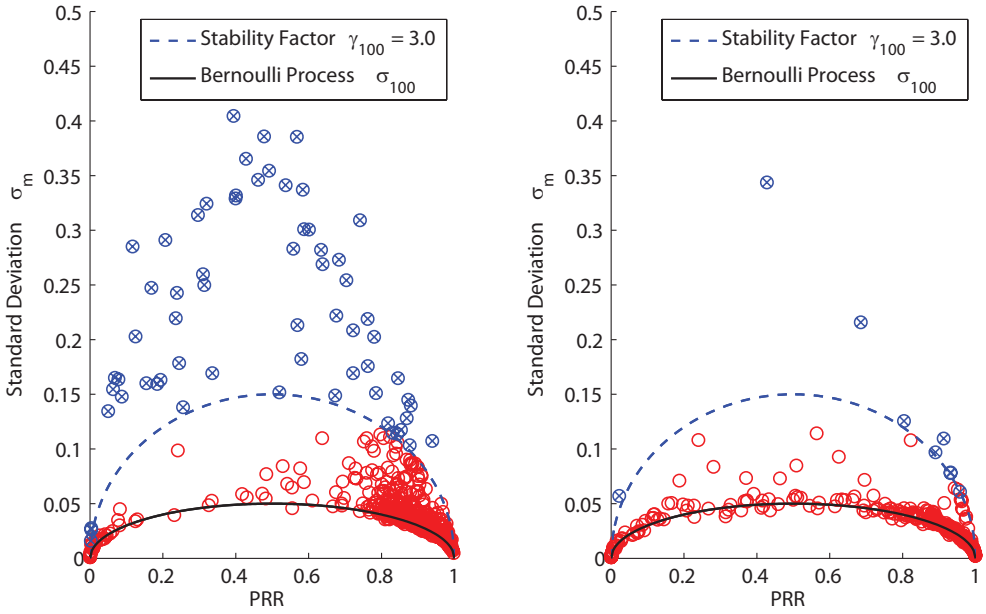
The upper traces (a-c) shown in Figure 5 show all a stable behavior which is also reflected by the stability factor γ_{50} highlighting a value of about 1, namely 1.11 (a), 1.00 (b), and 0.97 (c). Having a $\gamma_{50} < 1$ in (c) denotes that this link has an even smaller standard deviation than the Bernoulli process. This is always possible, since a finite stochastic process will usually show a minor deviation from the infinite Bernoulli process assumed in (2.2).

The lower plots (d-f) show an increased instability from the left to the right: whereas the leftmost trace (d) only shows a short dropout, the step behavior of the rightmost link is clearly unstable. This behavior is reflected by the stability factor γ_{50} showing an increasing value for the three traces, namely 2.33 (d), 3.23 (e) and 4.84 (f).

The index m of the stability factor γ_m represents the granularity of the link instabilities: Whereas a small value m shows very short time fluctuations of the link, a larger value of m allows detecting long term instabilities. In order to detect whether a link is stable, it should neither show increased short nor long time fluctuations compared to the Bernoulli process.

Definition 1. Stable Link Criteria:

$$\gamma_{m_i} < \alpha_i \quad \forall i \in [1, I]$$



(a) $\mathcal{A}$, $m = 100$, $\gamma_m = 3.0$ (b) $\mathcal{B}$, $m = 100$, $\gamma_m = 3.0$

Fig. 6: Link stability: The solid line represents the standard deviation of a Bernoulli Process; the dashed line indicates the boundary of stable links. Each circle represents an instance of a measured trace. A manual inspection showed considerable more unstable links in Network $\mathcal{A}$. This finding is confirmed by the σ_m metric.

This definition allows to determine a set of factors α_i that defines a global stability criteria. In order to determine this set α_i, a manual inspection of all measured traces is required. In particular $\gamma_{100} > 3.0$ revealed to detect links with a lot of short time fluctuations, and is represented by the dashed line in Figure 6. Long term instabilities on the other hand, can be detected using a second criteria $\gamma_{500} > 4.8$. Combining the two criteria finally allows distinguishing the stable from the unstable links.

The manual inspection of the traces shows a largely increased number of instabilities in Network $\mathcal{A}$ deployed in a rather malicious environment compared to Network $\mathcal{B}$ deployed in a benign environment. The two constraints for short and long-term instabilities allow quantifying this observation and is presented in Table 2. Most notable: a routing protocol in Network $\mathcal{A}$ has to deal with 10.4% unstable links whereas Network $\mathcal{B}$ shows only 1.9% of such links. This inherently suggests that the parameter optimization of the topology maintenance depends largely on the network's deployment site.

Network	$\mathcal{A}$ (Adversarial)	$\mathcal{B}$ (Benign)
Number of traces	656	828
Number of packets per trace	10,000	10,000
Packet frequency	5 Hz	5 Hz
Short time instabilities ($\gamma_{100} > 3$)	9.6%	1.1%
Long term instabilities ($\gamma_{500} > 4.8$)	9.4%	1.9%
Fraction of stable links	89.6%	98.1%
Fraction of unstable links	10.4%	1.9%

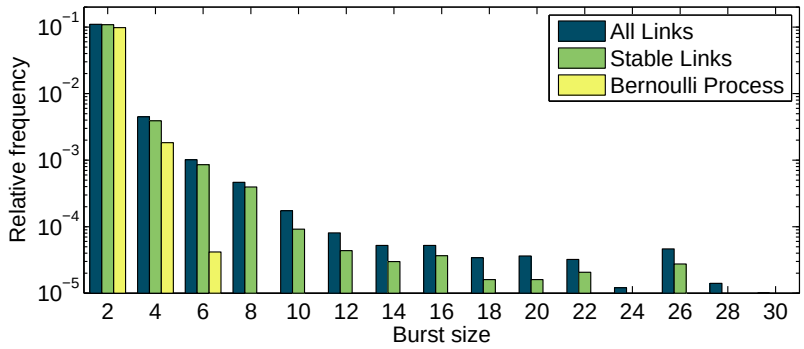
Tab. 2: Link stability: Network $\mathcal{A}$ shows an increased number of unstable links.

2.3.3 Burst Size

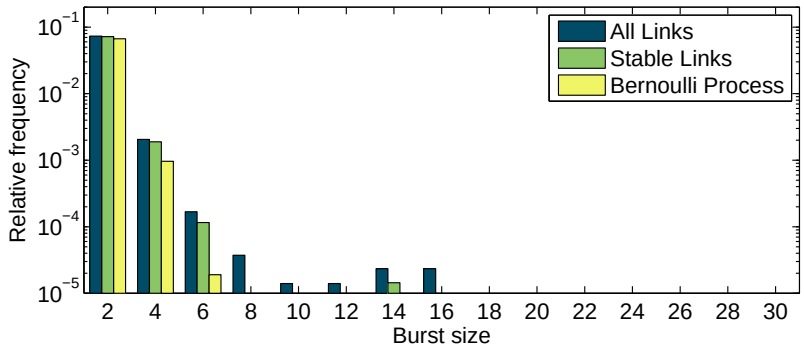
Analyzing the occurrence of bursts was proposed by Willig et al. [WM06]. Their measurements are based on a setup with 10 receivers (RFM TR1001), lined up on plank having a node distance of 30 cm (2.7 m in total). They measured the burst occurrence with the same packet frequency of 5 Hz than the measurements taken in Network $\mathcal{A}$ and $\mathcal{B}$, allowing a comparison of the results. However, their testbed was rather small and did not show nodes close to the (maximum) reception range, which resulted in a minimal PRR of 0.908. In order to compare their results, only the links with this minimal PRR are selected from Network $\mathcal{A}$ and $\mathcal{B}$.

The artificial lineup of the nodes in [WM06] shows much less packet bursts, having 98.4% "single bursts" compared to the 91.2% and 95.0% in Network $\mathcal{A}$ and $\mathcal{B}$, respectively. In particular Network $\mathcal{B}$, not having distinct interfering networks close by, also shows an increased number of bursts. This suggests that the 868 MHz frequency, used by the RFM TR1001, is much less susceptible to burst errors than the 2.4 GHz frequency of the CC2420.

Packet bursts become even more frequent, especially in Network $\mathcal{A}$, if all links with a PRR > 0.8 are considered (cf. Figure 7). Even though more than 80% of all packet bursts consist of a single packet failure, the hostile Network $\mathcal{A}$ shows packet bursts with up to 30 subsequent packet misses, i.e., six seconds with no possibility for communication. Network $\mathcal{B}$ on the other hand shows a rather similar behavior as the Bernoulli process, especially when only the stable links are accounted for. This correlation of the packet-failure bursts and the link stability does not hold for Network $\mathcal{A}$. This indicates that the adversarial environment in Network $\mathcal{A}$ is likely to introduce frequent short-time outages in the order of seconds. Hence an aggressive retransmission strategy is likely to waste a lot of energy and might even result in unnecessary topology changes due to subsequent unsuccessful retries.



(a) Network $\mathcal{A}$ (Adversarial)



(b) Network $\mathcal{B}$ (Benign)

Fig. 7: Burst size of packet failures for links with PRR > 0.8. Network $\mathcal{A}$ exhibits a distinct number of packet-failure bursts with a size of up to 30 packets, i.e., 5 seconds without the possibility for a successful packet transmission. These bursts are less common in Network $\mathcal{B}$.

2.4 Efficient Link Assessment and Estimation

For safety-critical, low-power routing protocols it is essential that only high-quality links are selected, which allows for a reliable and efficient operation. Typical routing protocols usually select a random neighbor for their routing and assess the link during operation, exchanging nodes in the routing table with other (random) neighbors when conditions change. This approach leads to a lot of inefficient adaptation of the network topology, possibly also oscillations, in particular during the start-up of the network [DHS+07].

In contrast to this "best practice" this thesis suggests assessing a node's link quality prior to addition to the routing table. A neighbor will only be considered if the link provides a reasonable link quality. Of course, the continuous link assessment should still take place and will possibly benefit from the novel link-assessment scheme executed at startup. It

is discussed in the following, how such an initial link estimation can be performed most efficiently by identifying the best links with high probability. In particular the impact of the number of packets and the packet interval for the link assessment is analyzed. Moreover it is shown that the use of the RSSI value allows to further increase the accuracy of the link-quality estimation.

2.4.1 Evaluation Method

The evaluation of the different strategies for the link assessment is based on all $N = 1448$ measured traces in both networks $\mathcal{A}$ and $\mathcal{B}$ (cf. Table 2). In particular, the two different networks are not distinguished for the following evaluation. Furthermore only the stable links are considered in the evaluation, since the hardly predictable unstable links would distort the comparison of the different estimation strategies.

The average packet reception rate X of a trace is estimated by means of subsampling [PRW99]. This means that only a few samples (packets) of the trace are considered for estimating the link quality. In the following it is detailed how different estimation strategies are analyzed and compared based on so called *estimation patterns*.

2.4.1.1 Subsampling

The subsampling is based on a pattern illustrated in Figure 8. In this example the pattern selects the packet numbers 1, 2, 5, 6, 9 and 10 from the trace and estimates the link quality based on the success rate of these six packets. By using a sliding window technique, this single pattern will provide $n \simeq 10,000$ link estimations $(x_1 \dots x_n)$ for every evaluated trace. For instance, starting the pattern on the first packet results in $x_1 = 4/6$ received packets, whereas starting from the fourth packet results in receiving all packets $x_4 = 6/6$ (cf. Figure 8).

The resulting n estimations $x_1, \dots, x_n$ are compared with the average packet reception rate X. This is done by calculating the variance of the estimations from X and hence averaging the squared distance from all x_i to X:

$$\sigma^2_{\text{Trace}} = \frac{1}{n} \sum_{i=1}^{n} (X - x_i)^2. \tag{2.4}$$

The smaller σ^2_{Trace} gets, the more accurate the estimation pattern is.

If the estimation strategies are adopted for all $N = 1448$ traces, a variance vector V with N elements results:

$$V = \left(\sigma^2_{\text{Trace1}}, \sigma^2_{\text{Trace2}}, \dots, \sigma^2_{\text{TraceN}}\right)'. \tag{2.5}$$

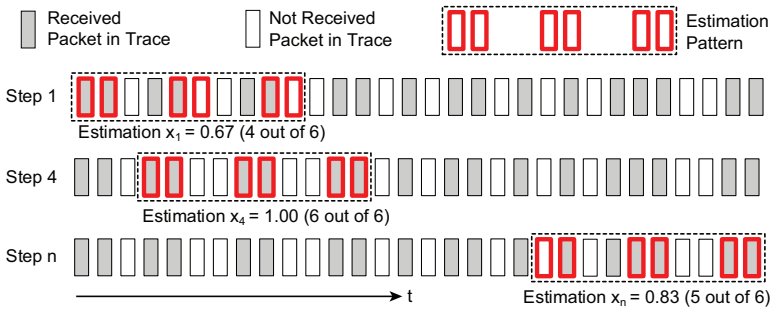


Fig. 8: Based on the measured traces a sliding window is used to analyze the performance of the estimation pattern.

2.4.1.2 Comparing Estimation Strategies

For the differentiation of two estimation strategies $\mathcal{X}$ and $\mathcal{Y}$, the two corresponding variance vectors ($V^{\mathcal{X}}$, $V^{\mathcal{Y}}$) are compared. This however requires to standardize the corresponding elements, which is necessary to weight all evaluated traces of the link estimation the same. Otherwise the evaluation would mainly be based on the traces with the highest variance.

$$v_i'^{\mathcal{X}} = v_i^{\mathcal{X}} / v_i^{\mathcal{X}} = 1, \quad v_i'^{\mathcal{Y}} = v_i^{\mathcal{Y}} / v_i^{\mathcal{X}} \quad \forall i \in [1, N] \tag{2.6}$$

The averages ($\overline{V'^{\mathcal{X}}}$ and $\overline{V'^{\mathcal{Y}}}$) of the standardized vectors are then set into proportion. This allows comparing the accuracy of the two underlying estimation strategies:

$$V' = \overline{V'^{\mathcal{Y}}} / \overline{V'^{\mathcal{X}}} = \overline{V'^{\mathcal{Y}}} \tag{2.7}$$

Hence a value $V' < 1$ denotes that the estimation strategy $\mathcal{Y}$ is more accurate than $\mathcal{X}$.

2.4.2 Static Estimation Pattern

The following, rather simple estimation pattern, is going to be the reference for the subsequent more elaborate estimation strategies and is used for the standardization (2.6). Hence the average of the corresponding standardized variance vector V^{SA} is equal one ($\overline{V'^{SA}} = 1$).

Standard Pattern (SA): The standard pattern consecutively sends 20 packets at a frequency of 5 Hz. The estimated link quality is the PRR over these 20 packets.

A solid link estimation should not only differentiate between high and low-quality links, but should give an accurate estimation of the link quality. In terms of retransmission overhead, it makes a big difference whether a PRR of 85% or 95% can be expected.

The standard pattern (SA) reliably performs the task of separating the good from the bad links: The links with a PRR < 0.5 are estimated with a negligible probability of 0.02% of having a PRR > 0.8, never estimating a PRR > 0.9. High-quality links with a PRR > 0.9, are estimated with a probability of 1.84% for having a PRR < 0.8 and only 0.02% for having a PRR < 0.5.

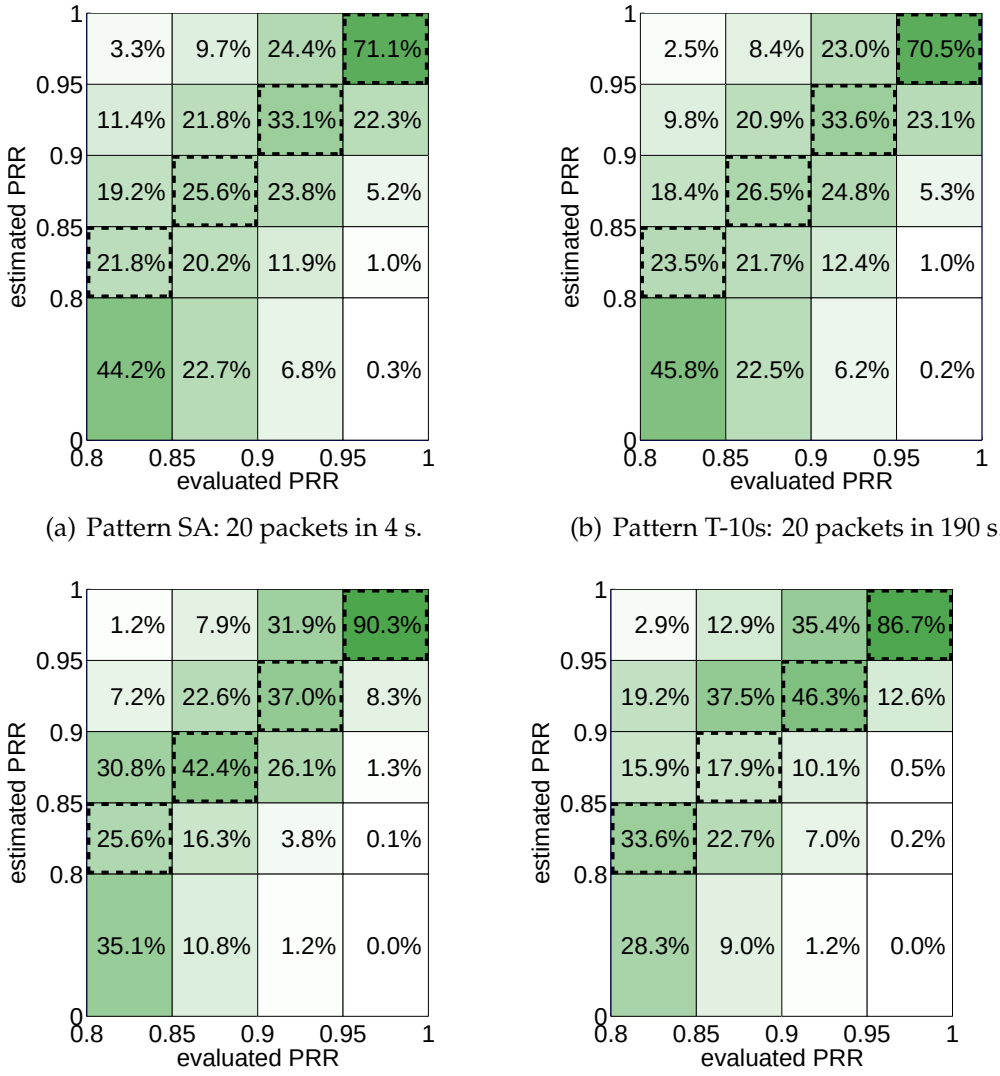
The accuracy of estimating high-quality links is shown in Figure 9(a), depicting the detailed assessment for the links with a PRR > 0.8 in steps of 0.05. The links with a very high PRR > 0.95 are correctly assessed with a probability of 71.1%. For the links with a PRR lower than 0.95, the probability for an exact link estimation drops significantly to less than one third. On the other hand, the probability for a distinct misjudgment, i.e., more than 10% difference between the evaluated and estimated PRR, is also less than one third.

The SA solidly distinguishes between high and low-quality links. However, the task of assessing the links with a fine granularity is much harder and demands for a refinement of the pattern.

Patterns T-Y: These patterns estimate the link with the PRR out of 20 packets. The value Y indicates the packet interval and therefore determines the estimation period. For instance T-10s traces a packet every 10 s, requiring 190 s for the estimation. T-0.2s is equal to the standard pattern (SA).

The evaluation of the patterns T-Y is shown in Figure 10(a). It can be seen that a lower packet frequency improves the estimation; however, the improvement gets insignificant for packet intervals longer than 5 s. A relationship to the burst size of the packet losses evaluated in Section 2.3.3 is very likely. If such a burst occurs during the estimation, and the estimation is mainly based on packets of this burst then the resulting estimations will likely be incorrect. Therefore, a packet interval that is longer than the majority of the burst sizes (i.e., a few seconds) improves the quality of the estimation. The pattern T-10s is shown in Figure 9(b) exhibiting a slightly increased estimation accuracy compared to the SA.

More complicated patterns have been studied also. For instance taking 5 packets with a frequency of 1 Hz, waiting for 20 seconds and taking another 5 packets with a frequency of 1 Hz and so one until 20 packets are sent. These patterns improved the estimation in the same scale as the algorithms T-X. Hence, the basic characteristic for the pattern's efficiency is the temporal extension of the link estimation.



(a) Pattern SA: 20 packets in 4 s.

(b) Pattern T-10s: 20 packets in 190 s.

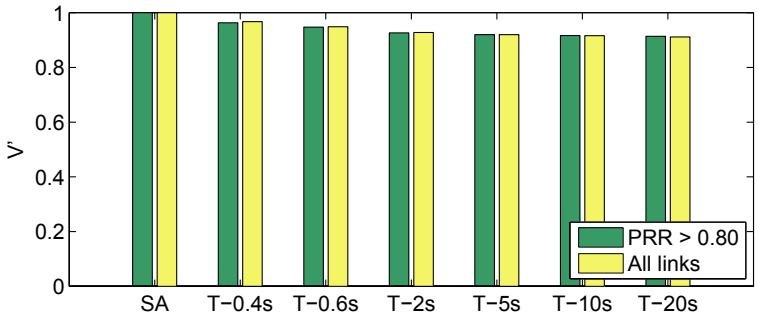
(c) Pattern N-50: 50 packets in 10 s.

(d) Adaptive Pattern R: in average 18 packets (31 max) in 6 s.

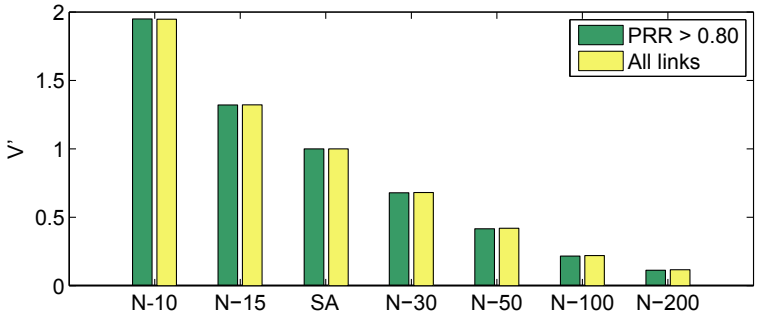
Fig. 9: Detailed comparison of different estimation strategies for PRR>0.8. The target squares are highlighted by dashed borders.

Patterns N-Z: These patterns consecutively send Z packets with a frequency of 5 Hz. The estimation is performed by calculating the PRR over these Z packets. The algorithm N-20 is equal to the standard pattern (SA).

The number of packets has a clear impact on the quality of the estimation as detailed in Figure 10(b). The relation between the number of packets X and the resulting value $\overline{V'}$ (compared with the SA) is almost indirectly proportional. This means that when the number of packets is



(a) Adapting the packet frequency



(b) Adapting the number of packets

Fig. 10: Comparison of different estimation-strategy parameters.

doubled, the variance of the estimation is divided by two. This result however needs to be quantified with respect to the power and time requirements: For instance, the pattern N-50 requires 2.5 times more energy and time than the SA, but shows a distinct improvement for assessing the high quality links. In particular, 90.3% of the links with a PRR > 0.95 are estimated correctly (cf. Figure 9(c)), in contrast to the 71.1% of the SA.

2.4.3 Adaptive Estimation

As indicated in Section 2.2.2 and further detailed in Table 3, the RSSI value allows refining the link estimation. The table shows, that for high RSSI values, i.e., RSSI > −70 dBm, the average PRR is more than 98%, showing only a small standard deviation, and hence allows to detect very good links. On the other hand, a low RSSI value does not necessarily indicate a bad links. The following adaptive pattern R shows how this additional information can be used for an increased performance of the link assessment.

RSSI Range	Avg PRR	σ(Avg PRR)	Max PRR
-70. . . 0 dBm	98.2%	0.022	99.9%
-75. . . -70 dBm	94.6%	0.045	99.0%
-80. . . -75 dBm	91.5%	0.088	98.8%
-85. . . -80 dBm	88.7%	0.076	98.6%

Tab. 3: Statistics on the PRR for different RSSI thresholds: Even links with a low RSSI can have a very high PRR.

Adaptive Algorithm R: The algorithm R is based on a pattern that evaluates up to a maximum of 31 packets at 1 Hz. The algorithm is divided into three rounds: (1) If the first packet has an RSSI > −70 dBm, a PRR = 0.98 is estimated and the algorithm stops. (2) Another 15 packets are evaluated. The algorithm stops returning the PRR of the first 16 packets if this PRR > 0.9 AND min(RSSI) > −80 dBm. (3) Another 15 packets are taken into account and the PRR of all 31 packets is returned.

The evaluation of this adaptive algorithm based on all traces, considers an average number of 18 packets for estimating the link quality. In particular 36% of the estimations stopped after the first round, 13% after the second, whereas 51% of the links required evaluating all 31 packets for the estimation. With less packets considered on average, this adaptive algorithm achieves a significantly better assessment than the SA pattern ($V' = 0.75$) and is further detailed in Figure 9(d). The overall estimation accuracy is slightly below that one of the N-30 pattern ($V' = 0.68$). This indicates that for many good links a lot of energy and time can be saved by quickly assessing them, still pursuing a thorough assessment for the less predictable links. In contrast to the static estimation patterns, an adaptive algorithm requires some sort of handshake protocol, when being incorporated into a protocol.

The possibilities of using the LQI for an increased accuracy for the link estimation of an extended algorithm R has been analyzed in detail. However, such extensions could not achieve reasonably better results. Furthermore, tweaking the parameters of algorithm R for an optimized performance is not recommended, since this optimization depends mostly on the environment the target network is being deployed in.

A reasonable value for a minimal link quality (MLQ) for a neighbor in the routing table is a PRR ≥ 0.9. Selecting the neighbors without a prior link assessment will result in high likelihood that the topology needs to be changed since many links do not provide this MLQ. In the case of the analyzed networks $\mathcal{A}$ and $\mathcal{B}$, 44.6% of the initially selected links will have to be replaced during operation. On the other hand, if the adaptive algorithm R is used for a prior link assessment, this percentage is reduced

to 14.8%. Hence there are 3.0 times less network topology changes during the initialization of the network. This comes at an average cost of sending 18 packets and a 0.4% possibility of wrongly estimating such a MLQ neighbor with a bad link quality of PRR < 0.8.

2.5 Related Work

The link quality in wireless sensor networks, i.e., using low power radio devices such as the CC1000, RFM TR1000, is analyzed in several studies [ZG03, RHL04, WTC03, CWPE05, WM06, SL06]. Most of them focus on the spatial properties of wireless links, based on rather artificial node deployments such as a line or a grid. This provides a first insight into link characteristics, however it lacks to disclose effects that only occur in a less protected environments. This work on the other hand, evaluates the link characteristic of real deployments, exemplary to the ones being used in a wireless fire-alarm application. Furthermore, this study takes into account that technology moves towards ample bandwidth, packet based radios, such as the Chipcon CC2420 based on the IEEE 802.15.4 standard. Most related studies mentioned above, measure the link quality based on first generation radio hardware, which are byte stream based and operate in the sub gigahertz band.

The existence of the gray area has first been shown by Zhao et al. [ZG03] in a linear topology with a 868 MHz radio device. Woo et al. [WTC03] (grid, 868 MHz) provide similar findings based on a uniform grid over a large, essentially unobstructed indoor space. Reijers et al. [RHL04] (linear, 868 MHz) extensively investigates the gray area and the influence of the environment. They point out, that for indoor deployments, the extent of this gray area is very large due to the multi-path signal delivery. In particular there are more nodes inside the gray area than within the proper communication range. In the office-like deployments analyzed in this work, the gray area spans even further and covers almost the whole communication range.

Temporal link behavior is studied by Willig et al. [WM06] (linear within 3 meters, 868 MHz). They show that bit error occur in bursts, while packet errors do not. This latter finding is in contradiction to the finding in this work and clearly shows the large impact on the link quality in hostile environments. Cerpa et al. analyze in [CWPE05] (grid, 868 MHz) the temporal properties of wireless links and propose the RNP metric. This metric allows predicting the retransmission overhead of a link, yet is not well suited to reflect the link stability.

The RSSI and LQI values is also studied in [SL06] (random, 2.4 GHz) by Srinivasan et al.. They show that RSSI is usually a better link indicator

than LQI due to the former's smaller range of variance, a result that is confirmed in this thesis. Srinivasan et al. however lack to notice that the LQI value is well suited for indicating bad-quality links.

Link assessment and estimation for WSNs is not a well addressed topic in literature. Woo et al. [WTC03] analyze traditional techniques for the link assessment, pointing out that most of these approaches requires to overhear all neighboring traffic. They propose the so called WMEWMA(t,α) estimator, that does not rely on tracking down the sequence number of the neighboring traffic and addresses the issue of the resource constraints for maintaining the neighbor table. While this estimator allows for observing the link quality while routing, it does not assess the link quality prior adding the link to the routing table. Such an initial link quality assessment is proposed by Lal et al. in [LMH+03]. They assess the link quality over a period of a day, in order to get a good link estimate. This thesis on the other hand highlights, how link estimation can be performed within seconds.

2.6 Summary

This chapter provides significant insight into the link characteristics and behavior of a 2.4 GHz indoor environment typical for wireless sensor networks. It has been shown that a large fraction of the link suffer from random packet losses. These losses are not necessarily uniformly distributed over time, having up to 10% of the links showing temporal instabilities. It is even possible that complete dropouts (i.e., without any means for communication) lasting for several seconds occur.

For safety-critical networking, the results clearly show that redundancy in the network is indispensable. For WSNs in general it is highlighted that the topology needs to be monitored continuously and that links should and can be well estimated. Among others, these aspects are detailed in the remainder of this thesis.

3

Medium Access Control for Low-Power Operation

Application scenarios for (safety-critical) Wireless Sensor Networks often involve battery-powered nodes that should be operational for years without the necessity of exchanging the batteries. With today's hardware platforms drawing tens of milliamperes of current and battery capacity being limited to a few ampere hours, the need for energy management becomes apparent; without it a node would drain its batteries within a couple of days. This fundamental need for energy-efficient operation has drawn the attention to the radio, which is the component of a typical sensor node that consumes most energy.

Duty cycling the radio, that is, repeatedly switching it off for some time, is the only way to achieve the required two orders of magnitude reduction in energy consumption for extending lifetime from days to years. This duty cycling effectively reduces the available bandwidth on the radio channel, hence, limits the amount of data that can be communicated through the sensor network. Note that the reverse does not necessarily hold; applications constraining their payload do not automatically extend the lifetime of a sensor node because radios consume about as much energy when running idle as when transmitting or receiving data. Only by putting the radio into (deep) sleep, energy consumption reduces to zero (i.e., from mW to μW range), which is the task of the Medium Access Control (MAC) layer driving the radio hardware.

Since the introduction of the canonical S-MAC protocol [YHE02], a whole range of energy-efficient WSN MAC protocols have been devel-

oped (cf. Section 3.1). These MAC protocols all trade off performance (latency, throughput) for a reduction in energy, but differ in complexity and flexibility to adapt to traffic fluctuations, topology changes, and varying channel conditions. Simple protocols are often based on random access (CSMA), while more complex protocols organize channel access according to some predefined schedule (TDMA).

Developers of long-running applications must be careful in selecting the MAC protocol that suits their needs best, so that they can squeeze the most out of the limited hardware resources. As such it is essential to understand how MAC protocols operate in specific conditions (data rates, traffic patterns, interference levels, etc.). At the moment, however, there is no reference framework for doing so. Typical MAC protocols have been demonstrated to outperform S-MAC, but with different simulators, hardware platforms, and workloads, making it very difficult to assess their behavior in another context. Comparative studies like [HDL05] shed some light on the relative performance of a few protocols, but again only for a limited number of deployment scenarios. The latter aspect can be addressed by analytical models capturing relevant system parameters, as for example the analytical model for the energy consumption of the data-link layer by Zhong et al. in [ZRW04], which models the network traffic and the radio characteristics. However, they discuss very few and meanwhile outdated MAC protocols (ALOHA, CSMA) while advanced MAC protocols, as considered in this chapter, contain a number of internal parameters that must be taken into account too. For optimal performance, it is simply not sufficient to compare MAC protocols running with their standard settings. Instead the whole parameter range for all protocols have to be compared against each other.

This chapter addresses this exploration void and presents a model-based approach for benchmarking MAC protocols for the case of low data-rate applications where energy efficiency is needed the most. The analytical models of state-of-the-art MAC protocols are driven by a set of context parameters (e.g., radio characteristics, data rate and network topology) as well as internal MAC-protocol parameters (e.g., duty cycle, slot length, and number of slots). As the low data rates make complications like collisions and congestion rare events, the models can be kept rather simple in nature. This has three advantages. One, it makes the analytical approach scalable in the sense that analyzing various protocols is feasible. Second, it provides fairness for the comparison of the protocols, as with the rather simple parameters it is avoided to get lost in minor model details while loosing sight of the big picture. Three, the MAC protocols can be evaluated in a large number of different settings.

This chapter presents a few results from the design-space exploration, showing that reducing idle listening and overhearing, and managing

clock drift are keys to achieving energy efficiency for low data-rate applications. However, the main contribution of this chapter are the analytical models, which can be used to select the MAC protocol that favors an application's specific requirements and conditions. In particular, it allows identifying the most suitable MAC protocol for the safety-critical WSN protocol stack presented in Chapter 5.

The remainder of this chapter is structured as follows. Section 3.1 presents a brief overview of MAC protocols especially developed for WSNs. Section 3.2 introduces the modeling framework, followed by a few example models of state-of-the-art MAC protocols in Section 3.3. These models are analyzed in Section 3.4 for a set of delay-insensitive monitoring applications in typical network topologies, where energy efficiency is the prime consideration. Most MAC protocols were originally designed for being used with a byte-stream radio. It is discussed in Section 3.5 how the MAC protocols and the modeling framework can be adapted for being used with packet-based radios. Section 3.6 concludes this chapter.

3.1 Energy-Efficient MAC Protocols

The primary concern of WSN-specific Medium Access Control protocols is to switch the radio into sleep mode; otherwise energy would be wasted due to so-called *idle listening* by nodes waiting for potential incoming traffic. Other sources of overhead that should be avoided include *overhearing* of messages destined to other nodes, *unnecessary sending* if the receiver is not listening, *collisions* for example due to hidden terminals, and *protocol overhead* for medium reservation and clock synchronization. An additional complication for the low data-rate applications considered in this study is that amortizing overheads, for example by piggybacking protocol information on data messages, becomes rather difficult when applications only send out a message every few seconds, or even minutes.

All energy-efficient MAC protocols duty cycle the radio, at the expense of reducing bandwidth and increasing latency. The reduction in throughput is of little concern since commonly-used radios like the TI CC1000 offer ample bandwidth (76.8 kbps) compared to what most applications need ($\ll$ 1 kbps). The latency increase, on the other hand, is much more of a concern especially when targeting duty cycles of less than 1 %; in a multi-hop scenario a message may encounter a delay up to a complete sleep interval (1 s or more) on every transfer, resulting in long end-to-end latencies.

A whole range of WSN-specific MAC protocols has been developed, each with its own trade-off between latency, throughput and energy savings. In general, three classes of protocols are distinguished, depending

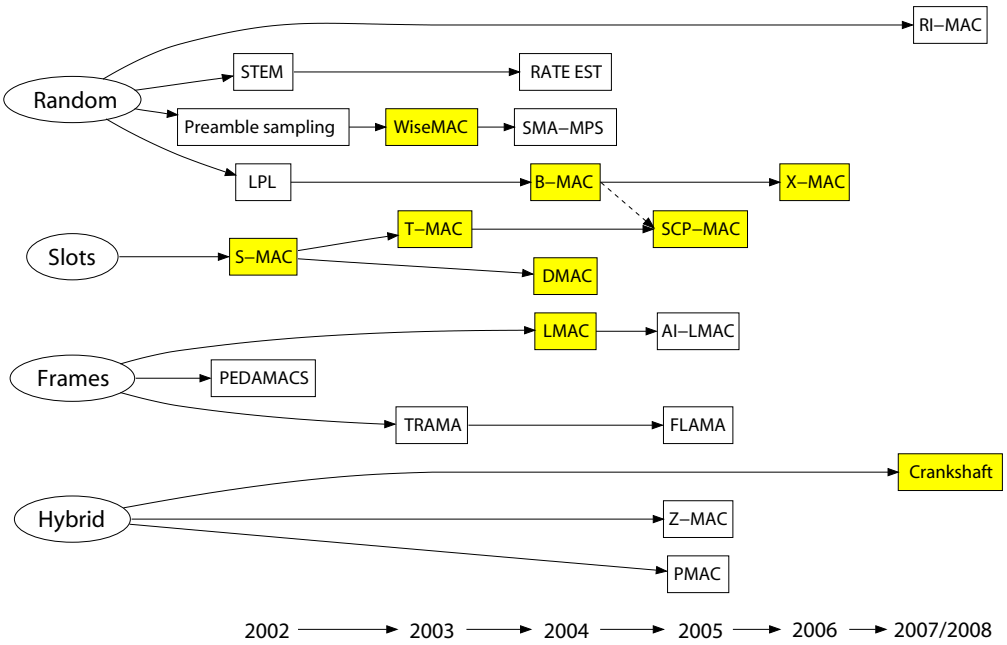


Fig. 11: Taxonomy of MAC protocols according to organization and historic development; the protocols analyzed in this study are highlighted.

on how strict access to the channel is organized. Figure 11 shows a taxonomy of MAC protocols along this classification of *random*, *slotted*, and *frame-based* (TDMA) access, as well as the historic development within each class.

The class of *random access* protocols puts no restrictions on when a sleep/active cycle is taking place. Neighbors do therefore not need to coordinate their cycles and consequently wake up independently of each other. This avoids the overheads and bookkeeping associated with running a time synchronization protocol, but leaves it up to the sending nodes to arrange a rendezvous with the intended receiver whenever it wakes up. An effective way is to stretch the length of the standard preamble to cover one complete sleep interval, which assures that the receiver (when polling the channel for activity) will detect a signal and eventually detect a start symbol, followed by the true message. This technique is known as *low-power listening* (LPL) [HC02] or *preamble sampling* [EH02], and was subsequently refined by the B-MAC protocol [PHC04], which added a user-controlled sleep interval, and by the WiseMAC protocol [EHD04], which tracks the phase offsets of neighbors' schedules allowing senders to transmit a message just in time with a short-length preamble saving energy and bandwidth. The X-MAC protocol [BYAH06] in turn enhances B-MAC by adapting it for packet-based radios sending out streams of packets instead of one long preamble.

A reverse approach to LPL was proposed with the RI-MAC protocol [SGJ08]. Instead of listening periodically, beacons are sent at a regular interval, indicating to be ready subsequently to receive a message. This avoids sending a long wake-up preambles (the sender has to listen for the receiver's beacon instead) and is therefore especially suited for high data-rates. Another approach to arrange a rendezvous is to use a second, low-power radio to send a wakeup signal, which will prompt the receiver to power up its primary radio to listen for the message that follows shortly. The idea of using wakeup radio is explored by STEM [STGS02] and RATE EST [MV04] in simulation. Since there is no hardware implementation in regular use, wake-up radios are not considered in the remainder of this chapter.

The class of *slotted access* protocols requires nodes to synchronize on a global notion of time, which is then organized as a sequence of slots. This organization allows nodes to collectively iterate through a sequence of active/sleep cycles. In the simplest case of the S-MAC protocol [YHE02], nodes spend a fixed amount of time in active and sleep mode, i.e., they wake up at the beginning of each slot and go back to sleep after a fixed-length interval. Within an active period, nodes follow the classic CSMA with collision avoidance (RTS/CTS signaling) approach to gain access to the channel. To account for variations in traffic, both in time and location, T-MAC [vDL03] introduces a simple timeout mechanism to adapt the length of the active period to the actual load. At the start of each slot, nodes listen for a short period (around 10 ms) to see if there is any communication to engage in, if not, they switch back to sleep mode. This allows saving a lot of energy over S-MAC running at a duty cycle that matches the load at the busiest node in the network. The SCP-MAC protocol [YSH06] manages to reduce the length of the active period to just 1-2 milliseconds by orchestrating senders to resolve contention before the receivers poll the channel (see Section 3.3.2). A down-side shared by all slotted protocols is that communication is grouped at the beginning of each slot, raising the chances on collisions, hence, limiting their dynamic range to low traffic rates only.

The class of *frame-based access* protocols groups slots into frames, and eliminates contention by precisely scheduling who is allowed to send in which slot. Computing these (periodic) schedules is rather difficult. Simple (distributed) policies lead to overprovisioning, as in the case of LMAC [HH04]; complicated policies taking actual traffic loads into consideration induce great complexity and relatively large memory footprints for maintaining neighbor state, making protocols like TRAMA [ROGLA03] hard to use in practice.

Lately, a number of hybrid protocols have been proposed that try to combine the best of all domains, basically, by putting a TDMA-overlay

structure on top of CSMA/CA. This combination employs the flexibility of random access with a (much) lower chance of collision. The P-MAC [ZRS05], Z-MAC [RWAM05], and Crankshaft [HL07] are example protocols from this hybrid category, with Crankshaft being used as the representative in this chapter.

3.2 Modeling Framework

Given the wide variety of MAC protocols it is important to understand their (relative) performance, such that the best protocol can be selected for a specific deployment. The focus of this study is on long running, low data-rate applications, which already corners the set of possible operational conditions that has to be considered when evaluating MAC protocols. Nevertheless, a thorough exploration should consider variation in workload (application traffic), radio and channel characteristics, and network topology (e.g., node density).

Taking an analytical, model-based approach allows for a fast evaluation of a number of MAC protocols in a rather large space of operational conditions. In particular this chapter presents models for the network topology, radio hardware, application traffic, and nine MAC protocols. The former three models capture the essential operating conditions that are varied when evaluating the MAC protocols in Section 3.4. These three models are presented in the following, whereas the MAC models are discussed in the next section.

3.2.1 Application Characteristics

Two kinds of applications are distinguished in this study, namely event-based and periodic reporting. In many monitoring type of applications, nodes simply send a status report on a regular basis to a central sink node for (offline) processing and storage. Example deployments include observing the nesting habits of Storm Petrels by monitoring the presence of birds in individual burrows [MPS+02], recording the change of light intensity at various heights in a Redwood tree [TPS+05], and observing the microclimate (temperature and humidity) in a potato field [LBV06]. The typical communication pattern that emerges from periodic reporting is a spanning tree with traffic flowing from the leaves to the data sink at the root (see Figure 12). Depending on the specific deployment data capturing may be synchronized (network-wide snapshots), and data may be aggregated at intermediate nodes. For simplicity unsynchronized data capturing without aggregation is considered, but extending the models to include these features has proven to be straightforward.

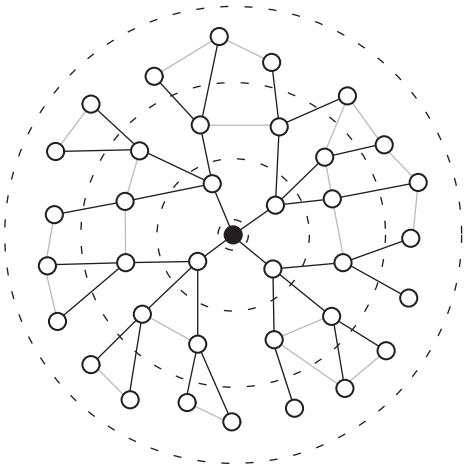


Fig. 12: Sample spanning tree with the sink at level 0 and a depth of 3.

The class of event-based applications shows a much more erratic communication pattern as network activity is triggered by some external event. For example, in the case of a surveillance system, the detection of an intruder prompts the forwarding of an alarm message along some route to the sink. Also, many monitoring applications can be optimized to report only significant changes (bird enters/leaves a burrow) instead of continuously streaming redundant data (the presence of the bird).

For both classes of applications the amount of energy consumed is of major interest, because that determines the lifetime of the network (i.e., the time until the first node runs out of energy), and hence the feasibility of the deployment. In the case of event-based applications, almost all energy is spent on keeping the network alive as the data rate is close to zero, assuming applications where events are rare. In the case of periodic reporting, additional energy is spend on forwarding data packets to the sink, with nodes close to the sink typically handling more traffic. A MAC model should take both classes into account and incorporate system-level traffic (for keeping the network alive) as well as application-level traffic.

The second performance metric of interest is latency. In many event-based scenarios (end-to-end) latency is bounded by application requirements, for example, an intrusion detection must be reported at the sink within a few seconds. Depending on the MAC protocol there can be a rather large difference between average latency and worst-case latency. For example with LMAC, a TDMA-style protocol in which each node owns a time slot, the average delay at each hop is half the frame length, but in the worst case the send slot is just missed and the message is delayed for (almost) a complete frame. Reporting worst-case latency, however, has little merit as in real life collisions and external interference

Parameter	Description	Value (Range)
P	Payload [byte]	32
F_S	Sampling frequency [#pkts/node/min]	0.01 – 10
C	Connectivity [#neighbors]	4 – 16
F_I	Aggregated input traffic	$\sum_{n \in I} F^n_{out}$
F_B	Aggregated background traffic	$\sum_{n \in B} F^n_{out}$
F_{out}	Output frequency	$F_I + F_S$

Tab. 4: Traffic model (for a node N) with typical parameter values; the topology information is encoded by means of sets I and B (cf. Figure 13), satisfying $C = |I| + |B|$. Note that all parameters are node specific, but that the indices (superscripts) have been omitted for clarity whenever possible (e.g., write I instead of I^N).

rule out any strict guarantees to begin with. Therefore the average latency of the MAC protocols is modeled. For periodic reporting on the other hand, latency is of minor concern because such data is quite often stored for offline processing (e.g., generate daily/weekly/monthly statistics) and even for online monitoring a delay in the order of a minute is sufficient for most applications.

As the focus of this study is on low data-rate applications, throughput is of no concern, other than that MAC protocols should not be driven into overload leading to collisions and queue overflows. This will be safe guarded by adding boundary constraints on the amount of traffic flowing through the network. In order to keep the analysis tractable, the impact of external interference, i.e., random packet loss is not considered. As a consequence, the MAC models do not need to consider retransmissions, which greatly simplifies the models.

3.2.2 Traffic Model

To keep a MAC model as simple as possible, the actual network topology is abstracted by detailing for every node what *input* traffic it is handling, and what *background* traffic is potentially bothering it being send out by its other neighbors. Note that background nodes may be peers at the same level of the tree, as well as nodes at levels below and above that of the node itself. The latter group includes the parent node forwarding a node's outgoing traffic (amongst others) further up in the tree. Figure 13 illustrates the local view of a MAC model, which is sufficient to express the energy consumption of a node using the particular traffic parameters from Table 4. The network lifetime can then be obtained by simply iterating over all nodes and recording the maximum energy consumption for the traffic parameters at each node.

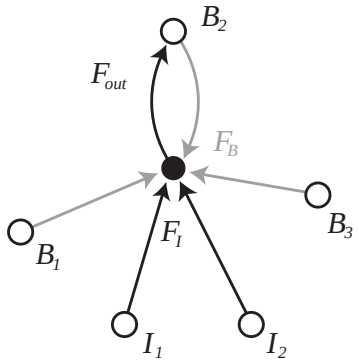


Fig. 13: Local view of the traffic model: black arrows indicate traffic flowing through, grey arrows indicate interference from background nodes.

The traffic model embeds the spanning tree of the application on top of a raw network topology by specifying for each node the set of input (child) nodes I and the set of overheard (background) nodes B. This allows for accurate modeling of specific, irregular deployment scenarios, as well as for modeling regular topologies like grids and the ring structure of Figure 12.

In the following a set of equations is derived that allows to calculate the induced traffic of the ring structure, which is being used in the MAC performance analysis in Section 3.4. In particular, a spanning tree is constructed in the network that is based on shortest-hop routing to the sink located in the center. Assuming a uniform node density on the plane and a unit disk graph communication model, there are $C+1$ nodes on the unit disk. Hence all nodes are in communication range with a fixed number of C neighbors. The nodes are grouped into rings according to their distance d (minimal hop count) to the sink ($d = 0$). The first ring contains C nodes, from which the node density is derived, and subsequently the number of nodes N_d in ring d:

$$N_d = \begin{cases} 1 & \text{if } d = 0 \\ Cd^2 - C(d-1)^2 = (2d-1)C & \text{otherwise.} \end{cases}$$

Knowing the number of nodes in each ring allows to compute the average number of input links of a node at level d as the ratio between N_{d+1} and N_d:

$$|I_d| = \begin{cases} 0 & \text{if } d = D \\ C & \text{if } d = 0 \\ N_{d+1}/N_d = (2d+1)/(2d-1) & \text{otherwise,} \end{cases}$$

where D denotes the maximum distance to the sink. Note that the number of input links is independent of the node density. Knowing the number

of input nodes and the nodes' basic sampling frequency F_S, the output frequency is computed as

$$F_{out}^d = \begin{cases} F_S & \text{if } d = D \\ F_I^d + F_S = |I_d|F_{out}^{d+1} + F_S & \text{otherwise.} \end{cases}$$

This iterative formula can be reduced to

$$F_{out}^d = F_S(D^2 - d^2 + 2d - 1)/(2d - 1), \tag{3.1}$$

which gives a closed formula for the input rate (using $F_I^d = F_{out}^d - F_S$):

$$F_I^d = \begin{cases} F_S D^2 C & \text{if } d = 0 \\ F_S(D^2 - d^2)/(2d - 1) & \text{otherwise.} \end{cases} \tag{3.2}$$

To arrive at the aggregate background traffic it is assumed that the B nodes in Figure 13 on average generate the same load (F_{out}^d) as the node itself:

$$F_B^d = |B_d|F_{out}^d = (C - |I_d|)F_{out}^d. \tag{3.3}$$

With equations (3.1) - (3.3) determining the node that consumes most energy requires D evaluations (evaluate one node per ring) of a MAC model, whereas brute-force (node by node) processing would involve CD^2 evaluations. When considering real-world, irregular topologies similar speed-ups can be achieved by focusing on a few bottleneck nodes through specifying their individual parameters (F_I, F_{out}, F_B, C, etc.).

3.2.3 Radio Model

Besides the traffic parameters, the evaluation of a MAC model requires input about the radio hardware that is, or will be, used in the specific deployment scenario. Since the objective of this study is to compare the relative performance of various MAC protocols, many details can be left out. Instead of computing the absolute energy-consumption level, the effective duty cycle is calculated. That is the fraction of time the radio is switched on; regardless whether transceiving, idle listening or powering up. This allows to omit exact energy consumption numbers and only focus on the timing aspects. As such a radio can be modeled with just three parameters: the time needed to power it up (i.e., to transit from sleep into active mode), its data rate, and the time needed to do a carrier sense (including power up).

An important issue with low data-rate applications is that clock drift becomes an issue for advanced MAC protocols that arrange nodes to wake up at precise moments in time to minimize energy consumption; without any data traffic the clocks of different nodes may drift apart and needs to

Par.	Description	TI CC1000	TI CC2420	RFM TR1001
Type	Byte-stream vs. packet based	Byte	Packet	Byte
R	Data rate [kbyte/s]	2.40	31.25	57.50
T_{on}	Turn radio on into RX or TX [ms]	2.10	2.40	0.5
T_{cs}	Time for carrier sense [ms] (T_{on} included)	2.45	2.60	0.53
θ	Frequency tolerance [ppm]	30	30	30
L_{pbl}	Minimal preamble length [byte]	6	4	2.5

Tab. 5: Radio model with typical parameter values for the radios used in Section 3.4 and 3.5.

be accounted for. Therefore, the clock-drift parameter θ is included that captures the precision of the (external) quartz crystal that determines the timing of the underlying (radio) hardware. An overview of the radios and their parameters used in this study are depicted in Table 5. It should be noted that a byte-stream radio is assumed for the MAC models, since most protocols were originally designed for such radios. It is discussed in detail in Section 3.5, how the models are applicable to packet-based radios.

3.3 MAC Models

There are many parameters influencing the performance of a MAC protocol. First, there are the (external) radio and network parameters as introduced in the previous section. Second, there are the internal MAC parameters, such as duty cycle, number of slots, and polling time. This section shows how the energy efficiency and latency of the protocols is modeled based on both the internal and external parameters.

Altogether nine different protocols are analyzed for this study, taking a selection of the whole MAC design space as indicated in Figure 11. An overview of these nine protocols and their internal parameters is provided in Table 6 and 7 for reference. Discussing each protocol in detail would take up too much space. Therefore the most sophisticated protocol of each category is presented only: LMAC (scheduled), SCP-MAC (slotted), WiseMAC (random access with wake-up prediction), and Crankshaft (hybrid). The remaining five protocols, namely S-MAC (slotted), T-MAC (slotted with timeout), D-MAC (slotted with convergecast), B-MAC (random access), and X-MAC (packet-based random access), are briefly discussed in Appendix A on page 147.

Parameter	Description	Value (Range/Set)
B-MAC		
T_w	Sample/polling period [s]	[0.02, 2]
Crankshaft		
L_{data}^{max}	Maximum data length [byte]	32
T_{sync}	Time between sync packets [s]	[12, 60]
N_u	Number of unicast slots per frame	[4, 32]
N_b	Number of broadcast slots per frame	2
D-MAC		
N_{sleep}	Number of sleep slots	[6, 100]
T_{sync}	Time between sync packets [s]	[60, 600]
LMAC		
L_{data}^{max}	Maximum data length [byte]	{32,64,128,256}
N_{slots}	Number of slots per frame	32
S-MAC		
DC	Duty Cycle [%]	[0.1, 10]
$T_{discover}$	Discovery interval [s]	360
T_{active}	Active phase/slot [s]	[0.02, 0.1]
SCP-MAC		
T_w	Sample/polling period [s]	[0.02, 2]
T_{sync}	Time between sync packets [s]	[12, 60]
T-MAC		
T_{slot}	Duration of a single slot [s]	[0.15, 10]
T_{sync}	Time between SYNC packets [s]	100
$T_{discover}$	Discovery interval [s]	360
WiseMAC		
T_w	Sample/polling period [s]	[0.02, 2]
X-MAC		
T_w	Sample/polling period [s]	[0.02, 2]
T_{al}	Acknowledgement listen period [ms]	0.95

Tab. 6: Internal (configurable) protocol parameters.

Protocol	Control Packet	Size [byte]	CW [#slots]
B-MAC	L_{hdr}	9	15
	L_{ack}	$9 + L_{pbl}$	
Crankshaft	L_{hdr}	11	15
	L_{ack} (Sync)	$9 + L_{pbl}$	
D-MAC	L_{hdr}	10	15
LMAC	L_{hdr}	$7 + L_{pbl}$	–
S-MAC	L_{ctrl}	$8 + L_{pbl}$	15
SCP-MAC	L_{hdr} (Sync)	10	7 + 8
	L_{ack}	$8 + L_{pbl}$	
T-MAC	L_{ctrl}	$8 + L_{pbl}$	15
WiseMAC	L_{hdr}	7	15
	L_{ack} (Sync)	$9 + L_{pbl}$	
X-MAC	L_{hdr}, L_{ack}	$9 + L_{pbl}$	15
	L_{ps}	$5 + L_{pbl}$	

Tab. 7: Implementation-specific (fixed) protocol settings for headers and contention windows (CW), the latter ones having a slot time of $T_{Slot}^{CW} = 0.62\,ms$.

Given that for long-running applications most of the time (energy) will be spent in the operational phase of a MAC protocol, the models ignore any initialization procedures, which might be rather complex as in the case of setting up a TDMA schedule. Another important simplification follows from the assumption that messages do not get distorted (due to interference) or lost (due to collisions), so retransmissions are not modeled. Collisions are unlikely for low data-rate traffic, but boundary conditions are included to safe guard against the improper selection of protocol parameters. For example, a MAC protocol that samples the radio channel every second cannot possibly receive two packets per second.

In the following discussions of the four advanced MAC protocols (LMAC, SCP-MAC, WiseMAC, and Crankshaft) first provides a short description of the protocol, then discuss the synchronization (clock drift) aspects, and finally provide equations for the average h-hop latency and energy efficiency (duty cycle).

3.3.1 LMAC

The Lightweight MAC [HH04] protocol features a self-organizing TDMA scheme that organizes time into frames containing N_{slots} slots each (cf. Fig-

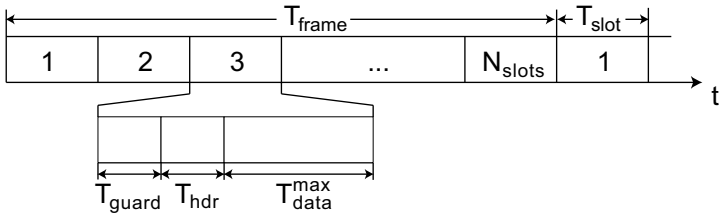


Fig. 14: The frame structure of LMAC.

ure 14). Every node owns one slot in which it sends out a header (to mark its occupancy), possibly followed by a data payload either addressed to a specific recipient (unicast) or to all nodes in range (broadcast). Consequently, a node must listen (i.e., perform a carrier sense) in all slots other than its own to check for incoming data. To allow for collision-free transmissions and spatial re-use of slots, a header includes a list of all occupied slots in the owner's one-hop neighborhood; after merging the occupancy information of its neighbors, a new node joining the network can select a free transmission slot within its two-hop neighborhood. This distributed, interference free slot selection mechanism obviates the need for explicit acknowledgement messages, which are left out from the LMAC protocol to save energy; the recovery from external interference is left to the upper layers in the protocol stack.

Synchronization. Synchronization is performed with every header that is sent, i.e., in every occupied slot. In the worst case a node hears one header per frame, hence, a sender and receiver can drift at most $2\theta T_{frame}$ apart (one running ahead, the other running behind). For efficiency the (receiving) nodes perform only a carrier sense, so the slot owner has to guard for the maximum clock drift by sending out a stretched preamble. Since it is unknown who is running ahead, the guard time must be twice the maximum drift:

$$T_{guard} = 4\theta T_{frame}, \quad \text{where } T_{frame} = N_{slots} \cdot T_{slot}. \tag{3.4}$$

Depending on the length of the slot T_{slot}, only a certain amount of payload can be transmitted at once. Therefore, if the slot should fit a payload of L_{data}^{max}, the length of the slot results in

$$T_{slot} = T_{guard} + T_{hdr} + L_{data}^{max}/R. \tag{3.5}$$

Latency. After receiving a message, a node must wait for its own slot for forwarding. The message is therefore sent in one of the next $N_{slots} - 1$ slots, which results in an average latency of $T_{hd} = (N_{slots} - 1) \cdot T_{slot}/2$. This is based on the assumption that no queues occur at any given node due to the low data rate of the application. For the first hop however, the message could be triggered right after the beginning of the owned slot

and hence the message is delayed for $T_{init} = (T_{frame} + T_{slot})/2$ on average. Finally, for the last hop only part of the transmission slot is occupied when the payload is shorter than L_{data}^{max}. For a message that needs to be forwarded h hops, this results in an average latency of

$$\begin{aligned} L(h) &= T_{init} + (h-1) \cdot T_{hd} - (L_{data}^{max} - P)/R \\ &= (h \cdot T_{frame} - (h-2) \cdot T_{slot})/2 - (L_{data}^{max} - P)/R. \end{aligned} \tag{3.6}$$

Energy-Efficiency. The energy efficiency is assessed considering the different sources individually. LMAC requires performing a carrier sense (E_{cs}) in every slot but the owned one. In every frame, C neighbors are sending a (guarded) header that is overheard (E_{hdr}), after which the radio can be switched off in most cases; only the incoming traffic F_I for the node itself is received (E_{rx}). Energy is spent also by transmitting (E_{tx}) a header in every frame (that needs to be guarded for potential clock drift) and the payload if there is data to send.

$$\begin{aligned} E_{cs} &= (N_{slots} - 1) \cdot T_{cs}/T_{frame} \\ E_{hdr} &= C \cdot (T_{guard}/2 + T_{hdr})/T_{frame} \\ E_{rx} &= F_I \cdot P/R \\ E_{tx} &= (T_{on} + T_{guard} + T_{hdr})/T_{frame} + F_{out} \cdot P/R \\ E &= E_{cs} + E_{hdr} + E_{rx} + E_{tx} \end{aligned} \tag{3.7}$$

Parameter Constraints. Every node has its own transmission slot, and hence, the bottleneck is at the nodes having the most packets to send, i.e., the bottleneck nodes are the ones next to the sink. In order to avoid queues, the bottleneck bandwidth is set to 50 %, i.e., having a packet in every second (owned) slot only:

$$F_{out}^{1} \cdot T_{frame} < 1/2. \tag{3.8}$$

3.3.2 SCP-MAC

The scheduled-channel-polling MAC [YSH06] protocol combines low-power listening (LPL) with a global synchronized channel access, especially designed for low duty-cycle operation. All nodes in the network wake up at a regular interval T_w and perform a synchronized carrier sense. A sender node has to contend for the channel *before* the scheduled wake-up, so the receivers do not waste energy on listening to a complete contention window. The synchronized channel poll must be guarded to account for possible clock drift, which results in a prolonged wake-up preamble (tone).

To limit the length of the senders' contention window (T_{cw1}), SCP-MAC employs a second contention window after the wake-up time to

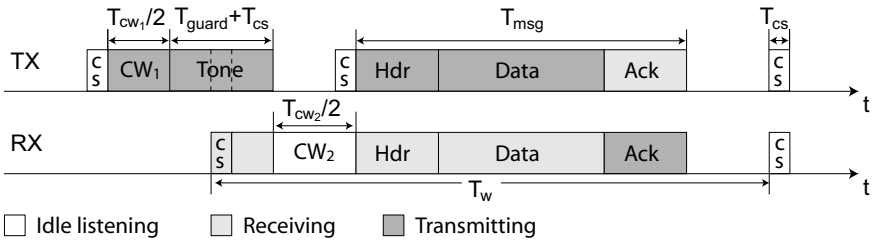


Fig. 15: The channel access policy of SCP-MAC.

handle the (increased) chance of multiple senders picking the same slot initially. This two-stage contention resolution policy allows for two short contention windows instead of a single long one. After this second contention window (T_{cw2}), the packet is sent and acknowledged[1].

Synchronization. SCP-MAC requires that at least one message is sent every T_{sync} in order to keep the neighboring nodes synchronized, which results in a guard time of $T_{guard} = 4\theta T_{sync}$. If the data rate is too low, additional synchronization messages have to be sent:

$$F_{sync} = \begin{cases} 0 & \text{if } F_{out} > 1/T_{sync} \\ 1/T_{sync} & \text{otherwise.} \end{cases} \tag{3.9}$$

Latency. A message is generated somewhere during the wake-up interval before the first contention window, which results in an average delay of $T_w/2$ at the first hop. For every additional hop, the packet will be delayed for another T_w. At the last hop, the time required for the packet transfer sequence needs to be taken into account.

$$L(h) \quad = \quad T_w/2 + (h-1) \cdot T_w + T_{cw1} + T_{guard} + T_{cs} + T_{cw2}/2 + T_{msg}, \tag{3.10}$$

where $T_{msg} = T_{hdr} + P/R + T_{ack}$ is the time required for sending the packet header, the payload and the acknowledgement.

Energy-Efficiency. The energy consumption of SCP-MAC is calculated by adding up its sources. The nodes are required to perform a (synchronized) carrier sense E_{cs} in every slot to check for a potential message. If a message is sent (E_{tx}), the node has to guard for potential clock drift, has to perform a carrier sense in the second contention window, and has to transfer the message as illustrated in Figure 15. A receiving node (E_{rx}) on the other hand will listen for half the guard time on average, half of the second contention window, and the message. For the messages that

[1]SCP-MAC offers several options, namely an RTS/CTS handshake with acknowledgement, an acknowledged, and an unacknowledged data transfer. Considering the low data rate and the short payload size in sensor networks, the handshake is a big overhead. Therefore the acknowledged service is modeled.

are overheard (E_{ovr}), i.e., sent to another node, the radio is switched off after receiving the header. The synchronization messages are handled the same way as data messages. However, only a header without an acknowledgement is broadcast (E_{stx}) and received (E_{srx}).

$$\begin{aligned} E_{cs} &= T_{cs}/T_w \\ E_{tx} &= F_{out} \cdot (T_{cw1}/2 + T_{guard} + T_{cs} + T_{msg}) \\ E_{rx} &= F_I \cdot (T_{guard}/2 + T_{cw2}/2 + T_{msg}) \\ E_{ovr} &= F_B \cdot (T_{guard}/2 + T_{cw2}/2 + T_{hdr}) \\ E_{stx} &= F_{sync} \cdot (T_{cw1}/2 + T_{guard} + T_{cs} + T_{hdr}) \\ E_{srx} &= C \cdot F_{sync} \cdot (T_{guard}/2 + T_{cw2}/2 + T_{hdr}) \\ E &= E_{cs} + E_{tx} + E_{rx} + E_{ovr} + E_{stx} + E_{srx} \end{aligned} \tag{3.11}$$

In contrast to LMAC, the length of the guard time can be influenced through a protocol parameter (T_{sync}). Sending a synchronization message entails overhead (i.e., adds E_{stx} and E_{srx}), but reduces the length of the guard interval (i.e., decreases E_{tx}, E_{rx}, and E_{ovr}). When the data rate of the application (F_s) is known, the optimum synchronization interval can be determined by taking the derivative of Equation (3.11) with respect to T_{sync}, and setting that to zero. For the evaluation, however, an exhaustive search over the whole parameter space is performed (cf. Table 6). This ensures that the most efficient operation point is selected.

Parameter Constraints. The bottleneck is at the sink node, which has to receive all (data & sync) messages injected into the network. In order to avoid hidden terminal collisions at the sink, there should only be one message every fourth slot. Note that this is a tighter bound (1/4) on the maximum bandwidth than for LMAC (1/2), which rules out hidden terminals by design (i.e., ensures conflict-free transmissions in a two-hop neighborhood). An additional constraint on the parameter settings is that a complete message sequence must fit into one slot.

$$\begin{aligned} (F_I^0 + |I^0| \cdot F_{sync}) \cdot T_w &< 1/4 \\ T_{cw1} + T_{guard} + T_{cw2} + T_{msg} &< T_w \end{aligned} \tag{3.12}$$

3.3.3 WiseMAC

The Wireless Sensor MAC [EHD04] is, like SCP-MAC, a refinement to the periodic carrier sense of LPL. With WiseMAC, the nodes wake up independently from each other at a periodic interval T_w. Instead of sending a long preamble, WiseMAC maintains a table with the neighboring nodes' poll schedule (updated individually with every packet that is sent), which allows sending a short wake-up preambles only. The clock drift is compensated by dynamically adapting the length of the wake-up preamble

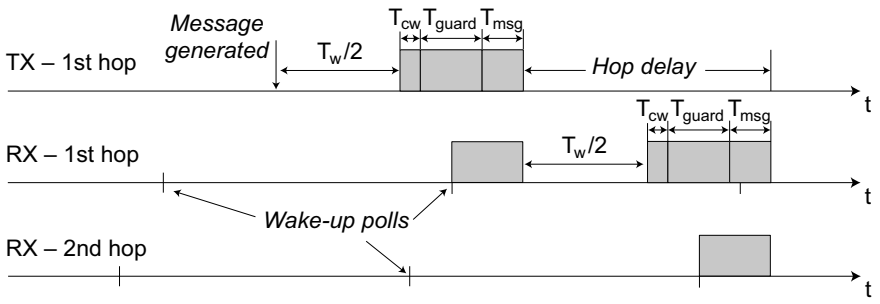


Fig. 16: WiseMAC latency estimation: the sending node has to wait for $T_w/2$ on average before starting to transmit the preamble and message.

according to the maximal possible clock drift since the last message exchange. If no information about a neighbor's poll schedule is available, WiseMAC falls back to the long wake-up preamble of LPL with length T_w. However, instead of just sending the long wake-up preamble, WiseMAC is sending consecutive data packets. Therefore, all neighboring nodes, except the intended receiver, will only receive a truncated first packet plus the header of a second one. The intended receiver must wait for the complete sequence, before it can acknowledge the data.

Synchronization. WiseMAC updates the polling schedule of the neighbor with every received acknowledgement. In particular no special synchronization messages are required to be sent. Unlike SCP-MAC, the guard time that compensates the clock drift is adapted dynamically: from the moment the last message exchange took place, the guard time increases up to the long wake-up preamble of LPL.

$$T_{guard} = \max(4\theta/F_{out}, T_w) \tag{3.13}$$

Latency. WiseMAC determines the starting point of the preamble based on the estimated wake-up time of the receiving node, subtracting half the dynamically adapted guard time and the contention window[2]. On average the sending node has to wait for $T_w/2$, before starting to transmit the the wake-up preamble and message as illustrated in Figure 16.

$$L(h) = h \cdot (T_w/2 + T_{cw} + T_{guard} + T_{msg}), \quad \text{where } T_{msg} = T_{hdr} + P/R + T_{ack}. \tag{3.14}$$

Energy-Efficiency. The energy consumption can best be approximated by analyzing its sources individually. There is the idle listening (E_{cs}), which requires switching on the radio every wake-up interval in order to

[2]The contention window, also referred to as medium reservation preamble, is included to prevent multiple senders with the same guard time from transmitting at the same time.

perform a carrier sense. The energy consumption for sending a message (E_{tx}) depends on the length of the preamble, consisting of the contention window and the guard time, and the time to transfer the message and the acknowledgement. A receiving node will on average listen to the second half of the guard time and take part in the message transfer sequence. Not all messages transmitted by background nodes are overheard; only those in progress when a node polls the channel are observed. The probability of overhearing a message is thus related to the length of an actual message sequence in relation to the length of the wakeup interval: $p_{ovr} = (T_{cw}/2 + T_{guard} + T_{msg})/T_w$. Furthermore, WiseMAC sends consecutive data packets instead of a long preamble. So only a part of the first, and the header of a second data packet will be overheard.

$$
\begin{aligned}
E_{cs} &= T_{cs}/T_w \\
E_{tx} &= F_{out} \cdot (T_{cs} + T_{cw}/2 + T_{guard} + T_{msg}) \\
E_{rx} &= F_I \cdot (T_{guard}/2 + T_{msg}) \\
E_{ovr} &= \begin{cases} F_B p_{ovr}((T_{hdr} + P/R)/2 + T_{hdr}) & \text{if } T_{cw}/2 + T_{guard} > T_{hdr} + P/R \\ F_B p_{ovr}((T_{cw}/2 + T_{guard})/2 + T_{hdr}) & \text{otherwise} \end{cases} \\
E &= E_{cs} + E_{rx} + E_{tx} + E_{ovr} + E_{sync}
\end{aligned} \tag{3.15}
$$

Parameter Constraints. There is a bottleneck at the sink node. However, the randomly distributed wake-up slots of WiseMAC provide a natural way of increasing the available bandwidth, i.e., the sink does not have to share its slot with neighboring nodes as with synchronized protocols. Therefore the traffic at the sink can be higher with WiseMAC than with SCP-MAC, i.e., having a message in every second wake-up slot. A second constraint needs to ensure that the message sequence and the contention window fit into one slot.

$$(F_I^0 + |I^0| \cdot F_{sync}^1) \cdot T_w < 1/2 \tag{3.16}$$

$$T_{cw} + T_{msg} < T_w \tag{3.17}$$

3.3.4 Crankshaft

The Crankshaft [HL07] protocol is a hybrid MAC that combines scheduled with contention-based access. Time is divided into frames consisting of N_b broadcast and N_u unicast slots. Every node is required to listen to *all* broadcast slots and to *one* of the unicast slots, which is assigned based on its MAC address modulo N_u. A node that needs to transmit a packet to a particular node has to wait for this node's unicast slot and content for it using the Sift [JBT06] contention resolution scheme. The contention resolution is required since several nodes might want to send a packet in a particular slot (but not necessarily to the same node). Even though the

data traffic is divided into several unicast slots and contention resolution is performed, collisions might still occur or data packets are not received due to other interference. Therefore, data packets are acknowledged. Furthermore, Crankshaft explicitly provides a special mode for the usually line-powered sink. Since energy consumption is of no concern, the sink listens into all unicast slots. A message to the sink can therefore be sent in any unicast slot, which increases the receiving bandwidth of the sink substantially. Note that the access control of Crankshaft basically reduces to that of SCP-MAC when operating with broadcast slots only ($N_u = 0$).

Synchronization. The Crankshaft protocol requires every node to send a synchronization message every T_{sync} in one of the broadcast slots to keep the network synchronized. It should be noted that the ordinary data transfer cannot be used to keep the network synchronized, as these messages are sent in unicast slots and, hence, not received by all neighboring nodes. In order to reduce the energy consumption for idle listening, the nodes do only perform a carrier sense in their slots leaving the transmitting node to guard for potential clock drift.

$$T_{guard} = 4\theta T_{sync} \tag{3.18}$$

As with LMAC, the slot length T_{slot} limits the payload that can be transmitted. If the slot should fit a payload of L_{data}^{max}, the length of the slot results in

$$T_{slot} = T_{cw} + T_{guard} + T_{hdr} + L_{data}^{max}/R + T_{ack}, \tag{3.19}$$

and therefore the frame length in

$$T_{frame} = (N_b + N_u) \cdot T_{slot}. \tag{3.20}$$

Latency. The latency for Crankshaft is very similar in nature to that of LMAC. There is an initial delay $T_{init} = (T_{frame}+T_{slot})/2$, followed by waiting at each hop until the next forwarder's slot shows up taking $T_{hd} = T_{frame}/2$ on average. For the final transmission to the sink, the node has only to wait for the next unicast slots, i.e., has to wait for $N_b/N_u + 1$ slots on average before the message can be sent.

$$\begin{aligned} L(h) &= T_{init} + (h-2) \cdot T_{hd} + (N_b/N_u + 1) \cdot T_{slot} - (L_{data}^{max} - L_{data})/R \\ &= (h-1) \cdot T_{frame}/2 + (N_b/N_u + 3/2) \cdot T_{slot} - (L_{data}^{max} - L_{data})/R \end{aligned} \tag{3.21}$$

Energy Efficiency. Crankshaft requires performing a carrier sense (E_{cs}) in one unicast slot and all broadcast slots per frame. A node sending a message (E_{tx}) has to contend for the channel and guard the clock drift before starting the actual message transfer sequence. If a message is received (E_{rx}), the node will on average overhear half of the guard time only. Sending (E_{stx}) and receiving (E_{srx}) synchronization messages is very similar to

ordinary data messages except that they only consist of a header. Data messages are only overheard if a neighboring node has the same unicast slot assigned. Since slots are assigned based on MAC addresses, which is assumed to be randomly distributed, the number of nodes sharing the same slot follow a binomial distribution. That is, the probability that n out of $|B|$ nodes share a particular node's unicast slot is

$$P_r(X = n) = \binom{|B|}{n} p^n (1-p)^{|B|-n}, \quad \text{where } p = 1/N_u. \tag{3.22}$$

Calculating the number of neighbors that are overheard, does not assume the worst-case distribution of having all $|B|$ neighbors having the same slot assigned. Instead the number of overheard neighbors is estimated N_{ovr} according to the paradigm $P_r(X \leq N_{ovr}) < 0.9$, i.e., the number of slot collisions (N_{ovr}) is in 90 % of the cases less than the expected value.

$$\begin{aligned}
E_{cs} &= (N_b + 1) \cdot T_{cs}/T_{frame} \\
E_{rx} &= F_I \cdot (T_{guard}/2 + T_{msg}), \quad \text{where } T_{msg} = T_{hdr} + P/R + T_{ack} \\
E_{ovr} &= N_{ovr} \cdot F_B/|B| \cdot (T_{guard}/2 + T_{hdr}) \\
E_{tx} &= F_{out} \cdot (T_{cs} + T_{cw}/2 + T_{guard} + T_{msg}) \\
E_{srx} &= C \cdot (T_{guard}/2 + T_{hdr})/T_{sync} \\
E_{stx} &= (T_{cw}/2 + T_{guard} + T_{hdr})/T_{sync} \\
E &= E_{cs} + E_{rx} + E_{ovr} + E_{tx} + E_{srx} + E_{stx}
\end{aligned} \tag{3.23}$$

Parameter Constraints. Due to the special sink mode of Crankshaft, the bottleneck is only at the sink node if the number of incoming links exceeds the number of unicast slots (3.24). Otherwise, the bottleneck is at the nodes next to the sink (3.25). Then, there must be enough broadcast slots for sending the synchronization messages (3.26). Similar to LMAC, a message should only be received in every second slot.

$$F_I^0/N_u \cdot T_{frame} < 1/2 \tag{3.24}$$

$$(F_I^1 + N_{ovr}^1 \cdot F_B^1/|B^1|) \cdot T_{frame} < 1/2 \tag{3.25}$$

$$C/N_b \cdot F_{sync} \cdot T_{frame} < 1/2 \tag{3.26}$$

3.4 Analysis

The previous section modeled the main characteristics of four advanced energy-efficient MAC protocols (LMAC, SCP-MAC, WiseMAC, and Crankshaft); an additional five models of well-known, protocols (S-MAC, T-MAC, D-MAC, B-MAC, and X-MAC) are provided in Appendix A on

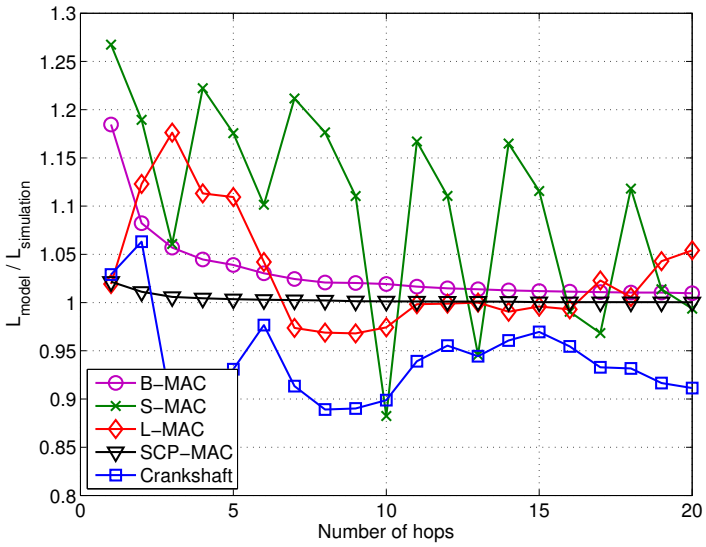
page 147. This section analyzes the fundamental latency-efficiency trade-off of the individual protocols, as well as how they compare to each other. First, however, the models are validated, demonstrating the soundness of the MAC models when compared to detailed, and time-consuming, simulations. Furthermore the tuning process is detailed for obtaining the best performance of a MAC protocol given a set of external conditions and constraints.

3.4.1 Validation

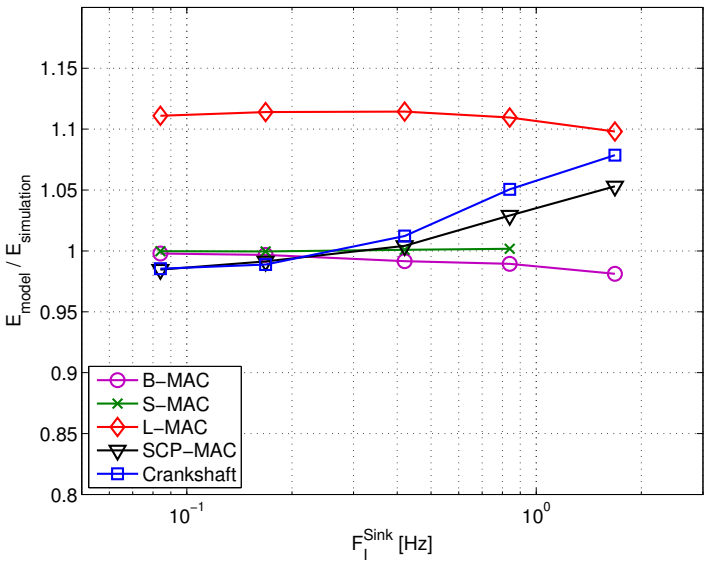
The art of modeling is to "make everything as simple as possible, but not simpler" (Albert Einstein). The MAC models are rather simple and abstract away many implementation details. The concern is then if the models are accurate enough to capture the essential performance characteristics. To answer this question the latency and energy efficiency of the models is compared with simulation, for those protocols (namely B-MAC, S-MAC, LMAC, SCP-MAC, and Crankshaft) where detailed simulation code was available [HDL05, HL07]. The simulator, an OMNeT++ based discrete-event simulator, uses an SNR-based reception model to determine which packets are dropped due to contention and external interference. This reception model in combination with the MAC protocols was proven to be rather accurate; most simulation results are within 5 % of actual delivery ratios and energy consumption numbers obtained on a 24-node testbed when the measured RSSI values are made available to the SNR-based channel model [HL09].

Figure 17 shows the ratio of the MAC performance models over the outcome of the corresponding simulations assuming no external interference. Each simulation result is the outcome over a series of 10 runs with a different random seed to average out the non-deterministic effects introduced by channel access policies, collisions, and the like. Figure 17(a) shows that the end-to-end latency as predicted by the models is usually within 10 % of the value determined by simulation. The ratio for S-MAC is more erratic, and can be attributed to the model fixing (rounding down) the number of hops that can be made in one active period (cf. Equation (A.3)); in reality the number of hops depends on the choice of waiting times in the contention windows introducing a certain amount of variability that is not accounted for.

Figure 17(b) shows that efficiency (duty cycle) as predicted by the models is generally within a 10 % margin when compared to simulation results, with very good accuracies for low data rates. LMAC is the exception with the model always being too pessimistic by roughly 12 % of the true efficiency. Unfortunately, it was not possible to identify the source of this discrepancy, nor can it simply be corrected as the overshoot de-



(a) Latency



(b) Energy efficiency

Fig. 17: Comparing model and simulation results (CC1000 radio, ring topology, $C = 8$, $D = 4$).

pends on the network topology. Nevertheless with the majority of errors within 10 % the MAC models strike a good balance between complexity and accuracy.

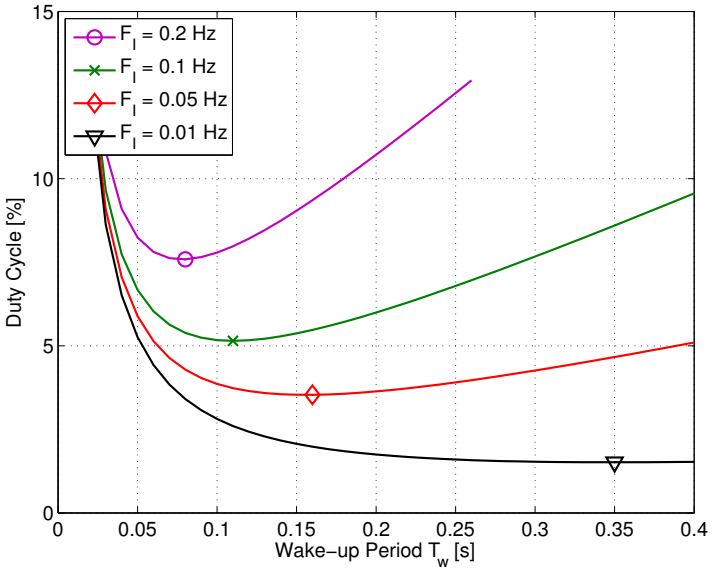
3.4.2 Protocol Optimization

The exact behavior of the MAC models depends on the settings of some protocol-specific parameters as listed in Table 6. For example, the performance of LMAC strongly depends on the number of slots in a frame (N_{slots}) and the length of an individual slot ($L_{hdr} + L_{data}^{max}$). The value ranges for these internal protocol parameters are derived (centered around) the settings provided in the original protocol descriptions and an earlier simulation-based comparison study [HDL05]. The notable exception is the setting of the synchronization interval (T_{sync}) of SCP-MAC, which is varied between 12 and 60 seconds, because the advocated range of 300-3600 seconds in [YSH06] yields worse results in the evaluation. This discrepancy is attributed to a different view on the effectiveness of synchronization messages.[3] Table 7 provides implementation details regarding the length of the protocol headers, control messages, and contention window (if applicable) of each protocol.

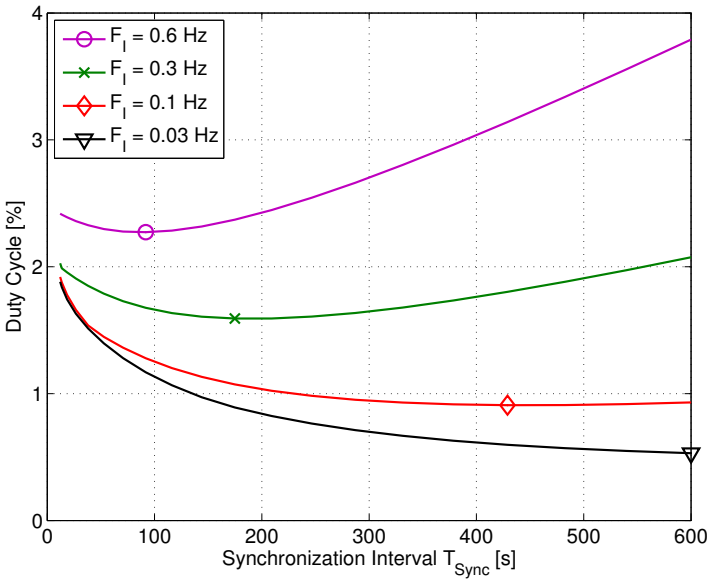
Since the optimal settings of the internal protocol parameters depend on the external conditions (e.g., traffic rate, network density, radio characteristics), comparing MAC models is not completely straightforward. In the analysis presented in this section a simple, exhaustive method is adopted that, given a set of external conditions, iterates over all combinations of parameters considered for a given MAC protocol (cf. Table 6). For all these settings the performance metrics of interest is computed first, e.g., latency and energy efficiency. Subsequently all the settings are pruned that are inferior to so-called Pareto points [Deb01], which offer in this case either lower latency for the same energy efficiency, or higher energy efficiency for the same latency, or provide both lower latency and higher energy efficiency. The end result of this multi-objective optimization process is that the parameter setting are found that offer the best trade-offs in the latency, energy-efficiency and data-rate space under consideration.

As an example, consider the plots in Figure 18 that illustrate the optimization process for B-MAC and Crankshaft. B-MAC allows trading off a more frequent channel polling (controlled through T_w) for a shortened

[3] It is argued that a node should receive a synchronization message from *all* neighbors within one period, while Ye et al. state it is sufficient to receive one message from *any* neighbor; this relaxed constraint, however, is not enough for synchronizing nodes in sparse network topologies like linear chains. Furthermore this ensures fairness for the comparison, since SCP-MAC is now following the same synchronization policy than the other protocols that maintain a global structure.



(a) B-MAC



(b) Crankshaft

Fig. 18: Optimization of MAC parameters with respect to data load (F_I). The markers indicate the optimal operating points of the protocols for different data loads.

wake-up preamble. As shown in Figure 18(a), both a very short and a very long sampling period will result in a high duty cycle (i.e., low efficiency), but their causes differ. While a short polling period will increase the energy consumption for idle listening, a long one will increase the energy consumption for transceiving the wake-up preamble. This results in an optimal sampling time T_w, being dependent on the data load in the network. Furthermore, the polling period cannot be chosen arbitrarily large, as indicated by the topmost line ($F_I = 0.2\,\text{Hz}$) discontinued at $T_w = 260\,\text{ms}$; the polling period limits the maximum amount of traffic in the network as denoted in Equation (A.17).

Crankshaft requires sending special messages to keep the network synchronized. As illustrated in Figure 18(b), the interval T_{sync} of these messages can be optimized with respect to the data load. For high data-rates, the synchronization interval should be chosen rather short, which results in a shortened guard time for all messages sent (cf. Equation (3.18)). For low data rates on the other hand, it does not pay off to send synchronization messages at a high rate, since the potential savings for the shortened guard time are minimized. Crankshaft does not only allow to parameterize the synchronization interval, but also to adapt the number of unicast slots N_u being used. Hence, for minimizing the duty cycle for a given data load, both parameters N_u and T_{sync} need to be considered, which results in a two-dimensional parameter optimization.

3.4.3 Data Load vs. Energy Consumption

The first experiment studies the impact of the data load on the energy consumption of the nine modeled MAC protocols. The data load as it arrives at the sink is a function of the number of nodes in the network and the sampling rate. In this experiment the network size (and topology) is kept fixed and the rate F_S at which messages are injected into the network is varied. For ease of understanding though, the aggregate rate of the incoming traffic at the sink (F_I^{Sink}) is reported, which directly shows the load at the bottleneck in the network. As with the validation experiments, the network is structured as a set of four rings ($D = 4$) with a uniform density of eight neighbors per node ($C = 8$), resulting in a network size of 108 nodes. For the radio model the popular CC1000 radio is used (cf. Table 5), and perfect links are assumed (i.e., external interference is not taken into account). Unless overruled explicitly, these network and radio settings are also used in the other experiments discussed in the remainder of this study.

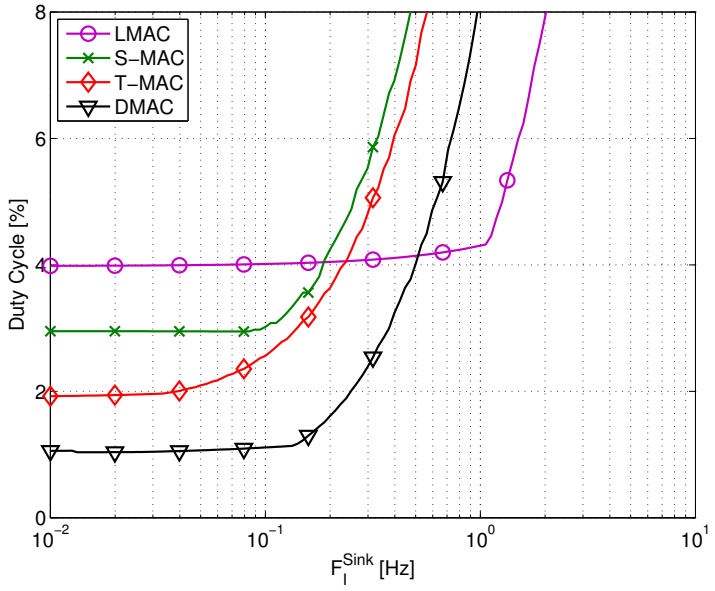
Figure 19 shows the individual Pareto fronts for the fundamental data-load versus energy-consumption trade-off. It consists of two plots each having the (optimized) duty cycle on the vertical axis, and the increasing

data load on the horizontal axis. The left plot features the "slot-based" protocols having the receiving nodes listen for a long period (S-MAC, T-MAC and D-MAC) or into many slots (LMAC). The right plot, on the other hand, features the "CP-based" protocols, i.e., the channel polling ones that periodically check for activity. Comparing the two plots clearly shows the advantage of the CP-based protocols where receiving nodes spend energy only in the case of ongoing activity; the difference is especially large for aggregate data rates below 1 message per 10 seconds.

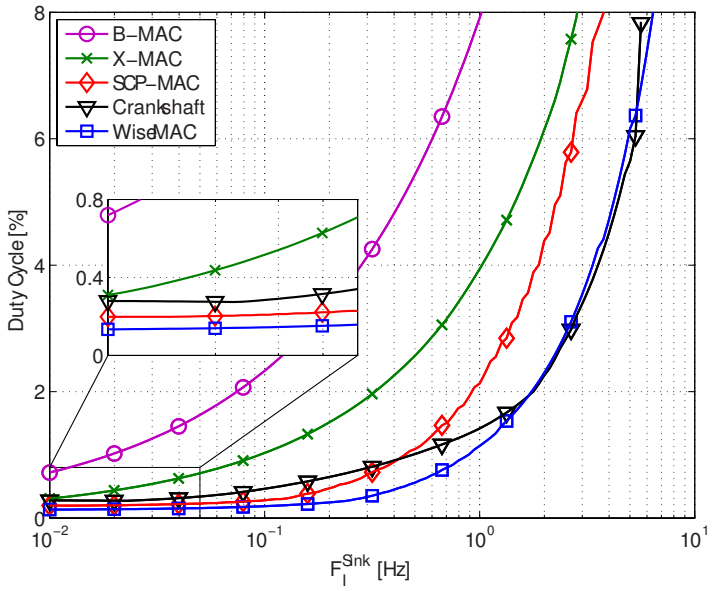
The slot-based protocols do all have a quite high offset for very low data rates, i.e., a lot of energy is spent even when almost no data is communicated through the network. A consequence of this "hot" idle mode is that a certain traffic load can be accommodated for free as indicated by the initial flatness of the curves. Once the data load crosses a certain threshold (around 10^{-1} Hz for S-MAC, T-MAC and D-MAC; around 10^{0} Hz for LMAC) default parameter settings need to be adjusted to handle the increased traffic to the best of the protocol's capabilities. The reason that S-MAC is less efficient than T-MAC and D-MAC in idle mode is a direct outcome of the minimal active period which is the longest for S-MAC and the shortest for D-MAC. LMAC on the other hand spends a lot of energy in idle mode due to its large synchronization overhead required to maintain the TDMA structure. The advantage of this structure becomes apparent with higher data rates, showing a much better energy efficiency than the other slotted protocols.

The CP-based protocols consume significantly less energy in idle mode since the nodes only perform short carrier sensing and do not have to listen into long slots. One might anticipate that, for very low data traffic, SCP-MAC and Crankshaft perform worst in the class of CP-based protocols since they incur the overhead of maintaining a slotting structure. However, the results in Figure 19 show that this overhead already pays off compared to B-MAC and X-MAC for very little traffic. This is due to the parameterization of the polling interval, which can be set to a very large value when a structure is maintained. For B-MAC and X-MAC on the other hand, a long polling interval also results in very long messages (preambles) when transmitting, hence the optimized polling interval is shorter for the unstructured CP-based protocol variants. WiseMAC exhibits the best energy efficiency for very low data rates. This can be attributed to its design of having the nodes synchronize on a per-link basis without the necessity of maintaining an expensive global structure. When comparing WiseMAC with B-MAC and X-MAC, a much longer polling interval can be chosen, since this does not imply an increased message length.

The energy consumption of SCP-MAC increases the fastest once the data rate at the sink exceeds 1 message per 10 seconds. This is due to



(a) Slot-based protocols



(b) CP-based protocols

Fig. 19: Energy-load trade-off (after Pareto optimization of MAC parameters).

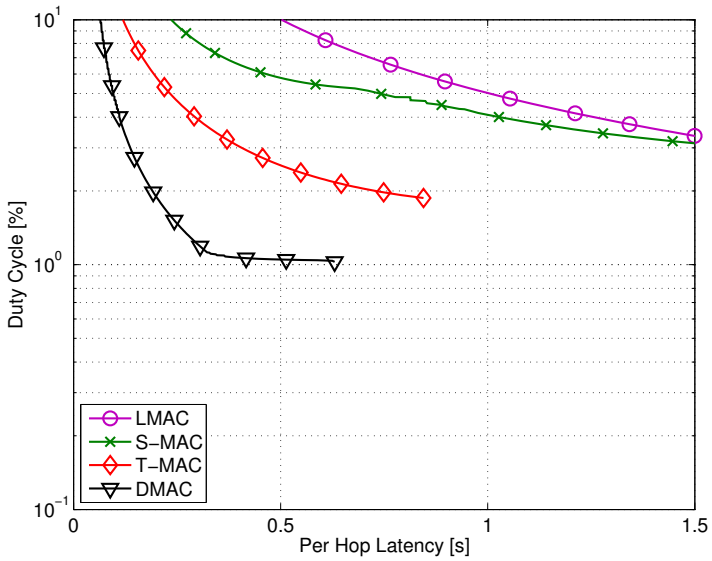
its global synchronization, grouping all communication activity into a single slots. This does greatly reduce the available radio bandwidth and further results in frequent overhearing. This overhearing is especially expensive due to the two contention-window scheme of SCP-MAC, which results in overhearing the second contention window for all nodes. It would therefore be more energy efficient to have a single contention window using the Sift [JBT06] contention resolution scheme. WiseMAC and Crankshaft on the other hand use the complete channel bandwidth, which allows for an increased energy efficiency for higher data rates. For Crankshaft this is achieved by orchestrating to different slots, while WiseMAC inherently balances the channel activity by having random drifting channel-access times for the different nodes.

For the highest data rates Crankshaft shows about the same energy efficiency as WiseMAC. This can be attributed to the special sink mode of Crankshaft that spreads the load evenly across all unicast slots instead of using just one slot. By itself this does not change the energy spend on sending and receiving, but the reduced pressure allows for a different setting of the internal protocol parameters of Crankshaft; in particular, it may operate with fewer, larger slots per frame reducing the energy spend on carrier sensing. This gives Crankshaft an advantage over WiseMAC, which must select a relatively short wakeup interval (T_w) to meet its boundary condition of handling at most one message every two wake-up slots of the sink (cf. Equation (3.17)). The impact of having a dedicated sink mode is further detailed in Section 3.4.6.

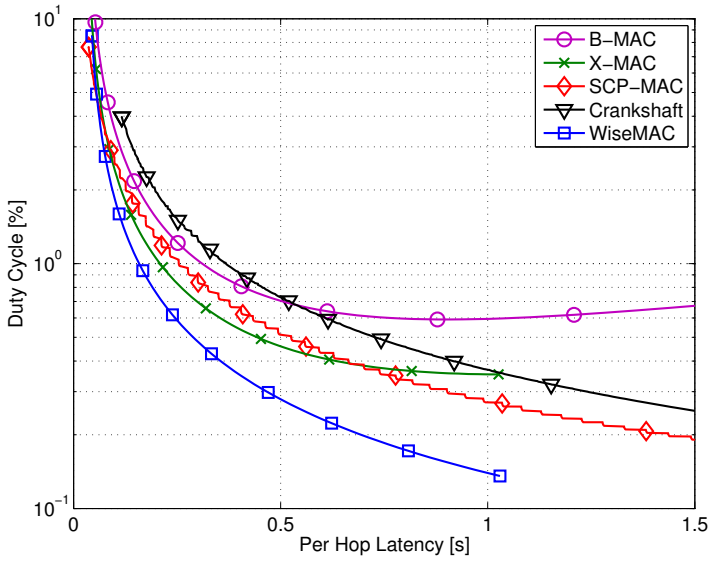
3.4.4 Energy Consumption vs. Latency

The second experiment studies the trade-off between energy consumption and latency. This is of particular importance for event-based applications such as burglar alarms that rarely exercise the sensor network, but do need a fast response. The low-latency requirement forces, for example, B-MAC to select a much shorter wake-up period T_w than is necessary for handling the near-zero data rate. Figure 20 shows the fundamental trade-off between average per-hop latency and energy consumption (duty cycle) for a six-hop event message injected into an idle network. In order to ensure that the topology is being maintained, it is assumed that every node sends a status message to its parent node every 10 min, checking for its availability.

The energy-latency trade-off is related to the efficiency plot discussed in the previous section. In particular, the high offset in the energy consumption of the slot-based protocols can also be observed in Figure 20(a), limiting the minimal energy consumption to a duty cycle of 1 % at best (D-MAC). The message latency depends a lot on the protocol design. Es-



(a) Slot-based protocols



(b) CP-based protocols

Fig. 20: Energy-latency trade-off (after optimization of MAC parameters).

pecially the TDMA structure of LMAC delays the message greatly due to the rather long frame time, whereas the staggered wake-up slots of D-MAC pay off well. However, it needs be considered that D-MAC is likely to result in a largely increased delay in the case of a link error, since there is no effective way of signaling the higher levels in the tree of a pending retransmission.

The CP-based protocols depicted in Figure 20(b), show the possibility for very low duty cycles if latency cut-backs are possible. WiseMAC stands out with its superior energy-latency trade-off. The reason is two-fold: Firstly, WiseMAC already showed to operate very energy efficient for low data rates in the previous section. Secondly, due to the random access times of the nodes, the average waiting time for the parent to wake up is $T_w/2$. This is in contrast to SCP-MAC, which also operates very energy efficient for very low data-rates, yet delays the message by T_w at every hop. Overall, this results in SCP-MAC roughly having a doubled latency compared to WiseMAC. A similar trend holds for the frame structure of Crankshaft, which delays the message due to the long frame time (analog to LMAC). The delay of B-MAC and X-MAC is quite different, despite their similar design. This is attributed to the strobed preamble of X-MAC, which reduces the average preamble length and message delay by a factor of two. Note that for both B-MAC and X-MAC that the latency-efficiency curve levels off for high message delays. This is due to the regular status messages that are sent every 10 min. Furthermore it can be observed that the curves for WiseMAC and X-MAC are limited to a maximum latency of about 1 s. The underlying cause is the polling period T_w, having an upper bound of 2 s (see Table 6).

It should be considered that Figure 20 plots the *average* latency, while some real-time applications might want to consider the *worst-case* latency. For B-MAC and SCP-MAC these are very similar, but for WiseMAC, Crankshaft and X-MAC the worst-case latency is in fact almost doubled. Depending on the application at hand this may, or may not, change the picture for selecting the most suitable protocol.

3.4.5 Sensitivity

The optimization presented before tunes the MAC-protocol parameters for the most energy-efficient operation in a specific setup, that is, for a specific set of network characteristics, radio parameters, etc. The following sensitivity analysis shows how the most energy-efficient protocols, namely Crankshaft, SCP-MAC and WiseMAC, are influenced by changes in the setup.

The three most energy-efficient protocols are all based on periodic channel polling combined with some form of synchronization; whereas

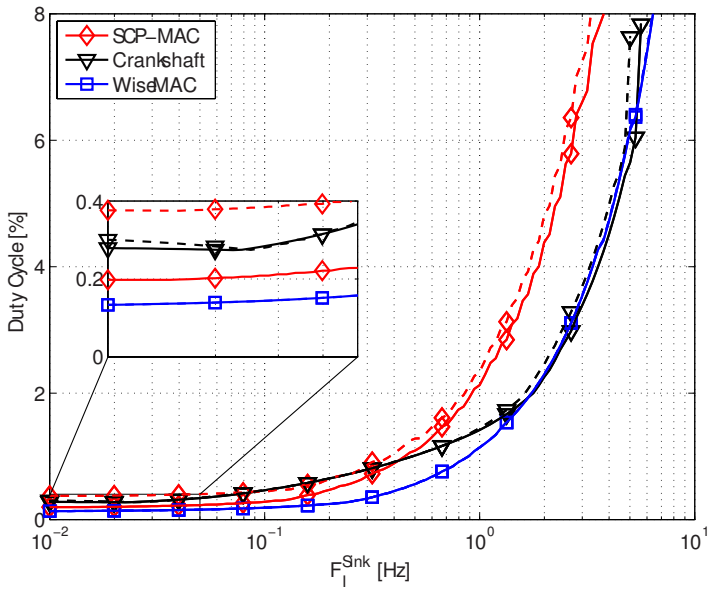


Fig. 21: Sensitivity for clock drift: 30 ppm (solid) vs. 120 ppm (dashed).

WiseMAC synchronizes with each node individually, Crankshaft and SCP-MAC are based on globally synchronized slots. The three protocols have in common that they guard for potential clock drift, making it likely that the node's clock drift parameter θ impacts the energy efficiency. This effect is depicted in Figure 21, which shows the energy efficiency of the protocols for clock drift settings of 30 (default) and 120 ppm. (Initial experiments with 60 ppm showed that none of the protocols are significantly affected.) WiseMAC only shows one (rather thick) line, indicating its robustness against clock drift, which is a consequence of the dynamically adapted guard time of WiseMAC. SCP-MAC shows some sensitivity to clock drift. Especially for low data rates, the duty cycle is almost doubled (0.20 % vs. 0.38 %). This is attributed to the long synchronization interval in combination with a quadrupled guard time, resulting in long guard times overheard by all nodes in the network. For higher data rates, the guard time is not the dominating factor anymore, since the network is tightly synchronized due to the frequent traffic. Crankshaft is less affected by the clock drift than SCP-MAC. This is rather surprising, since both protocols require global synchronization. Crankshaft minimizes overhearing of the data traffic with its slot assignment, which explains the smaller offset for higher data rates. For low data rates there is almost no difference for the different clock drifts. Detailed inspection of the protocol parameters settings for this low data traffic showed, that the prolonged guard time results in an increased frame time (cf. Equation (3.20))

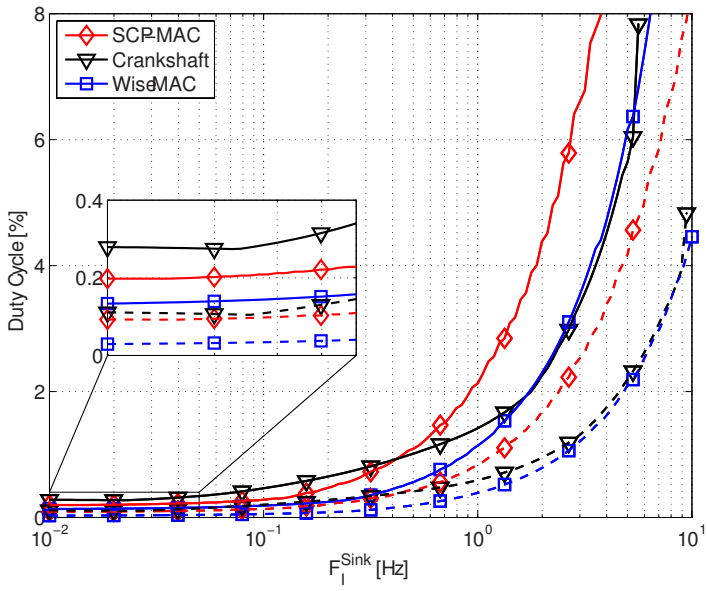


Fig. 22: Sensitivity for radio: CC1000 (solid) vs. RFM TR1001 (dashed).

(3.5 s vs. 10.9 s). This is still sufficiently short to accommodate all traffic in the network and allows compensating the prolonged guard time by fewer channel polls.

The analysis so far is based on the CC1000 radio transceiver, featuring multi-channel operation but having limitations in the available bandwidth and rather long switching times. Figure 22 depicts the impact of using the fast RFM TR1001 radio transceiver; the solid curves plot the default (CC1000) performance, whereas the dashed, more efficient curves derive from the faster (TR1001) radio. Using a faster radio (0.5 ms vs. 2.10 ms 'switch on' time and 5.75 kbps vs. 2.4 kbps bandwidth) impacts the energy demands of the protocols with improved efficiency for all data rates. Overall, the faster switching time is most beneficial for low data rates where most energy is spent on polling the channel, or rather on turning the radio (CC1000) on *before* probing the channel. The faster transmission rate is most beneficial for higher traffic rates where overhearing of headers and (partial) messages has a larger impact on the overall energy consumption.

The network model assumes a constant node density in the network. However, it is likely that a real deployment shows areas with an increased node density. This would increase the number of neighbors and therefore the number of messages that can be potentially overheard. Figure 23 shows this effect, featuring the energy consumption of the protocols when doubling the number of nodes, hence, with 8 (solid lines) and 16 (dashed

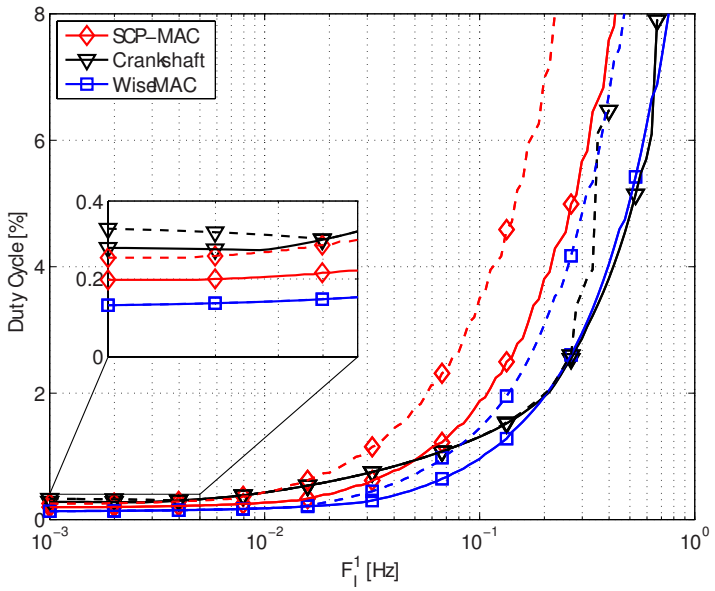


Fig. 23: Sensitivity for network density: 8 (solid) vs. 16 (dashed) neighbors.

lines) neighbors. Note that the x-axis plots the input frequency of the bottleneck nodes next to the sink (F_I^1) to ensure that the same amount of traffic is handled in both cases; when focusing on the aggregated input rate at the sink (F_I^{Sink}) as before, the send rate would have to be halved, making for an unfair comparison (i.e., protocols becoming seemingly more efficient for higher densities due to handling proportionally less traffic). All protocols show increased energy consumption in the denser network, however the amount varies. SCP-MAC shows a large increase in energy consumption since all messages (i.e., preambles and headers) are overheard. Crankshaft, especially designed for high density networks, is hardly affected (up to around 0.3 Hz) due to its unicast slots, minimizing the probability of overhearing messages. For very low data rates however, the duty cycle of Crankshaft is slightly increased. This is attributed to the increased number of synchronization messages overheard due to the increased number of neighbors. WiseMAC is not based on a global schedule and is synchronized with specific neighbors (i.e., parent and children) on an individual basis, resulting in shorter guard preambles compared to Crankshaft or SCP-MAC. This explains WiseMAC being only affected for higher data rates, as the probability of overhearing increases with higher node density and data rate.

3.4.6 Bottleneck Sink

In the typical, data-gathering scenario analyzed in this study the sink becomes the bottleneck because it has to handle most traffic. Therefore, any optimization in the access mode of the sink (and its immediate neighbors) is likely to have a large impact. Crankshaft includes an optimization in which the sink is always listening in contrast to ordinary nodes following an active/sleep duty cycle (at slot level). This increases the effective bandwidth to the sink, allowing Crankshaft to operate more efficiently, and causing it to challenge the energy efficiency of WiseMAC for high data rates (cf. Section 3.4.3).

The idea of employing such a special (always on) sink mode for other MAC protocols as well is rather attractive, but not always straightforward to implement or may even not be applicable at all. For instance, the concept of LMAC of scheduling send slots already gets *all* nodes to listen in on every slot, ruling out additional listening by the sink.[4] In the case of slotted protocols (S-MAC, T-MAC, D-MAC and SCP-MAC) it would be possible to have nodes send messages to the sink 'outside' the normal active period in a slot when the sink is always listening, but at a considerable increase in complexity (additional timers and bookkeeping) rendering it less attractive. For the class of random access protocols (B-MAC and WiseMAC), however, only a minor modification is required to take advantage of sink that is always listening. As suggested in [PHC04], the nodes next to the sink are no longer required to send a stretched wake-up preamble to meet a specific channel poll. This minimal optimization greatly reduces the transmission energy for nodes next to the sink and also reduces the channel load at the sink reducing overhearing overheads.

To determine the impact of a special sink mode the models are adapted for Crankshaft, B-MAC and WiseMAC (cf. Appendix B on page 153). The slotted protocols are not considered, because they would require a major protocol redesign and have not proven to be very energy efficient to start with. Figure 24 shows the gain in energy efficiency that results from optimizing the protocols to keep the sink listening at all times. All three protocols benefit from a special sink mode, but Crankshaft benefits the most. Its performance without the optimization rapidly deteriorates when the data rate increases, and the resulting bottleneck of all data passing through one slot causes it to violate the basic parameter constraints for data arriving faster than two messages per second.

WiseMAC and B-MAC follow the same channel access strategy to benefit from the increased resources of the sink. Nevertheless their impact

[4] A related optimization to LMAC [CvHH04] is to allocate more slots to the immediate neighbors of the sink, which increases the effective bandwidth to the sink, but does not reduce the overhearing overhead as Crankshaft does.

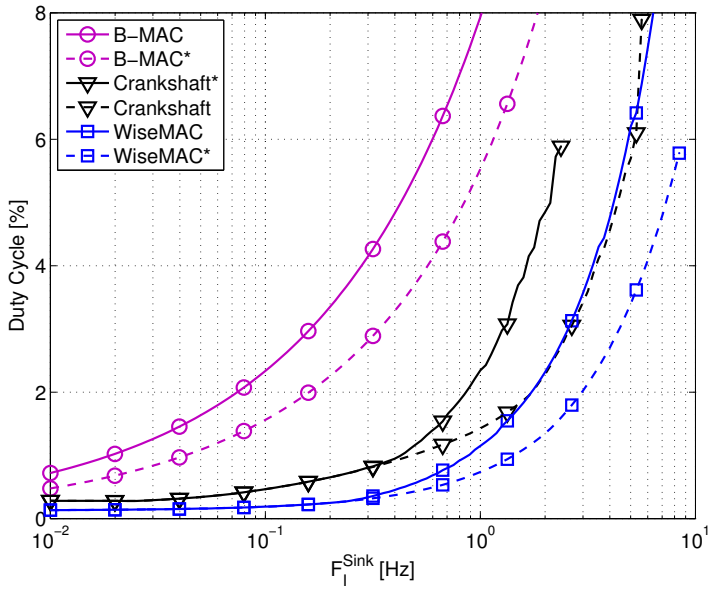


Fig. 24: Benefit of having the sink keeping its radio switched on (dashed line) over ordinary duty cycling (solid line). The variants marked with an asterisk represent the adapted protocols.

differs largely: B-MAC benefits a lot, rather independent of the data rate, whereas WiseMAC only shows a substantial gain for higher data rates. This can be explained by the very energy-efficient operation of WiseMAC for low data rates, spending most energy on regular channel polls, which makes the energy saved on transmitting messages of little importance. For higher data rates on the other hand, the energy consumption for transmitting and overhearing messages is a non-neglectable factor, allowing WiseMAC to benefit from the shortened preamble. In particular, WiseMAC does outperform Crankshaft for higher data rates, making this variant the overall most energy-efficient protocol.

3.4.7 Broadcast vs. Unicast

The common traffic pattern for low-power data gathering is unicast. Nevertheless, broadcasts are sent at times, i.e., for announcing a change in the topology. In the following it is analyzed how such broadcasts will impact the contention-based MAC protocols.

The impact of the energy consumption differs greatly for broadcast messages, as certain protocols are better suited for sending them. Crankshaft with its special broadcast slots and SCP-MAC with its single slot for all communication are very well suited broadcast traffic. For B-

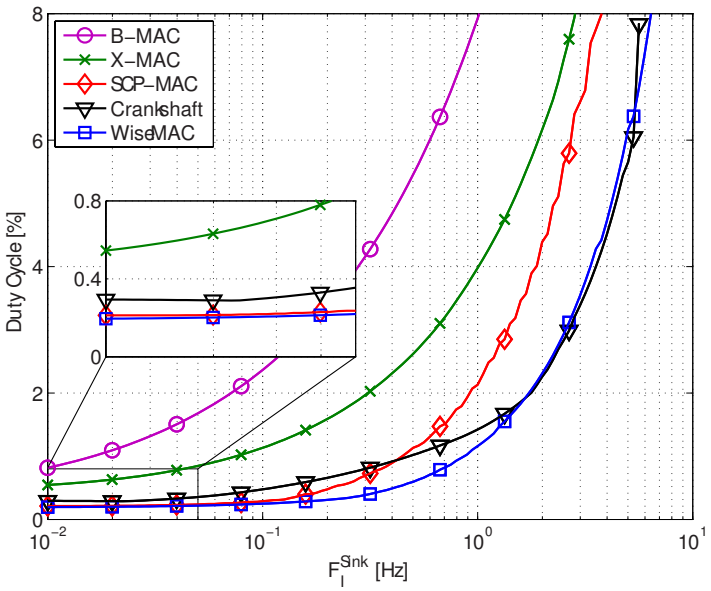
MAC the difference between a unicast and a broadcast is negligible, since the long wake-up preamble is waking-up all neighbors anyhow. X-MAC and in particular WiseMAC highlight a reduced wake-up preamble for unicast messages, which has to be extended to the long wake-up preamble of B-MAC for broadcast traffic. For receiving broadcast messages X-MAC and WiseMAC differ greatly: WiseMAC sends a packet stream which allows the receiver to switch off the radio after receiving one complete packet out of the packet stream. With X-MAC on the other hand the receiving node has to stay awake until the end of the wake-up strobe, waiting for the packet to be sent.

In order to analyze the impact of sending additional broadcast messages to the regular unicast traffic, the models were adapted according to the discussion above. If the broadcasts are limited to one per hour, per node as shown in Figure 25(a), not much costs are added compared to the solely unicast case as depicted in Figure 19(b). If however the broadcast frequency is increased to once every 5 minutes, as shown in Figure 25(b), a clear impact is observed for low data rates. Especially WiseMAC and even more X-MAC are affected by the additional broadcasts. On the other hand, SCP-MAC and Crankshaft are only moderately influenced. Especially Crankshaft is very well suited for a combined unicast and broadcast traffic, due to its dedicated unicast and broadcast slots.

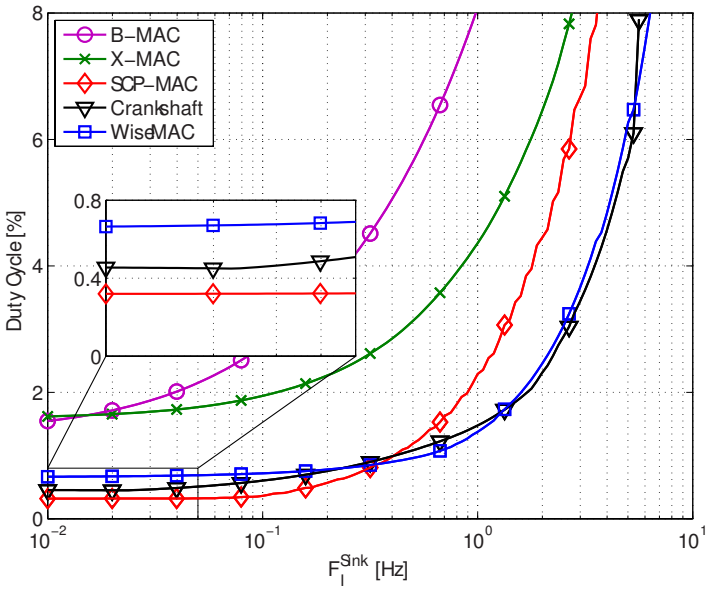
3.5 Packet-Based vs. Byte-Stream Radios

Most of the discussed protocols are designed for byte-stream radios like the CC1000 and the RFM TR1001 analyzed in the last section. However, state-of-the-art platforms tend towards packet-based radios such as the IEEE 801.15.4 compliant CC2420 used for the MicaZ and the TelosB node. With such a radio, the packet is first copied into a dedicated radio buffer. After receiving a trigger, the radio sends the packet autonomously, in particular also adding the preamble and CRC checksum. For the CC2420 the length of this preamble can be set between 4 and 16 bytes, which is generally too short to accommodate low-level MAC synchronization techniques like, for instance, the long wake-up preamble of B-MAC and SCP-MAC's guard preamble and contention resolution "tone".

As shown in [YSH06] a long preamble can be replaced by a stream of packets. The granularity of such an artificial preamble depends on the minimal packet size and the data rate (16 bytes and 250 kbps respectively for the CC2420), which results in a minimal packet transmission time of 0.51 ms. The gap between two consecutive packets can be reduced to 30 μs. It is therefore possible to imitate an arbitrarily-long preamble with a granularity of 0.54 ms. This allows adopting the protocols discussed



(a) One broadcast every hour



(b) One broadcast every 5 min

Fig. 25: Additional to the unicast traffic (cf. Figure 19(b)) every node sends broadcasts.

in this study to packet-based radios with minimal loss compared to a byte-stream radio. In exchange, the MAC implementation does not have to burden the microcontroller with transceiving the byte stream, checking for a start-frame delimiter, and performing a CRC check.

Adapting the models for packet-based radios is straight forward. For instance, if the preamble exhibits a certain granularity, the model has to account for this as shown with the packet-based X-MAC model (cf. Equation (A.19)). For the popular CC2420 radio, this makes only a small difference, due to the fine granularity (0.54 ms) that can be achieved. This allows to analyze the energy vs. data-rate trade-off as well for the CC2420 radio, using the existing models. Since the general trend is very similar to the ones of the CC1000 and TR1001 radios, the plot is not shown but the results are briefly discussed for the CP-based protocols.

For very low data rates the energy consumption of the MAC protocols for the CC2420 radio is very similar to that of the CC1000 (cf. Figure 19(b) for $F_I^{Sink} < 0.1$). This can be attributed to the fact that both radios have similar switching times (cf. Table 5) and the fact that the regular channel polls are the main source of radio activity. For higher data-rates, it is mainly X-MAC, WiseMAC and Crankshaft that benefit from the increased bandwidth, which allows them to minimize the energy consumption in the same order as the faster TR1001 radio does (cf. Figure 22 for $F_I^{Sink} > 1$).

3.6 Conclusions

The fundamental need for energy-efficient operation has been a driving force behind the development of many WSN-specific Medium Access Control protocols. Each protocol strikes a different balance between performance (latency, throughput) and energy consumption; all claiming to be more efficient than the canonical S-MAC protocol, but evaluated with different workloads, simulation environments, and hardware platforms making it hard to assess the true benefits of the individual MAC protocols.

This chapter takes an analytical approach to answering the question "which protocol is best?" given a set of external conditions including radio hardware characteristics, network topology, and workload. The study focuses on low data-rate applications for which the energy-efficient operation of the MAC protocols is most critical as there is little room for amortizing overheads. This focus also keeps the analysis tractable as complications like contention need not be modeled in detail. However, boundary conditions are introduced for safeguarding the contention-free operation.

An extensive exploration of the MAC protocols that iterates over different data rates, clock drifts, network densities, and radio characteristics has

revealed a number of important protocol features that, collectively, warrant for a very efficient handling of low data-rate applications. First, the timing uncertainty introduced by clock drifts is best handled (guarded) at the sending node as that avoids any overhead at the receiver and other nodes overhearing the message (one vs. many). Second, (randomized) channel polling reduces idle listening to a great extent when compared to more structured approaches organizing time into slots (and frames). Third, taking advantage of the line-powered sink node by keeping it on at all times allows one-hop neighbors to short cut sending procedures and alleviates the bandwidth bottleneck in the network; usually the reduced pressure on the (bottleneck) nodes surrounding the sink allows for a more efficient setting of the internal protocol parameters, reducing overall energy consumption. Finally, protocols like T-MAC and SCP-MAC that cluster communication into a slot suffer from overhearing overheads when compared to protocols like X-MAC, Crankshaft, and WiseMAC that spread (randomize) traffic over time, which further increases the available bandwidth.

Although announcing an absolute winner is impossible, the study identified that the WiseMAC protocol shows a remarkable consistent behavior across a wide range of operational conditions. This can be attributed to its design choice of having a random access scheme in combination with local synchronization. This combination does not only minimize the energy consumption, but also maximizes the available channel bandwidth. WiseMAC is therefore the prime candidate for being used for safety-critical event monitoring, where the data-load is usually low, yet bandwidth is required in the case of an event. In particular, WiseMAC is being used in the safety-critical communication stack presented in Chapter 5.

4

DiMo: Distributed Node Monitoring

Safety-critical WSNs require that the health status of the sensor nodes is continuously monitored. Node failures need to be captured by the system and reported to administrators within a given delay constraint (e.g., within 5 min). Due to the resource constraints of WSN nodes, traditional network management protocols such as the Simple Network Management Protocol (SNMP) adopted by TCP/IP networks are not suitable. In particular, WSNs require a light-weight network-management approach tailored specifically to their unique constraints.

WSN deployments can be categorized by their application scenario: data-gathering applications and event-detection applications. For data-gathering deployments, health status monitoring is straight forward. Monitoring information can be forwarded to the sink by specific health status packets or the information can be embedded into the regular data packets. Administrators can usually diagnose the network with a helper program. NUCLEUS [TC05] is one of the network management systems for data-gathering application of WSN. Event-detection deployments on the other hand do not have regular traffic to send to the sink, which makes the solutions for data-gathering deployments unsuitable. In this context, health status monitoring can be quite challenging and has not been discussed explicitly in the literature.

In an event-detection WSN, there is no periodic data transfer, i.e., nodes maintain radio silence until there is an event to report. While this is energy efficient, it does mean that there is no possibility for the sink

to distinguish whether the network is still up and running (and waiting for an event to be detected) or if some nodes in the network have failed and are therefore silent. Note that an operational node that gets disconnected from the network is also considered as failed. Furthermore, for safety-critical systems, the specifications commonly include a hard time constraint for accomplishing the node-health status-monitoring task. In the exemplary case of a fire-alarm system, a failed node must be reported to a control station within 5 minutes, in order to comply with regulatory specifications [Eur08].

In safety-critical WSNs there is also a time bound for delivering data to a sink in the case of an event (e.g., a fire-alarm must be reported within 10 s). This makes it unfavorable to set up a routing table and neighbor list after the event has been detected. Hence the network should always be ready to forward a message to the sink immediately whenever necessary, even though there is no regular data transfer in the network. The lack of regular data transfer in the network also leads to difficulty in detecting bad quality links, making it challenging to establish and maintain a stable and robust network topology.

This chapter provides a solution for distributed node monitoring called DiMo, which consists of two functions: (i) Network topology maintenance and (ii) node health status monitoring. DiMo is developed based on the following design goals:

- In safety-critical event monitoring systems, the status of the nodes needs to be monitored continuously, allowing the detection and reporting of a failed node within a certain *failure detection time* T_D, e.g., $T_D = 5\,\text{min}$.
- If a node is reported failed, a costly on-site inspection is required. This makes it of paramount interest to decrease the false-positive rate fp, i.e., wrongly assuming a node to have failed.
- In the case of an event, the latency in forwarding the information to the sink is crucial, leaving no time to set up a route on demand. It is required for the system to maintain a topology at all times. In order to be robust against possible link failures, the topology needs to provide redundancy.
- To increase efficiency and minimize energy consumption, the two tasks of topology maintenance (in particular monitoring of the links) and node monitoring should be combined.
- Maximizing lifetime of the network does not necessarily translate to minimizing the average energy consumption in the network, but rather minimizing the energy consumption of the node with the

maximal load in the network. In particular, the monitoring should not significantly increase the load towards the sink.

- It is assumed that there is no regular data traffic in event-detecting WSNs, with the possibility that no messages are sent to the sink for days, weeks or even months. Hence it is not required to optimize routing or load balancing for regular data. This also rules out approaches that estimate the link quality based on the ongoing data flow.

- Wireless communications in sensor networks is known for its erratic behavior as shown in Chapter 2. DiMo assumes such an environment with unreliable and unpredictable communication links, and hence message losses must be taken into account.

The remainder of this chapter is organized as follows. Section 4.1 introduces a redundant topology, allowing for reliably and timely forwarding events, even in the presence of link failures. Section 4.2 details DiMo, a distributed solution for monitoring nodes within a bounded delay, a minimized message load and at the same time allowing checking the quality of the links used for forwarding events. DiMo is compared analytically in Section 4.3 with the performance of two prominent monitoring schemes Memento and Sympathy. Section 4.4 evaluates DiMo based on extensive simulations performed on the state-of-the-art WSN simulator Castalia and furthermore introduces a load-balancing scheme for greatly prolonging the system's lifetime. Finally, Section 4.5 summarizes this chapter.

4.1 Topology Maintenance

Forwarding a detected event with minimal delay requires maintaining a topology that is robust against node and link failures. In the following, a *redundant topology* is proposed that provides two disjoint and loop-free paths to the sink for all nodes in the network, except for the neighbors of the sink. In this redundant setup, DiMo allows monitoring the topology and the nodes at the same time. It should be noted that the node-monitoring part of DiMo can be adapted to any other network topology.

Nodes have usually a multitude of neighbors they can communicate with. In order to save management overhead and resources, it is not advisable to connect to all of them for building a topology. In the following it is discussed how to select (and connect to) a suitable subset of them, for building a redundant topology.

The topology is based on so called *relay nodes*. A relay is a neighbor that has a smaller *cost metric* C and provides one or more routes to the sink. Loops are inherently ruled out if packets are always forwarded to relay nodes. In order to provide redundancy, a node must be connected with at least two relay nodes, but not necessarily to all its relay nodes. Such a node is called *redundantly connected*. If all nodes in the network are redundantly connected, the network is called redundantly connected.[1]

Within the existing topology, the *level* $\mathcal{L}$ of a node represents the minimal hop count to the sink. The level is infinity if the node is not connected. The *redundant hop count* $\mathcal{R}$ is defined as the maximum redundant hop count in the set of connected relays plus one. The redundant hop count is infinity if the node itself or one of its ancestors in the topology is not redundantly connected. Otherwise, $\mathcal{R}$ represents the longest path to the sink. If and only if all nodes in the network have a finite redundant hop count, the network is redundantly connected.

For the cost metric C, the tuple $(\mathcal{L}, \mathcal{R})$ is used in this work. A node A has a smaller cost metric than node B if

$$C_A < C_B \Leftrightarrow \mathcal{L}_A < \mathcal{L}_B \vee (\mathcal{L}_A = \mathcal{L}_B \wedge \mathcal{R}_A < \mathcal{R}_B). \tag{4.1}$$

The topology is set up starting from the sink broadcasting its current cost metric, i.e., the level $\mathcal{L}_S$ and the redundant hop count $\mathcal{R}_S$ both equal zero. Upon receiving such a topology announcement at a node A, the node checks whether the announcing node is a suitable relay and if yes, sends a *connection request* message. The potential relay can then accept or decline this request whereas in the former case, node A will update its cost metric and broadcast it in the case of a change.

The topology management function aims to establish and maintain a redundantly connected network whenever possible. This might not be possible for sparsely connected networks, where some nodes might only have one neighbor and therefore cannot be redundantly connected by definition. Sometimes it would be possible to find alternative paths with a higher cost metric, which in turn would largely increase the overhead for topology maintenance (e.g., for avoiding loops).

During the operation of the network, DiMo continuously monitors the links, which allows the detection of degrading links and allows triggering topology adaptation. In such a case, the node has to search for a more suited relay and disconnect the degrading one. Due to DiMo's redundant structure, the node is still connected to the network, while searching for a new relay. Hence in the case of an event, the node can forward the message

[1] Since there is a singularity at the sink, it is not possible for all nodes in the first-hop neighborhood of the sink to be redundantly connected. Nodes that have the sink as a relay node are therefore allowed to be each others relay, all having the same cost metric. This will result in a single loop in the first-hop neighborhood of the sink.

without delay. Whereas DiMo allows triggering a topology adaptation, the process of adapting the topology is well studied in literature and complementary to this work.

A redundant topology that provides two disjoint paths (based on two redundant trees) was already presented in [MFB99]. The problem with two single trees in a wireless environment is, that two single link failures (one on each of the trees) already results in a disconnected network. In comparison, the topology presented above can usually deal with several link failures at the same time due to its mesh characteristic. In a worst-case setting however, two single link failures will also result in a disconnected network. The topology setup is much simpler for the algorithm presented here. The drawback of this simplicity is that a redundant topology cannot be built for some special topologies (e.g., a circle), whereas [MFB99] always ensures two independent trees if this is possible.

4.2 Monitoring Algorithm

All links and nodes have to be monitored periodically, in order to assure the reliable transmission of events to the sink. In the following a distributed algorithm called DiMo for topology, link and node monitoring is presented. From the underlying MAC protocol it is required that an acknowledged message transfer is supported.

4.2.1 Algorithm

A monitoring algorithm is required to detect failed nodes within a given *failure detection time* T_D (e.g., $T_D = 5\,\text{min}$). A node failure can occur for example due to hardware and software errors or because a node runs out of energy. Furthermore, an operational node that gets disconnected from the network is also considered as failed.

The monitoring is done by so called *observer* nodes. The observer monitors, based on a timeout of the *monitoring time*, whether the node has *checked in* by sending a *heartbeat*. If yes, the timer is reset, if not, the observer sends a *node-missing* message to the sink. The node is being monitored only by one observer node at a time. If there are multiple observer nodes available, the node alternates the monitoring task between them. For instance, if there are three observers, every one of them observes the node every third monitoring time.

An observer should not only check for the liveliness of the nodes, but also for the links that are being used for sending data packets to the sink in case of a detected event. These two tasks are combined by selecting the relay nodes as observers, which greatly reduces the communication load

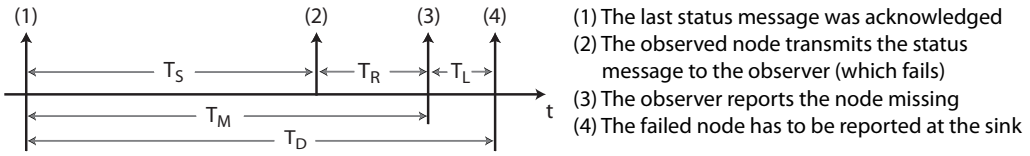


Fig. 26: DiMo's node monitoring scheme in a node-missing scenario.

and maximizes the network lifetime. The monitoring scheme is illustrated in Figure 26 and is further detailed in pseudo code in Algorithms 1, 2 and 3.

The given failure detection time T_D sets the upper bound for the *monitoring interval* T_M, i.e., the interval within which a node has to send a heartbeat. Since failure detection is measured at the sink, the detection of a missing node at the relay needs to be forwarded, resulting in an additional maximal delay T_L. The monitoring interval has to account for this delay and is set to

$$T_M \leq T_D - T_L. \tag{4.2}$$

It has to be ensured by the node that it is being monitored every T_M time units by one of its observers.

The observed node has to take into account that the heartbeat can be delayed, either by message collisions or link failures. Hence the node should send the heartbeat before the monitoring timer of the relay expires and leave room for retries and clock drift within the time window T_R. Hence the node should send the heartbeat in a shorter interval T_S than the monitoring timer of the observer:

$$T_S \leq T_M - T_R \leq T_D - T_L - T_R. \tag{4.3}$$

The schedule of reporting to an observer is only defined for the next monitoring time for each observer. Whenever the node checks in, the next monitoring time is announced with the same message. For every heartbeat sent, the old monitoring timer at the observer can be cancelled and a new timer can be set according to the new time.

4.2.2 Packet Loss

Wireless communication always has to account for possible message losses. Sudden changes in the link quality are always possible and even total link failures in the order of a few seconds are not uncommon (cf. Section 2.3.3). The time T_R for sending retries should therefore be sufficiently long to cover such blackouts. Though unlikely, it is possible that even after a duration of T_R, the heartbeat could not have been successfully forwarded to the observer and thus was not acknowledged, in spite of multiple retries.

Algorithm 1 Node: without queueing control (cf. Section 4.2.4)

```
1:  upon MonitoringTimerFires
2:      R_ID := FindNodeToCheckIn()
3:      T_S := 0
4:      if NodeStillRelay(R_ID) then
5:          T_S := NextUnscheduledMonitoringTime()
6:          SetMonitoringTimer(R_ID, T_S)
7:      end if
8:      T_M := T_S + T_R
9:      SendHeartbeat(R_ID, T_M, MaxRetries)
10: end upon
11:
12: upon NewRelayRegistered(R_ID)
13:     T_S := NextUnscheduledMonitoringTime()
14:     SetMonitoringTimer(R_ID, T_S)
15:     T_M := T_S + T_R
16:     SendHeartbeat(R_ID, T_M, MaxRetries)
17: end upon
18:
19: upon FailedHeartbeatToNode(R_ID)
20:     if RedundantRelaysAvailable(R_ID) then
21:         RR_ID := RedundantRelay(R_ID)
22:         T_S := NextHeartbeatToRelay(R_ID)
23:         T_M := T_S + T_R
24:         SetMonitoringTimer(RR_ID, T_S)
25:         SendRecoveryMsgToSink(RR_ID, T_M)
26:     end if
27: end upon
```

Algorithm 2 Observer: without queueing control (cf. Section 4.2.4)

```
1:  upon ReceivingHeartbeat(Msg, C_ID)
2:      if MonitoringTimerSet(C_ID) then
3:          CancelTimer(C_ID)
4:      end if
5:      UpdateTimer(Msg, C_ID, true)
6:  end upon
7:
8:  upon FiringMonitoringTimer(C_ID)
9:      ReportNodeAsMissing(C_ID)
10: end upon
11:
12: upon ReceivingRecoveryMsg(Msg, C_ID)
13:     ForwardMsg(Msg)
14:     UpdateTimer(Msg, C_ID, false)
15: end upon
16: function UpdateTimer(Msg, C_ID, regularMsg)
17:     if MonitoringTimeInMsg(Msg) then
18:         T_M := getMonitoringTime(Msg)
19:         if NodeNotYetMonitored(C_ID) then
20:             RegisterNodeAtTheSink(C_ID)
21:         end if
22:         SetMonitoringTimer(C_ID, T_M)
23:     else if regularMsg then
24:         UnregisterNodeAtTheSink(C_ID)
25:     end if
26: end function
```

For the observed node it is impossible to distinguish whether the heartbeat or the acknowledgement from the observer was lost and therefore has to account for both cases considering the monitoring timer. In case that only the acknowledgement got lost, the observer will assume that the node checks in according to monitoring time T_M announced in the heartbeat. In case the heartbeat got lost, the node will not be monitored at the announced time. Since the observed node cannot distinguish the two cases, the node has to ensure that it is being monitored by an additional observer at the time T_M.

The node has to assume that it will be reported failed at the sink, despite the fact it is still up and running. Should the node be redundantly connected, a *recovery message* is sent to the sink via another relay announcing being still alive. The sink receiving a recovery beacon and a node-missing message concerning the same node can neglect these messages as they cancel each other out. The details for handling packet losses and recovery messages are given in full detail in Algorithm 1.

Algorithm 3 Sink

```
1: upon ReceivingNodeRegisterMsg(C_ID, R_ID)
2:     UpdateNodeList(C_ID, R_ID, 'Add')
3: end upon
4:
5: upon ReceivingNodeUnRegisterMsg(C_ID, R_ID)
6:     UpdateNodeList(C_ID, R_ID, 'Remove')
7:     if NodeNotRegisteredAnymore(C_ID) then
8:         SignalNodeNotObserved(C_ID)
9:     end if
10: end upon
11: upon ReceivingNodeMissingMsg(N_ID)
12:     WaitForRecoveryMsg(MaxWait)
13:     if RecoveryMsgNotReceived(N_ID) then
14:         SignalNodeMissing(N_ID)
15:     end if
16: end upon
```

4.2.3 Topology Changes

In the case of a new relay being announced from the topology management, a heartbeat is sent to the new relay, marking it as an observer node. On the other hand, if a deprecated relay is announced, this relay might still be acting as an observer, and the node has to check in as scheduled. However, no new monitor time is announced with the heartbeat, which will release the deprecated relay of being an observer.

Whenever, a node is newly observed or not being observed by a particular observer anymore, this is indicated to the sink. Hence the sink is aware of which nodes are being observed in the network, and therefore always knows which nodes are up and running. This registration scheme at the sink is an optional feature of DiMo and depends on the user requirements.

4.2.4 Queuing Policy

Despite the redundancy in the network, it is always possible that a message cannot be forwarded due to degraded links, which is most crucial if this message announces a node being missing. Since a link can always show temporal recovery, it would be disastrous if the node manages to forward its own heartbeat, letting the sink believe that everything is in best order, while still having a node-missing message stuck in its queue.

Therefore, a monitoring buffer exclusively used for monitoring messages (i.e., heartbeat, node-missing and recovery messages) is introduced. These messages are queued according to their priority level: node-missing messages have the highest, heartbeat the second-highest and recovery messages the lowest priority. Since the MAC protocol and routing engine usually have a queuing buffer as well, it must be ensured that only one single monitoring message is being handled by the lower layers at the same time. Only if an ACK is received, the monitoring message can be removed from the queue (if a NACK is received, the message remains). DiMo only prioritizes between the different types of monitoring messages and does not require prioritized access over data traffic.

As long as no node is reported as missing, the sink can be certain that all nodes are up and running. However, if a node fails, the sink has to assume that monitoring messages were lost in the failed node's queue, and other nodes might be lost as well. Depending on the user requirements, the sink will flood a request to all relay nodes, asking to report a list with the currently observed nodes.

4.3 Analytical Analysis

In this section DiMo is analyzed analytically and compared to two state-of-the-art monitoring protocols Sympathy [RCK+05] and Memento [RB06].

4.3.1 Sympathy and Memento

In literature, there are very few existing solutions for health monitoring in wireless sensor network deployments. DiMo is the first solution specifically designed for event detection applications. However, the two prominent solutions of Sympathy and Memento for monitoring general WSNs can also be tailored for event gathering applications. Both these approaches send regular status messages to the sink: With Sympathy the status messages are forwarded individually to the sink, whereas Memento aggregates the status information in a bitmask. For both protocols, the sink requires to receive a status update from all nodes every so called *sweep interval*. This allows the sink to determine whether all nodes are still up and running. Subsequently the two protocols are detailed and their performance is compared to the node monitoring part of DiMo.

Sympathy [RCK+05] is a tool for detecting and debugging failures in pre- and post-deployment sensor networks, especially designed for data gathering applications. The nodes send periodic heartbeats to the sink that combines this information with passively gathered data to detect failures. Without the periodic data, passive snooping for application traffic is not possible, and hence the monitoring relies on the active heartbeats. For the failure detection, the sink requires receiving at least one heartbeat from the node every sweep interval, i.e., its lacking indicates a node failure. Sympathy performs poorly in practice without adaptation to wireless packet losses. To meet a desired false positive rate, the number of heartbeats n sent every sweep interval has to be increased, which however increases the communication cost. Similarly, the sweep interval can be prolonged, resulting in an increased detection latency.

Memento [RB06] is a failure detection system that requires a node to periodically send heartbeats to its so called observer node. Those

heartbeats are not directly forwarded to the sink node, but are aggregated in form of a bitmask that provides sufficient space to represent all nodes as a single bit (i.e., bitwise OR operation). The observer node clears its bitmask every sweep interval. If a node fails to send a heartbeat during the sweep interval, its status bit is not set in the bitmask. Hence the information of the missing node is disseminated every sweep interval by one hop, eventually receiving the sink (e.g., if a failed node is in a d-hop distance to the sink, it takes d sweep intervals until the node is detected missing). Due to this aggregation, there is no increased message load for nodes close to the sink, which is traded off by an increased latency. Memento is not making use of acknowledgements on the link layer and proactively sends multiple heartbeats every sweep interval, whereas this number is estimated based on the link's estimated worst-case performance and the targeted false positive rate.

The three different approaches for the monitoring, namely Sympathy, Memento, and DiMo result in different information the sink receives. With Sympathy, every single node reports at regular intervals and hence the sink can be sure that the node is still up and running. Memento on the other hand aggregates the information of the nodes. If a node fails to forward its heartbeat during the sweep interval, multiple nodes are usually reported missing at the sink. DiMo is reporting only the missing nodes to the sink.

4.3.2 False Positive Analysis

Wrongly assuming a node has failed, i.e., a false positive, normally results from packet losses during monitoring. In order to ensure a fair comparison between the protocols, the following three assumptions are made: (1) Whereas Memento sends r heartbeats every sweep, DiMo and Sympathy send up to $r-1$ retransmissions in the case of unacknowledged messages. (2) The three protocols are set to have the same sweep interval. This means that Memento's and Sympathy's sweep interval is equal to DiMo's monitoring interval. (3) The protocols have the same packet-loss probability p_l for each transmission. This results in a packet reception rate (PRR) of $p_r = 1 - p_l$. It should be noted, that the following false-positive analysis is based on link qualities with a $p_r \geq 0.8$. This is a minimal link quality a WSN should be able to cope with. In particular, no false positives should occur, and a possible event should still be forwarded successfully to the sink.

The false-positive rates for the three protocols can be determined as follows:

- For Sympathy, a false positive for a node occurs when the heartbeat does not arrive at the sink in a sweep interval, assuming $r-1$ retries

PRR	80%	85%	90%	95%
Sympathy (n=1)	7.97e-3	2.53e-3	4.99e-4	3.12e-5
Sympathy (n=2)	6.36e-5	6.39e-6	2.50e-7	9.77e-10
Memento	1.60e-3	5.06e-4	1.00e-4	6.25e-6
DiMo	1.26e-5	1.28e-6	4.99e-8	1.95e-10

Tab. 8: False positive rates for a node with hop count 5 and 4 transmissions under different packet success rates.

on every hop. A node will therefore generate false positives with a possibility $(1-(1-p_l^r)^d)^n$, where d is the hop count to the sink and n the numbers of heartbeats per sweep.

- With Memento, a node will be reported as failed, if it does not report to its parent successfully during one sweep interval despite sending r heartbeats. This results in a false-positive rate of ${p_l}^r$.
- In DiMo the node is reported missing if it fails to check in at the observer having a probability of p_l^r. In this case, a recovery message is triggered. Consider the case that the recovery message is not kept in the monitoring queue like the node-missing messages, but dropped after r attempts, the false positive rate results in ${p_l}^r(1-(1-{p_l}^r)^d)$.

Table 8 illustrates the false positive rates for the three protocols ranging the PRR between 80% and 95%. For this example the observed node is in a five-hop distance ($d = 5$) from the sink and a number of $r = 4$ attempts for forwarding a message is assumed. Sympathy clearly suffers from a high packet loss, but its performance can be increased greatly by sending two heartbeats every sweep interval ($n = 2$). This however doubles the message load in the network, which is especially substantial as the messages are not aggregated, resulting in a largely increased load and energy consumption for nodes next to the sink. Comparing DiMo with Memento shows the paramount impact of the redundant relay on the false positive rate. DiMo offers a mechanism that is not supported in Sympathy or Memento as it allows sending up to $r-1$ retries for the observer and redundant relay. Due to this redundancy, the message can also be forwarded in the case of a total blackout of one link, a feature both Memento and Sympathy are lacking.

4.3.3 Latency Analysis

For many application designers it is critical to tune the worst-case latency of detecting a missing node. This section analyzes the resulting latency if the same heartbeat frequency is assumed whereas the next section

fp (PRR = 90%)	1e-3	1e-4	1e-5	1e-6	1e-7
Sympathy	4.5	8.5	8.5	8.5	12.5
Memento	72	96	120	144	168
DiMo	5	5	5	5	5

Tab. 9: The latency [min] under the same false positive rate requirement with a heartbeat frequency of $T_S = 4$ min and the number of attempts set to $r = 4$.

compares the heartbeat rates for achieving a failure detection time of $T_D = 5$ min. For a fair comparison, the three protocols should show the same false positive rate fp, which can be estimated according to the analysis in the previous section, assuming a link's packet reception rate of $p_r = 0.9$.

With DiMo the latency of detecting a missing node is given by $T_D = T_S + T_R + T_L$. For example, the guard times $T_L = T_R = 30$ s results in a heartbeat rate of $T_S = 4$ min. For Sympathy, the worst-case latency is given by the sweep interval plus the worst-case latency for sending the message $T_D = nT_S + T_L$, where n is the number of heartbeats every sweep interval. With Memento, the bitmask is reset for every node in every sweep interval. Assuming synchronized sweeps throughout the network, this results in a worst-case latency of $T_D = r(d + 1)T_S$, whereas r denotes the number of heartbeats every sweep interval, d represents the maximum hop count in the network, and the '+1' accounts for the fact, that a node could fail right at the beginning of the observers sweep interval after just having sent a first heartbeat.

The resulting worst-case latencies are detailed in Table 9, allowing up to 3 retries to forward the message before the message is dropped. Memento's latency is largely increased due to its aggregation but also due to the design choice of explicitly not exploiting the possibilities of retransmissions. Sympathy shows a very quick detection time, however, requires sending multiple heartbeats per sweep interval if a small false positive rate is required. DiMo on the other hand achieves its design goal of reporting the missing node within the required 5 min with 3 retries ($r = 4$) for all false positive rates.

4.3.4 Message Overhead

All protocols achieve a given upper bound for the latency if the heartbeat interval is adapted accordingly. For instance, with a heartbeat frequency of $T_S = 4$ min, the latency for Memento results in 120 min, according to the results shown in Table 9. If a latency of 5 min is targeted, the heartbeat interval for Memento needs to be increased by a factor of 24 (i.e., one heartbeat every 10 s). However, at some point the increased message load

fp (PRR = 90%)	1e-3	1e-4	1e-5	1e-6	1e-7
Sympathy[a]	1.11	2.22	2.22	2.22	3.33
Memento	14.4	19.2	24.0	28.8	33.6
DiMo	1.25	1.25	1.25	1.25	1.25

[a]Messages are not aggregated resulting in an enormous routing overhead for nodes close to the sink.

Tab. 10: Number of heartbeats sent per node in each failure detection interval T_D for meeting the latency constraint T_D.

will result in a large amount of interference, which is especially critical for Sympathy's increased load close to the sink and also for Memento's large delay for disseminating the information to the sink that likely results in a very short heartbeat interval.

Table 10 details the effect on the message overhead if a certain maximal false-positive rate should be achieved and a link quality of $p_r = 0.9$ is assumed. The message overhead is denoted as the number of heartbeats that have to be sent every T_D time units at every node if a failed node has to be reported within this time. DiMo and Sympathy both show a very low heartbeat interval, whereas Memento shows a very high data rate and therefore energy consumption. DiMo highlights a constant message rate of only 1.25 messages every T_D, being very close to the theoretical optimum of 1.

Sympathy's low heartbeat rate is misleading, as every single message needs to be forwarded to the sink. Hence the nodes close to the sink have to handle a cumulated load: For instance if in a 100 node network the bottle-neck node close to the sink has to route the heartbeats of 1/5th of the nodes, the bottle-neck node has to handle a 20 times increased heartbeat rate. Memento's aggregation and DiMo's distributed approach both do not suffer from such an increased load towards the sink.

DiMo and Sympathy both include retries in case of a packet failure. Since both the message and the acknowledgement can get lost, the expected number of transmission is given by

$$E[\text{trans}] = (1-p_l)^2 \sum_{k=0}^{k \leq r} p_l^k (k+1) \leq \frac{1}{1/(1-p_l)^2}. \tag{4.4}$$

For example, this results in sending on average an expected number of 1.23 messages per heartbeat assuming $p_l = 0.1$ for both the message or the acknowledgment.

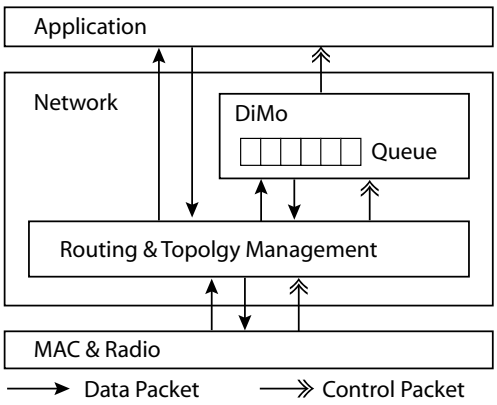


Fig. 27: DiMo is implemented as a part of the network layer, handling all monitoring messages.

4.4 Simulation-based Evaluation

For evaluation purposes DiMo is implemented in Castalia 1.3, a state of the art WSN simulator based on the OMNet++ platform. Castalia allows evaluating DiMo with a realistic wireless channel and radio model but also captures effects like the nodes' clock drift. Castalia's channel model is based on the empirical findings of Zuniga et al. [ZK04]. In particular, it allows modelling the grey area and furthermore there are links with asymmetric behavior, i.e., node A might be able to talk to node B, but not vice versa. Packet collisions are calculated based on the signal to interference ratio (SIR) and the radio model features transition times between the radio's states (e.g., sending after a carrier sense will be delayed). SpeckMAC [WA06], a packet based version of B-MAC, with acknowledgements and a low-power listening interval of 100 ms is used on the link layer. The characteristics of the CC2420 are used to model the radio. For setting up the redundant topology, the NoSE (cf. Chapter 6) initialization scheme is used for finding the neighbors and assessing the link quality. If not otherwise stated, two nodes are neighboring if the assessed link quality is at least 80%.

DiMo is implemented as a module of the network layer on top of the 'traditional' routing engine and topology management as illustrated in Figure 27. A packet is passed to the routing engine from the MAC, it is decided according to a flag in the message header, whether the packet is a monitoring message and needs to be forwarded to DiMo. Furthermore, control messages are sent from the topology management to DiMo in the case of a change in the topology (i.e., change in the set of relays) as discussed in Section 4.2.3 and whether a monitoring message was forwarded successfully. If DiMo receives a monitoring message, that needs to be forwarded to the sink, the message is added to the queue

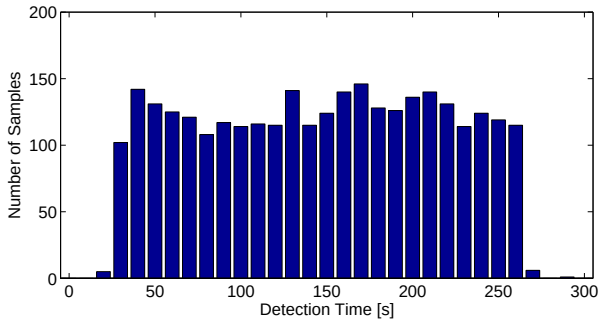


Fig. 28: Failing nodes are reported within the required $T_D = 5\,\text{min}$, showing the expected uniform distribution of the reporting delay.

and then sent back to the routing engine. This detour is made to ensure that monitoring messages do not get lost as detailed in Section 4.2.4. If the node is a sink node, control messages are sent to the application layer notifying about missing nodes and changes in the registration of observer nodes in the network.

The simulations are performed for a network containing 80 nodes, arranged in a grid with a small Gaussian distributed displacement (called jitter), representing an event detection system where nodes are usually not randomly deployed but rather evenly spread over the observed area. 500 different topologies are analyzed by feeding independent random seeds to the wireless channel model and the grid's jitter, resulting in different topologies for every simulation run. The topology management results in a redundantly connected network with up to 5 levels $\mathcal{L}$ and a redundant hop count $\mathcal{R}$ of 6 to 8.

4.4.1 Failure Detection Delay

The fundamental part of the monitoring is the detection of a failed node within the required failure detection time T_D. According to the example of a fire-detection system, regulations require to detect a missing node in $T_D = 5\,\text{min}$, which is taken as the reference to parameterize DiMo. The monitoring time is set to $T_S = 4\,\text{min}$, providing ample time for retries $T_R = 30\,\text{s}$ and to notify about a missing node $T_L = 30\,\text{s}$. The failure detection delay is measured by letting a node run out of energy at a random time and checking for the delay until the sink is notified about the missing node. All nodes but the sink are disabled in 40 different topologies. The result is illustrated in Figure 28 showing the detection delay of the node failures. In particular it highlights that all nodes are detected missing within the required 5 min.

Node failures occur at random times. The failures are therefore uni-

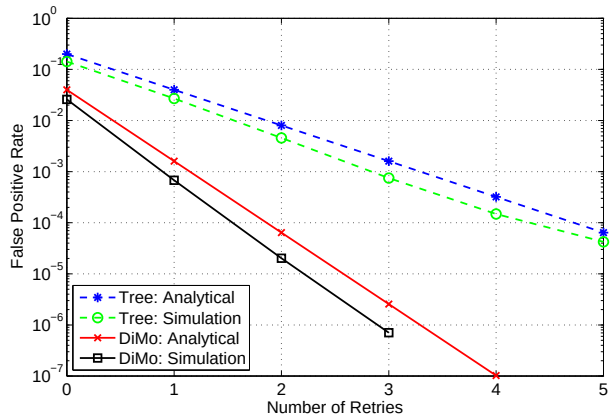
formly distributed during the nodes heartbeat interval T_S. At the observer, the node failure is observed T_R time units later, still following a uniform distribution. Since the delay for reporting the node missing at the sink is similar for all message, the reporting delay at the sink should also follow an uniform distribution. This is confirmed in Figure 28, where most nodes are reported failed in the interval $T_R \leq T_D \leq T_S + T_R$.

However, it is essential to keep in mind that the 5 min detection delay can only be met stochastically, making it important to carefully choose the guard time T_L. This is also indicated in Figure 28, showing that for most situations there is ample guard time, indicated by the gap in the histogram in the last 20 s. There is however also a case the node missing message being delayed, requiring such a long guard time. Furthermore Figure 28 shows that only a few failed nodes are discovered during the first $T_R = 30$ s. This happens if a node fails shortly after the heartbeat timer expires, but still before the heartbeat has been successfully forwarded to the observer.

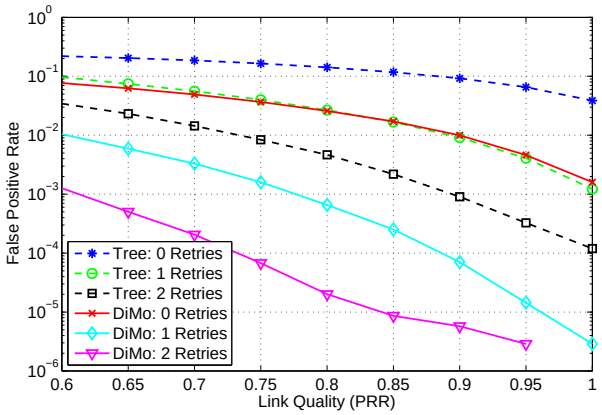
4.4.2 False Positives

The essential part of a safety-critical event monitoring system is to reliably forward a detected event and to timely detect a failed node. However, it is also crucial to prevent false positives, since for every node that is reported missing, a costly on-site inspection is required. A false positive occurs, if the node fails to forward its status message. DiMo uses the same links for monitoring the nodes and for forwarding an event message. A low false-positive rate does therefore indicate a high success rate when forwarding an event.

A false positive is triggered if the node fails to check in, which occurs due to packet losses, mainly induced by random packet failures of the wireless channel. In order to get a detailed view of the false positives, the available link's packet reception rate (PRR) is set to 0.8, which allows to detail the effects of the retransmission scheme. Furthermore, this fixed PRR also allows a comparison with the results of the analytical analysis. This comparison is given in Figure 29(a) and shows the expected number of false positives against the number of retries. Additionally to DiMo, the plot shows the monitoring based on a tree structure (i.e., without DiMo's possibility of sending a recovery message using an alternate relay). This tree structure is comparable with the performance of Memento. The plot highlights the advantage of DiMo's redundancy, however allowing sending twice as many heartbeats than the tree approach. This might not seem necessarily fair at first, however, in a real deployment it is always possible that a link fails completely, allowing DiMo to still forward the heartbeat. The simulation and the analysis exhibit a slight offset in



(a) Varying retries; PRR = 0.8.



(b) Varying link quality.

Fig. 29: False Positives: DiMo achieves the targeted false positive rate of 10^{-7}, also representing the reliability for successfully forwarding an event.

the false-positive rate, which is explained by a simulation artifact of the SpeckMAC implementation that occurs when the receiver's wake-up time coincides with the start time of a packet. This rare case allows receiving not only one but two packets out of the stream, which artificially increases the link quality by about three percent.

The nodes are observed every $T_S = 4$ min, resulting in being monitored $1.3 \cdot 10^5$ times a year. A false positive rate of 10^{-6} would result in having a particular node being wrongly reported failed every 7.7 years. Therefore, for a 77-node network, a false positive rate of 10^{-7} would result in one false alarm a year, being the targeted false-positive threshold for the monitoring system. DiMo achieves this rate by setting the numbers of retries for both the heartbeat and the recovery message to four. Hence the guard time T_R for sending the retries need to be set sufficiently long to accommodate up to ten messages and back-off times.

The impact of the link quality on DiMo's false positive rate is detailed in Figure 29(b). The tree topology shows a similar performance as DiMo, if the same number of messages is sent. But again, it does not show the benefit in the case of a sudden link failure, allowing DiMo to recover immediately. Furthermore it might be surprising that the false positives are not going to zero for a perfect link quality, which is explained by collisions. This is also the reason why DiMo's curve for two retries flattens for higher link qualities. Hence, leaving room for retries is as important as choosing good-quality links.

4.4.3 Load Balancing

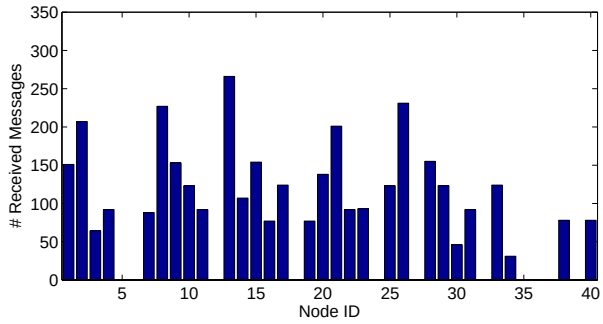
It is quite common in a WSN that certain nodes are well connected and serve as a relay node for many children. This will evidently lead to an increased load and therefore energy consumption as illustrated in Figure 30(a), where each bar represents a single node's received heartbeats in a forty-node network. The nodes without a bar are the ones that are not relays, i.e., the network's leaf nodes. There are distinct differences in the packet load between the different relay nodes. In particular, nodes 2, 8, 13, 20 and 26 show an increased load, which will result in them running out of energy first.

Traditionally, load balancing is tackled by the topology management, where for instance a node will only accept a predefined maximal number of child nodes. With some (hierarchical) topologies, job rotation can be performed for equalizing the node's balance, as proposed with LEACH's [HCB02] rotating clusterheads.

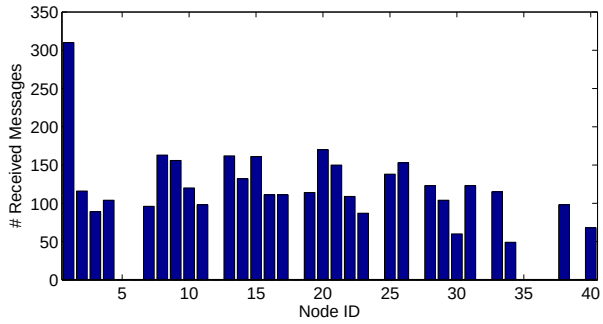
DiMo proposes a complementary approach that balances the load based on the topology provided by the topology management, independently whether being balanced to a certain extent already. First, the relay node provides its children with the current number of observed nodes, representing the relay's current load. Since the balancing of the load is a long term issue, it is not required that every small change in the topology is announced immediately and is rather piggybacked to another message.

Without load balancing, the monitored node is alternating between its relays for checking in and therefore sends the same amount of heartbeats to all of them. On the other hand, if the load balancing is enabled in DiMo, the different relays are burdened inversely proportional to the relays' current load. For instance, if a node has two relays, R_1 having 5 child nodes, and R_2 having 3, then the relative burden on the two relays is (5+3)/5 for R_1 to (5+3)/3 for R_2.

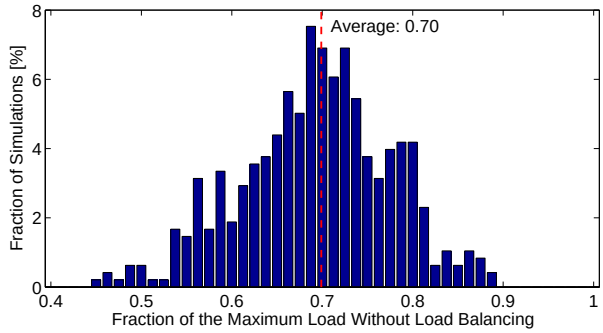
The effect of this simple load balancing is illustrated in Figure 30(b) for the same network and setting than in Figure 30(a). There is a distinct difference between the two plots, showing a well balanced load of the



(a) Without load balancing: Certain nodes (e.g., 8, 13 and 26) show a distinctly increased message load.



(b) With load balancing: The load is more evenly distributed. Since the sink (node 1) is not resource constrained, the load balancing boosts its load.



(c) The load balancing reduces the maximal load in the network by 30 percent on average.

Fig. 30: DiMo's load balancing for a prolonged lifetime.

relays now. The peak of node 1, representing the sink, is an intended feature, boosting the sink's load, usually having a higher energy budget. This is achieved by letting the sink announcing the minimal current load of one, even though the sink usually handles many more child nodes.

The performance gain depends a lot on the given topology. Hence, 500 random topologies are analyzed with and without the load balanc-

ing mechanism enabled. This is illustrated in Figure 30(c), showing a histogram of the gain G achieved through load balancing. To determine the gain G, the load of the node with the maximal load for the balanced solution L_B is compared to the maximal load for the unbalanced solution L_U ($G = L_B/L_U$). On average, this maximum load can be reduced by 30 %, substantially prolonging the lifetimes of the most heavy loaded nodes and therefore of the whole network.

This heuristic approach for the load balancing is not necessarily optimal; however, it fulfills the important criteria of not introducing additional communication overhead and being simple enough to be implemented on an embedded and distributed sensor node. It is left for future work to analyze and compare different balancing strategies, such as a min/max optimization, for a further improved load balancing.

4.5 Summary

This chapter presents DiMo, a distributed algorithm for node and topology monitoring, especially designed for use with event-triggered wireless sensor networks. As a detailed comparative study with two other well-known monitoring algorithm shows, DiMo is the only one to reach the design target of having a maximum error reporting delay of 5 minutes while keeping the false-positive rate and the energy consumption competitive.

The proposed algorithm can easily be implemented and also be enhanced with a topology management mechanism to provide a robust mechanism for WSNs. This enables its use in the area of safety-critical wireless sensor networks. It is detailed in the next chapter, how DiMo can be incorporated into a safety-critical fire-monitoring application. In particular, it also shows an evaluation of DiMo in a testbed as a part of a fire-alarm application.

5

Dwarf: Delay-aWAre Robust Forwarding

In essence, safety-critical event monitoring requires low latency *and* reliable delivery *and* energy-efficiency in a network where links and nodes may (temporarily) fail. The functionality can be decomposed into three tasks: data (alarm) forwarding, node (status) monitoring, and network management. These tasks are not unique in themselves, yet the strict real-time requirements render most existing solutions unsuitable. For example, the popular collection tree protocol (CTP) [FGJ+06] does adapt to failing links, but only within tens of seconds making it of little use for alarm forwarding.

Safety-critical event monitoring does therefore require an integrated system, in which the components (i.e., protocols) are carefully constructed to follow a coherent overall scheme. The system has been developed and evaluated through a four-phase approach, as presented in this chapter. In the first phase, a protocol suite is designed that meets the requirements for safety-critical operation. In particular two main protocols cooperatively enable safety-critical operation: (1) Dwarf (Delay-aWAre Robust Forwarding) allows for a timely delivery of alarm messages, whereas (2) DiMo (Distributed Monitoring) observes the integrity of the nodes and the network topology. In the second phase, simulation is used to verify the validity of the overall design and to define an operating point for the protocol parameters. In the third phase the system is implemented and an initial round of small-scale tabletop experiments is run. These initial tests prompted to adapt various aspects, for example, the MAC

layer was modified to reduce the amount of contention around the sink. In the fourth and final phase, a prototype wireless fire-alarm system is deployed in a typical office environment to assess the performance under realistic conditions. Furthermore, a final set of tabletop experiments is run to determine the behavior in more dense scenarios than the office setting provides. The results show that the integrated protocol suite for safety-critical event monitoring does meet stringent latency and reliability requirements, tolerates large fractions of link failures, yet is energy efficient enough to allow for an operational lifetime of several years.

The remainder of this chapter is organized as follows. Section 5.1 summarizes the requirements and assumptions derived from a real-world fire-alarm system. These requirements set the boundary for the actual alarm-forwarding algorithm, status notification, and network management presented in Section 5.2. The feasibility of this integrated protocol suite is studied in Section 5.3, followed by the implementation details in Section 5.4. The results from evaluating the system in tabletop and real-world experiments are provided in Section 5.5. Finally, Section 5.6 summarizes this chapter.

5.1 Requirements and Assumptions

As a case study, a distributed indoor wireless fire-alarm system is considered. The sensor nodes consist of a micro controller (e.g., TI MSP430), a communication unit (e.g., TI CC1020), a power supply (e.g., 2 AA batteries) and a sensor for detecting fire. There is at least one line-powered sink node that is connected to a central host. Most sensor nodes are expected to be out of communication range to a sink node and therefore require multi-hop networking.

The aim is to comply with real (commercial) systems that meet the currently known regulations and best practices in this domain. In particular there are two domain specific regulations [Eur08] to be considered concerning the latency within the network: Firstly, an alarm raised by a sensor must be reported at the sink **within 10 seconds**, over possible multiple hops. Hence there is only a little margin for hop delays and setting up a route on demand is hardly possible within this short time span. Secondly, failed nodes must be reported **within 5 minutes** at the sink. Typical failures are nodes running out of energy, hardware defects and software errors. Link errors due to environmental changes and interferences can cause a node being unreachable. Such a node must also be considered failed, since a possible alarm cannot be reported anymore.

Batteries are replaced by a qualified technician that has to assert the integrity of the complete alarm system. Since this is a costly operation,

it makes sense to replace the batteries of all nodes as soon as the first one runs out of energy. In order to reduce operational costs, this requires minimizing and equalizing the power consumption for all sensor nodes. Such a complete battery replacement should not occur more often than every three years to be economically reasonable.

5.1.1 Radio and MAC Protocol

Wireless alarm systems often utilize narrow-band radio frequency devices that enable the use of the reserved channels defined in [ERC08]. According to this recommendation, the frequency band from 868.6-868.7 MHz is exclusively to be used for wireless alarm systems. The channel spacing for this band is 25 kHz and hence provides 8 independent alarm channels. Communication on such a channel is limited to a transmission power of 10 mW and a channel-utilization of 1 %.

A low duty-cycle is essential for minimizing the energy consumption, which requires to carefully choose the MAC protocol. Chapter 3 has shown that the class of low-power-listening protocols is well suited to achieve very low duty cycles. In particular, WiseMAC [EHD04] exhibits superior performance and is being used in the protocol suite.

Using WiseMAC, the node wakes up periodically every T_w and checks for activity on the radio channel. If no activity is detected, the node goes right back to sleep. Otherwise the node keeps listening for a potential message. The sending node will therefore pretend each message with a wake-up preamble that is slightly longer than T_w, which ensures that the intended receiver senses an active channel and listens to the complete transmission. In order to minimize this long preamble, WiseMAC learns the receiver's wake-up schedule with every packet being exchanged. This allows the sender to start transmitting a greatly shortened preamble right before the intended receivers wakes up. This does not only save great amounts of energy, but also minimizes the overall activity of the channel. The exact length of the preamble depends on the clock drift and the time passed since the last message was exchanged with the corresponding receiver. In order to use WiseMAC effectively, the application should thus ensure that (important) neighboring nodes are periodically contacted.

WiseMAC does not coordinate the wake-up time with the surrounding nodes. The message can therefore be delayed by T_w at every hop, while waiting for the parent to wake up. The alarm-forwarding algorithm Dwarf presented in the next section takes this into account and minimizes the overall delay with its delay-aware routing scheme. Furthermore it should be noted that the careful staggering of the wake-up periods [LKR04] would allow for shortening the end-to-end latency, yet runs the risk of excessive delays in the case of link errors.

5.1.2 Definitions

The sensor network is represented by the graph $G := (V, E)$ consisting of the set of sensor nodes V and the set of edges E. All communication links are considered to be bidirectional and two nodes $u, v \in V$ can directly communicate with each other (i.e., are neighbors) if and only if $\{u, v\} \in E$. Furthermore, all sensor nodes $u \in V$ are organized according to their distance d_u (hop count) to the nearest sink. More precisely, the neighbors of a node are divided into parents, siblings, and children: The set $N_u^P := \{v : \{u, v\} \in E \wedge d_v = d_u - 1\}$ of neighbors that are closer to the sink are the parents of a node u, the set $N_u^S := \{v : \{u, v\} \in E \wedge d_v = d_u\}$ of neighbors that are at the same distance are the siblings of u, and the set $N_u^C := \{v : \{u, v\} \in E \wedge d_v = d_u + 1\}$ of neighbors that are further away from the sink are the children of u.

5.2 Safety-Critical Protocol Suite

This section presents the protocol suite for enabling safety-critical networking. It introduces the alarm-forwarding algorithm Dwarf and details how the status monitoring of the nodes and the network is performed in cooperation with DiMo. In addition, insights about the neighbor management and the start-up of the network is provided.

5.2.1 Alarm Forwarding

When a node detects an alarm, it assembles an alarm message m and tries to forward it towards the sink as fast as possible. To this end, each node keeps track of the (estimated) wake-up times of its parents and siblings and forwards the message to the k parents and siblings that wake up next. The siblings are only considered in the forwarding process if the message cannot be forwarded to k parents. Neighbors that are known to have already received the alarm (e.g., siblings the message has been received from) are not considered in this process. Forwarding the message to the parent that wakes up next minimizes the local forwarding delay, whereas sending the message to more than one neighbor decreases the probability that the alarm gets lots. More precisely, the forwarding algorithm Dwarf works as described in the following and as detailed in Algorithm 4.

For each newly created or received alarm message m, the alarm message is forwarded k times. For this purpose the node maintains a set of parent C_m^P and sibling C_m^S candidates. These sets contain all the parents and siblings which are assumed to have not yet received the message m (i.e., from which neither an acknowledgment of m nor the message m itself has been received). As long as the parent candidate set is not empty, the

Algorithm 4 Alarm forwarding for node u

```
1:  var H ← ∅
2:
3:  function initCandidates(m)
4:      C_m^P ← N_u^P \ B_m
5:      C_m^S ← N_u^S \ B_m
6:  end function
7:
8:  function getNextHop(m)
9:      if C_m^P = ∅ and C_m^S = ∅ then
10:         initCandidates(m)
11:     end if
12:     if C_m^P ≠ ∅ then
13:         select v ∈ C_m^P that wakes up next
14:         C_m^P ← C_m^P \ {v}
15:         return v
16:     else if C_m^S ≠ ∅ then
17:         select v ∈ C_m^S that wakes up next
18:         C_m^S ← C_m^S \ {v}
19:         return v
20:     else
21:         return ⊥
22:     end if
23: end function
24:
25: function sendAlarm(m)
26:     if r_m < r_a then
27:         r_m ← r_m + 1
28:         v ← getNextHop(m)
29:         if v ≠ ⊥ then
30:             send alarm message m to node v
31:         end if
32:     end if
33: end function
34:
35: upon drop of alarm message m sent to w
36:     sendAlarm(m)
37: end upon
38:
39: function forwardAlarm(m)
40:     H ← H ∪ {m}
41:     initCandidates(m)
42:     r_m ← 0
43:     a_m ← 0
44:     sendAlarm(m)
45: end function
46:
47: upon acknowledgment of alarm m sent to w
48:     if a_m < k − 1 and w is not Sink then
49:         a_m ← a_m + 1
50:         B_m ← B_m ∪ {w}
51:         sendAlarm(m)
52:     end if
53: end upon
54:
55: upon reception of alarm message m from v
56:     if m ∉ H then
57:         B_m ← {v}
58:         forwardAlarm(m)
59:     else if currentAlarm(m) and v ∉ B_m then
60:         B_m ← B_m ∪ {v}
61:     end if
62: end upon
63:
64: upon detection of an alarm
65:     create alarm message m
66:     B_m ← {}
67:     forwardAlarm(m)
68: end upon
```

parent that wakes up next is chosen as the next forwarding destination and removed from the candidate set. If there are no more parents to chose from ($C_m^P = \emptyset$), the sibling that wakes up next is used instead. Once both sets are empty, they are reinitialized with $N_u^P \setminus B_m$ and $N_u^S \setminus B_m$, respectively, where B_m is the set of neighbors that have already received the message.

Although the message has to be forwarded to $k \geq 2$ neighbors, only one alarm message is forwarded to the MAC layer at any given time. Once the node receives the acknowledgment or drop of the message, the next parent (or sibling) is determined and the message is sent again. A node aborts the forwarding process as soon as the message has been successfully forwarded to k neighbors or to a sink node. The forwarding process is also aborted if there are no more parents and siblings to forward the message to (i.e., $(|N_u^P| \cup |N_u^S|) \setminus B_m = \emptyset$). In any case, overall at most $r_a \geq k$ attempts are made for forwarding the message. This implies that an alarm is dropped without forwarding after r_a unsuccessful transmissions.

Upon reception of an alarm message m, a node first verifies that the alarm has not already been forwarded (i.e., m is not in the message history H). New messages are appended to the message history H by adding a

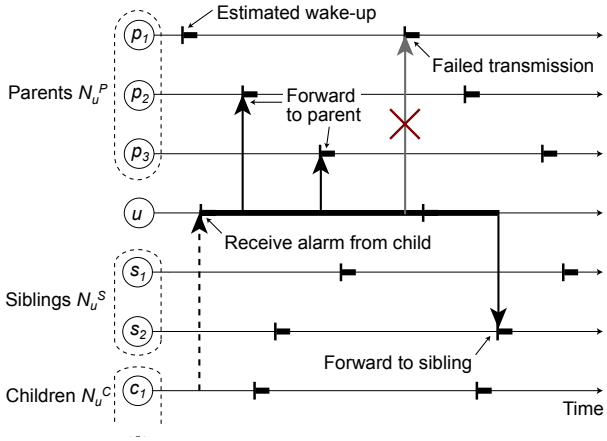


Fig. 31: Dwarf's alarm-forwarding scheme: Node u receives an alarm from its child c_1. Subsequently u forwards the message $k = 3$ times to its parents $N_u^P := \{p_1, p_2, p_3\}$ and siblings $N_u^S := \{s_1, s_2\}$. The nodes are selected according their distance from the sink (parents come first) and their estimated wake-up times.

tuple containing a reference to the alarm originator and the sequence number. The alarm is then forwarded in the same manner as a newly generated alarm.

An example for forwarding an alarm with $k = 3$ is depicted in Figure 31. After node u receives an alarm message from its child c_1, node u forwards the message to the parents p_2, p_3 and p_1 according their estimated wake-up times. After the failed transmission to parent p_1, the parent set is empty and the message is forwarded to the sibling s_2.

5.2.2 Node Monitoring

Node failures have to be detected within a given time bound T_D. In the case for fire-detection-systems regulations demand a maximum delay of $T_D = 5\,\text{min}$. The nodes are monitored locally with an algorithm adopted from the DiMo node-monitoring scheme, which is discussed in detail in Chapter 4 and is illustrated in Figure 26 on page 80. The monitoring scheme is however slightly modified. In particular, there is only one observer for each node at any given time, which reduces the complexity of the node monitoring. In order to observe the remaining links in the network, a neighbor-management scheme is being applied as discussed in the next section.

In essence, every node has a dedicated observer node (usually a parent, alternatively a sibling). The observer should receive a status heartbeat from the observed node in a regular interval of $T_M < T_D$. Should the node

fail to send the heartbeat within T_M since the last one, the observer reports the node missing at the sink using Dwarf with $k = 1$ and $r_a = \infty$ (i.e., the message can be dropped after T_D). The maximum latency T_L (e.g., $T_L = 10\,\text{s}$) for this message, defines the upper bound for the heartbeat interval at the observer of $T_M \leq T_D - T_L$.

Wireless links are susceptible to link failures and hence several retries might be required for sending the status heartbeat to the observer node. The retransmission scheme should be chosen rather conservatively, since a possible simultaneous alarm message must not be jeopardized. Hence ample time T_R is reserved for several retries which requires sending the status heartbeat after $T_S \leq T_M - T_R$ since the last successful transmission. Both the node and its observer reset their timer for T_S and T_M respectively with each successful transmission of the status message.

Should it not be possible to contact the observer despite several retries, the node will get reported missing, despite still running. In order to minimize such *false positives*, the node will select another parent or sibling as its new observer node. The newly chosen observer is sending immediately a message to the sink, reporting that the node has a new observer. Again Dwarf is being used ($k = 1$) for forwarding the notification. Hence the message of the deprecated observer reporting the node missing will be ignored, if the sink receives a message from the new observer.

Observer nodes have to handle an increased data load, decreasing their lifetime. In order to average the energy consumption of the nodes in the network, the observer nodes are chosen according to their current energy budget. Hence the node selects from the set of potential observers the one with the currently highest energy budget left. Due to the applications long-term operation of several years, observers are exchanged in the order of weeks, allowing for a well-balanced energy consumption in the network.

5.2.3 Neighbor Management

Dwarf requires to maintain a table with a node's neighbors with their hop count and wake-up times. Due to the constrained memory and energy resources, the number of entries in the neighbor table is limited to N_{max}.

The wake-up times are susceptible to drift due to clock inaccuracies. This makes it necessary to periodically poll the parents and the siblings in order to update the wake-up schedule. The child nodes on the other hand do not have to be polled, since they themselves poll their parent nodes. Furthermore it should be noted, that the observers are excluded from these regular polls, since they are contacted regularly anyhow.

The chosen polling interval T_i affects the overall energy consumption, which needs to be minimized. On the other hand it also affect the synchro-

nization accuracy since the wake-up preamble increases with T_i. Hence there is an increased likelihood that a parent that is just about to wake up is not considered in the forwarding process since the wake-up time estimation cannot ensure that the parent has not yet already woken up. It is detailed in Section 5.4 that a polling interval of $T_i = 33$ minutes is a well-balanced operating point.

The periodic polling is also required for a continuous monitoring of the link qualities. Hence failing links are detected, and the corresponding node can be evicted from the neighbor table. Should this eviction require to update the node's level, this information is forwarded to all neighbors in the neighbor table.

5.2.4 Startup

The required knowledge of a node u consists only of its level $l(u)$ and its neighbors and their levels. The information regarding level and neighbors can easily be obtained as part of the following startup algorithm:

1. Initially, the level of all nodes but the sinks is set to infinite and they have no information about their neighborhood.

2. The sinks initiate the algorithm by broadcasting their ID and level.

3. A node u receiving a message from a neighbor v with $l(v) < l(u) - 1$ sets its own level to $l(v) + 1$, updates its neighbors and (re)broadcasts its new level.

5.3 Simulation-Based Feasibility Study

As a first evaluation step, the protocol suite is implemented in the GloMoSim [ZBG98] simulation framework and analyzed for its behavior. Only the most essential parts of the protocol suite are implemented. In particular, the network topology is fed to the application at start up and is stable (except for link and node failures) during the tests. Nevertheless, regular packets are sent by the topology- and node-monitoring modules in order to maintain realistic background traffic and to keep the neighbors' wake-up schedule up-to-date.

With the help of the flexible and easy to adapt simulations, the protocols are analyzed in different setups and parameterization. This allows to quickly get better insight into the protocol behavior and to identify an appropriate working point for the subsequent experimental evaluation.

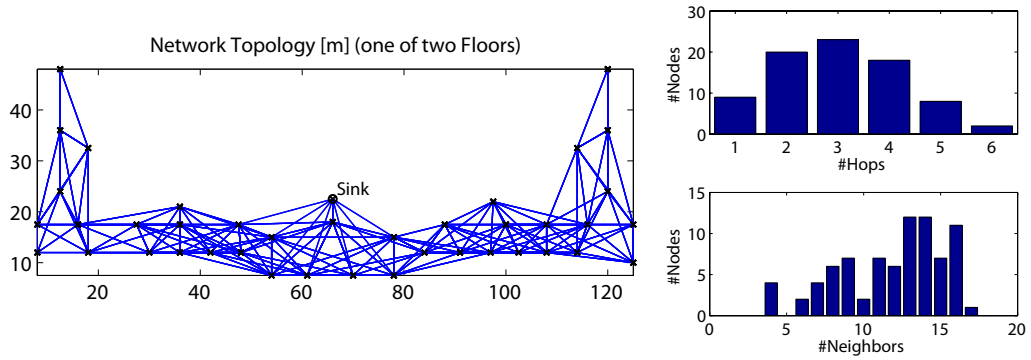


Fig. 32: 80 sensor nodes are positioned according to a real world, but wired deployment. The connectivity is based on measured path-loss coefficients.

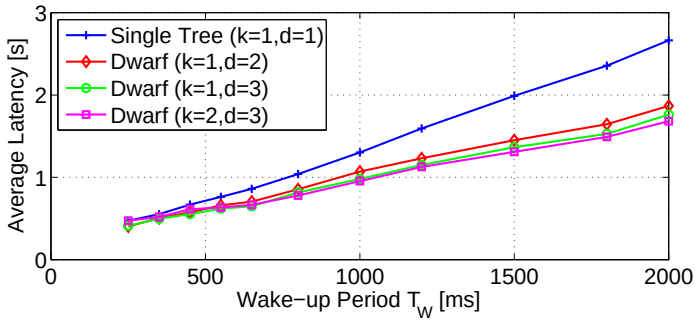
5.3.1 Simulation Setup

In order to simulate the application under realistic conditions, the network topology is based on an existing (wired) real-world alarm system. The sensors are deployed in a large historic public building, having 80 nodes deployed over two floors and one sink in the center of the network (cf. Figure 32). The complete 80x80 path-loss matrix is measured and provides the link qualities for all pair of nodes in the network. GloMoSim is enhanced and allows to be fed with such recorded data. Packet collisions are calculated based on the signal-to-interference-plus-noise ratio. A further enhancement of GloMoSim allows modelling transition times between the radio states. In particular the specifications of the TI CC1020 radio are used to feed this model in order to match the experimental implementation. For the MAC layer, the original WiseMAC [EHD04] implementation is used, but is enhanced by its author with an API providing the neighbors wake-up schedules.

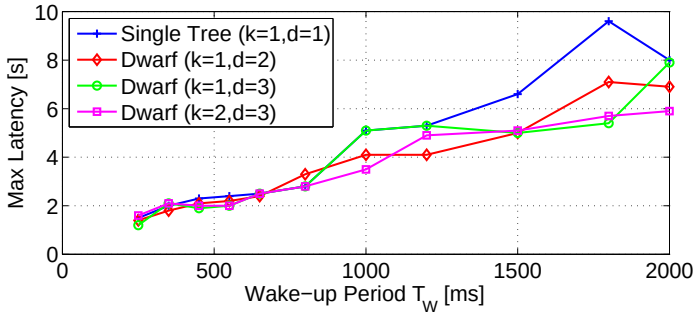
5.3.2 Alarm Latency

The latency for the alarm messages is most crucial and must be limited to $T_L = 10\,\text{s}$. Figure 33 presents the delay for alarm messages under different parameterization. It details the impact of WiseMAC's wake-up period T_w, the impact of the redundancy of the number of neighbors k the alarm message is forwarded to and the redundancy d provided by the network. This latter parameter d denotes the sum of parents and siblings the nodes maintain in their neighbor table. For instance, a single-tree topology is achieved with $d = 1$.

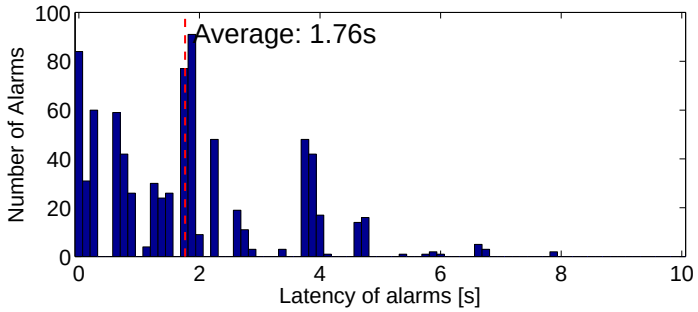
Figure 33(a) presents the average and Figure 33(b) the maximum latency for 800 alarms (for each data point) injected into the network. The most prominent observation is that the latency increases linearly with the



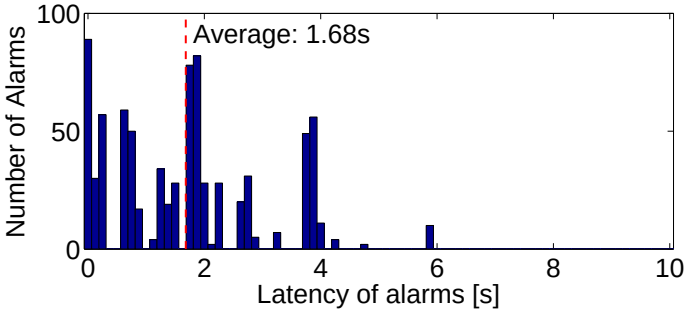
(a) Average latency



(b) Maximum latency



(c) $T_w = 2\,\mathrm{s}, k = 1, d = 3$



(d) $T_w = 2\,\mathrm{s}, k = 2, d = 3$

Fig. 33: Fire-alarm latency in simulation without link failures. Despite 6 hops and a wake-up period of $T_w = 2\,\mathrm{s}$, the alarm is forwarded within the required 10 s.

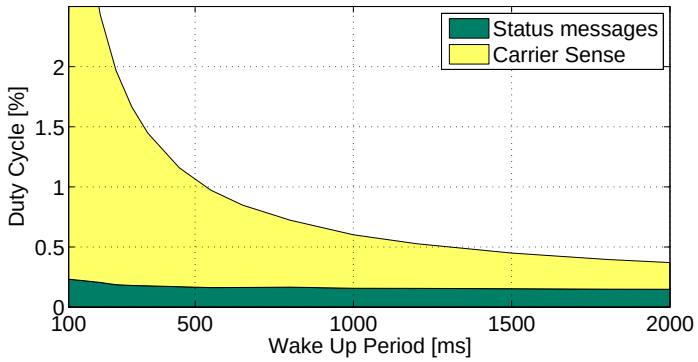
MAC's wake-up period T_w. This can be attributed to the wake-up period in the order of a few hundred milliseconds dominating the message transfer time lasting tens of milliseconds. Even with a wake-up period $T_w = 2\,\text{s}$ and a maximal hop count of 6 in the network, the alarm is always forwarded within the required 10 s. For a parameterization with $k = 2$ and $d = 3$, the maximal latency does not exceed 6 s despite the 6 hops and the $T_w = 2\,\text{s}$. This is a direct consequence of Dwarf's alarm-forwarding scheme that selects the next hop according to its level and wake-up time.

Having a detailed look at Dwarf parameterization (k and d) exhibits that the redundancy in the network provides the highest potential to minimize the alarm-notification delay. For a single-tree topology ($d = 1$), there is only one route to the sink, and therefore the message is delayed at every hop until the parent wakes up. Having redundancy ($d \geq 2$) allows to select the next hop according to the parents relative wake-up times, which clearly reduces the latency. However, it does not make a big difference whether $d = 2$ or $d = 3$ is chosen. The first reason is, that a large fraction of the nodes (36%) do have only one or two parents, and hence not the option to choose between three parents. The second is, that it makes a big difference whether there is only one parent or if there are two (in the former case one might have to wait for a long time for a single parent to wake up). The difference whether there are two or three parents is much smaller. The number of neighbors k the message is forwarded has only a minimal impact on the alarm latency. This means that a first message is usually not "overtaken" by a second message sent on another route. However the maximal latency shows a smoother curve with less outliers if $k > 1$. This observation can also be made comparing Figures 33(c) and 33(d). The main advantage of $k > 1$ is shown in Section 5.3.4, when the network has to deal with random packet losses.

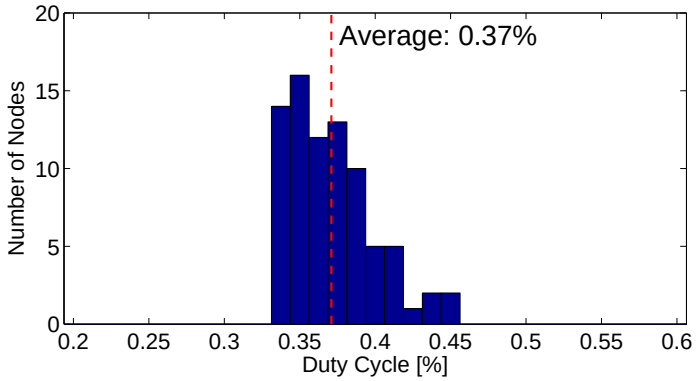
5.3.3 Energy Consumption

A minimal energy consumption is of utmost importance in order to ensure longevity of the network. The radio activity is the main consumer of the energy in the system and is analyzed in detail. There are two main sources in the network that consume energy: (1) WiseMAC requires polling the channel in a regular interval T_w and (2) the node and topology monitoring requires to send periodic status packets. The energy consumption for sending an alarm is neglected, since alarms are assumed to occur scarcely and therefore contribute little to the overall energy budget.

Figure 34(a) depicts the energy consumption of the node in the network (sink excluded) with the highest radio activity. The impact of the two sources for the energy consumption is very distinct, having the regular channel polls as the dominant energy consumer. The radio activity



(a) The energy consumption for the carrier sensing depends greatly on the wake-up period (T_w), while the status messages add an almost constant offset.



(b) The nodes' energy consumption is well-balanced (T_w = 1500 ms).

Fig. 34: Energy consumption in simulation.

for the channel polling increases linearly with the wake-up frequency $1/T_w$. The status messages on the other hand add the expected constant offset. There is however an increase for short wake-up periods, which is explained by a more frequent overhearing of status messages.

The slope of the overall energy consumption flattens for wake-up periods beyond 1 s. Especially for a wake-up period larger than 1.5 s the savings are getting rather small. On the other hand, the expenses for a first rendezvous increase with a prolonged wake-up period. Hence a wake-up period of 1.5 s is decided on for the remaining evaluation.

The status monitoring does not require to forward the messages to the sink and does therefore not show a substantially increased load towards the sink. Figure 34(b) shows the well balanced distribution of the energy consumption of all nodes. Whereas the leave nodes have a radio duty cycle of 0.34%, the maximum is 0.45%.

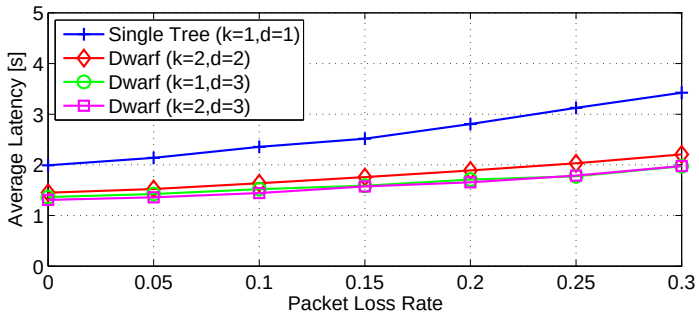
5.3.4 Impact of Link Failures

Wireless communication has to account for possible message losses, inferred from interference from outside the network. This section analyzes the impact of such message losses on the delay and robustness of the alarm notification. In order to simulate this effect, GloMoSim is enhanced with the functionality for random packet drops with a probability p_l on the wireless channel. For the following analysis the wake-up period is set to $T_w = 1.5\,\text{s}$ and the number of attempts for forwarding the message is set to $r_a = 3$.

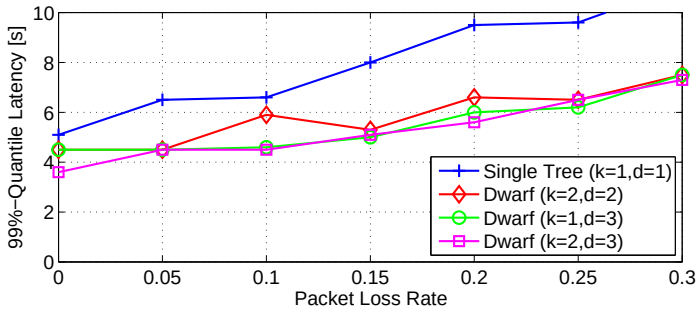
The impact on the message latency over a range of the packet loss rate from 0 to 30% is depicted in Figures 35(a) and 35(b). Instead of the maximum latency, Figure 35(b) presents the 99%-quantile latency. This allows to see the trend of the delay without having to deal with single outliers, distorting the general trend. Especially the single-tree configuration greatly suffers from link failures. This is attributed to the long waiting time of $T_w = 1.5\,\text{s}$ for every lost packet. If the network provides redundancy ($d > 1$), the impact of packet losses is greatly reduced. The average latency with 30% link failures increases only by 0.8 s compared to the case without link failures, whereas the single-tree configuration added 1.4 s. Having a look at the 99%-quantile latency shows that the alarm is forwarded within the required 10 s despite up to 30% link failures, up to 6 hops to the sink and a wake-up period of $T_w = 1.5\,\text{s}$. This is due to reduced waiting time for the next neighbor waking up after a missed packet.

The impact of the number of messages k is much smaller than the one of the redundancy in the network. In particular, the configuration $\{d, k\} = \{2, 2\}$ shows a slightly increased latency compared to the $\{3, 1\}$ case. In general, k has only a minor impact for the latency of the messages, and hence it rarely occurs that the first injected message is overtaken by a subsequent one on a different route. However, k allows to minimize packet drops as illustrated in Figure 35(c). Especially for this increased robustness Dwarf is designed for having multiple messages sent in the network. Furthermore it can be seen that the single-tree configuration even has packet misses if no link failures occur. This is attributed to the periodic traffic induced by the node and network monitoring.

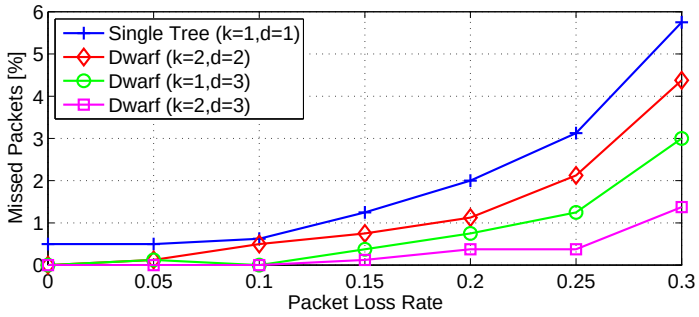
After doing extensive testing of the protocol suite in simulation, there is sufficient confidence into the protocol suite for starting with the implementation. Furthermore, the simulations results allow defining a working point for the implementation. In particular, WiseMAC will be set for wake-up period of $T_w = 1.5\,\text{s}$, Dwarf is parameterized with $k = 2$ and $r_a = 3$ and the size of the neighbor table is limited to $N_{max} = 6$.



(a) Average latency with link failures



(b) 99%-Quantile latency with link failures



(c) Missed alarms with link failures

Fig. 35: Fire-alarm latency in simulation with link failures ($r_a = 3$ and $T_w = 1.5\,\text{s}$). Redundancy minimizes the reporting delay and reduces lost packets.

5.4 Implementation

The implementation discussed in this section emerges from a collaboration with Siemens Building Technologies in Zug.

Wireless alarm systems require narrow-band radio functionality (cf. Section 5.1) and therefore specialized radio hardware. To meet this requirement, a custom sensor node is built as detailed in Table 11. The sensor node is based on the MSP430F148 microprocessor and the narrow-band TI CC1020 radio transceiver. Using a narrow-band instead of a

Param.	Description	Value
RAM	Data memory	2 kB
ROM	Program memory	48 kB
I_{sleep}	Current consumption sleep	5 µA
I_{idle}	Current consumption idle	12.9 mA
I_{RX}	Current consumption RX	23.7 mA
I_{TX}	Current consumption TX	44.0 mA
E_{cs}	Energy consumption carrier sense	1.03 µAs
R	Data rate radio	0.625 kbps
ϑ_{max}	Max. drift variation	10^{-8} s^{-1}

Tab. 11: The sensor node is based on a MSP430F148 microprocessor and a TI CC1020 radio transceiver.

wide-band transceiver such as the widely-used TI CC1000 transceiver with a channel width of 500 kHz has two major implications on the energy consumption: Firstly, the radio's data rate is limited to $R = 0.625$ kbps (CC1000: up to 76.8 kbps) resulting in a greatly increased packet transmission time of $T_{pkt} = 59.2$ ms (for a 23 bytes packet plus 14 byte acknowledgment). Secondly, the radio shows a long phase-locked loop (PLL) time of 2.5 ms (CC1000: 0.2 ms) and thus a long radio-switching time. Both factors increase the overall energy consumption; the price that has to be paid for using the reserved alarm channels with their minimized interference.

On this sensor node, a prototype implementation of the application including a complete wireless communication stack is developed. Furthermore, a customized OS is written, which essentially provides an event and an interrupt handler. The implementation comprises 20189 lines of code (SLOCCount) and has to fit into the minimal memory provided by the system (48 kB of ROM and 2 kB of RAM). The additional modules have the following memory footprint: physical layer 3210 Bytes, WiseMAC 7604 Bytes, topology maintenance 4836 Bytes, Dwarf 2148 Bytes, and DiMo 2594 Bytes. The available RAM is allocated as follows: physical layer 58 Bytes, WiseMAC 264 Bytes, topology maintenance 404 Bytes, Dwarf 26 Bytes, DiMo 122 Bytes, and message buffers 660 Bytes.

5.4.1 Neighbor Management

Synchronization with the neighbors is essential for Dwarf's delay-aware parent selection when forwarding an alarm. As discussed in Section 5.2.3, this requires that the sensor nodes periodically exchange messages with their parents and siblings. These exchanges allow the nodes to keep

track on their neighbors' wake-up schedules and to continuously observe the link qualities. In the following the synchronization interval T_i is determined that minimizes the energy consumption.

The synchronization message is based on two parts: (1) the fixed-size data packet (plus acknowledgment) with transmission time T_{pkt}, and (2) the wake-up preamble T_{pbl} that increases with the synchronization interval T_i. In order to minimize the wake-up preamble, a clock-drift estimation algorithm is added, which assumes a bounded-drift-variation of the quartz crystal. That is, the model assumes that the variation $\vartheta(t)$ of the clock drift $\rho(t)$ is bounded by a constant ϑ_{max}:

$$-\vartheta_{max} \leq \vartheta(t) = \frac{d}{dt}\rho(t) \leq \vartheta_{max}. \tag{5.1}$$

This is a reasonable assumption, as the drift is only gradually influenced by changing conditions such as temperature and battery voltage [RBM05].

The integration over the drift variation results in a preamble length of $T_{pbl} = 2\vartheta_{max}T_i^2$. Assuming that on average half of the wake-up preamble is overheard by the receiver, the overall radio duty cycle for keeping a link synchronized can then be computed as:

$$DC_{link}(T_i) = \left(3\vartheta_{max}T_i^2 + 2T_{pkt}\right)/T_i. \tag{5.2}$$

Hence, the energy consumption of the synchronization packets is minimized if a synchronization interval of $T_i^{opt} = \sqrt{2T_{pkt}/(3\vartheta_{max})} = 33.1$ min is chosen (cf. Figure 36). For the interval $T_i^{opt} = 33.1$ min the preamble has a length of $T_{pbl} = 79$ ms and results in an total transmission time (preamble and packet) of 138 ms. Since the nodes wake up every 1.5 s, this implies a 9.2% probability of overhearing synchronization messages from neighboring nodes, which further adds to the nodes' energy consumption. The energy consumption is further discussed in Section 5.5.

5.4.2 Performance Optimization

In order to get a first impression of Dwarf's implementation, a series of initial tests with a small tabletop setting are conducted. The setting consists of 12 sensor nodes and one sink node, all located within communication range of each other. Each of the 12 nodes is assigned a predefined level in order to ensure that for all tests a similar topology is formed. In particular, 5 nodes are assigned a one-hop, 4 nodes a two-hop, and 3 nodes a three-hop distance from the sink. During the topology set-up, only nodes with the same or an adjacent level are allowed to connect to each other.

The latency and success rate of the fire-alarm messages are tested and measured as follows: After powering on the nodes in the network, the

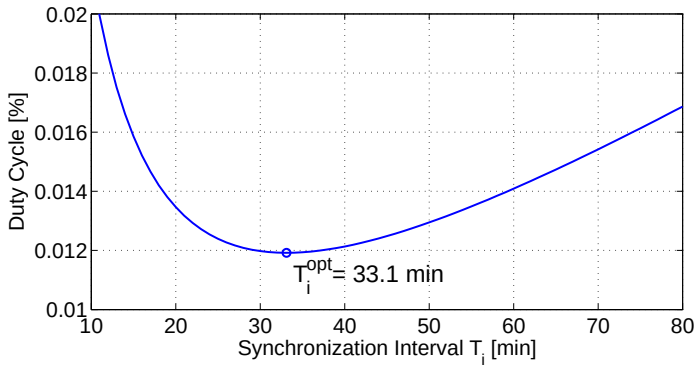


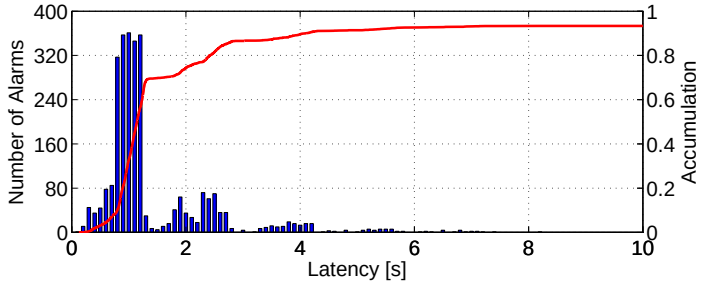
Fig. 36: The energy consumption for keeping a neighbor synchronized is minimized for T_i^{opt} = 33.1 min.

neighbor management is executed without interaction to build a stable topology. Subsequently, fire alarms are triggered every 30 s for all 12 nodes in a round robin fashion. For a single test run, 50 rounds of alarms are triggered. A single test run is repeated 5 times with a fresh start up each time, resulting in a total of 3000 traced fire alarms per experiment.

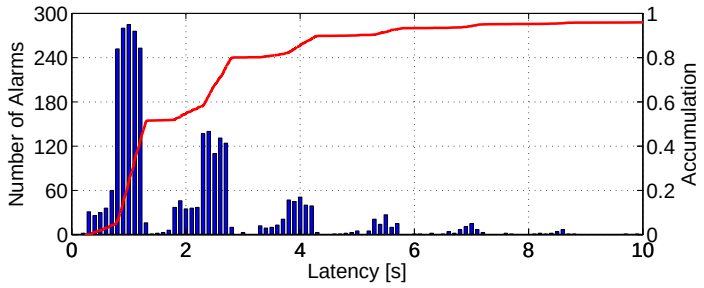
The first results of the implementation showed a rather unsatisfying performance as illustrated in Figure 37(a). Most notably, 6.7% of the alarms were not received at the sink node. Analyzing the message traces identified two problems in the implementation: Firstly, due to the tight synchronization of WiseMAC, collisions occurred despite the carrier detection before each transmission (i.e., the nodes started transmitting a message to the same receiver almost synchronously). Secondly, the sink node seemed to be a bottleneck.

In order to address the collision problem, a so called medium reservation preamble (MRP) is added to WiseMAC. Using the MRP, the nodes start sending the wake-up preamble before the calculated start time using a random (negative) back-off window; the overall preamble is thus prolonged by the MRP. Note that a MRP is only used for the time-critical alarm messages but not for the regular status messages. This prioritizes the alarm messages over the regular status messages while keeping the additional energy consumption low. The impact of the MRP is depicted in Figure 37(b) and shows a decrease of the miss rate from 6.7% to 4.7%.

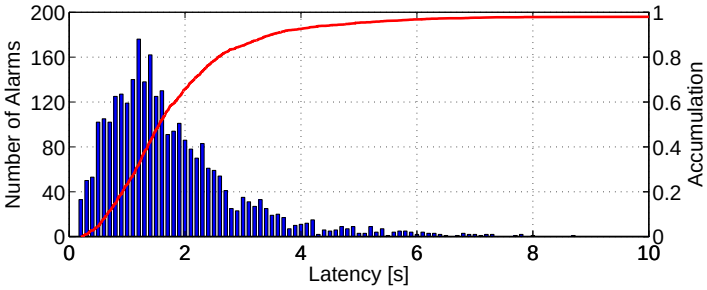
A common way to address the bottleneck at the sink is to have an always listening (line powered) sink. This means that the sink is always on and can receive messages at any time. This approach does however still allows for possible collisions due to hidden terminals, which are especially likely at the sink. Instead the bottleneck is tackled with a so-called slotted sink. Here, the sink divides the wake-up period T_w in N_{max}



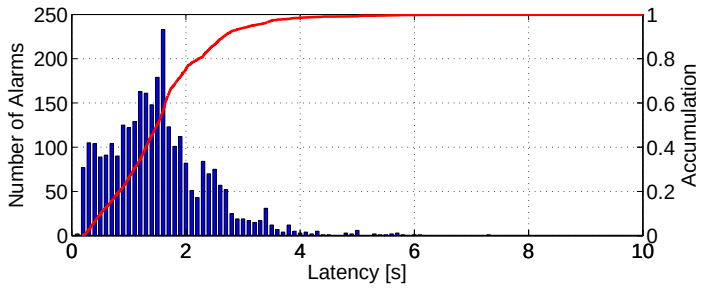
(a) Non slotted sink without MRP: 93.3% success rate.



(b) Non slotted sink with MRP: 95.3% success rate.



(c) Slotted without sink MRP: 97.3% success rate.



(d) Slotted with sink MRP: 99.9% success rate.

Fig. 37: Tabletop experiment with 12 nodes and one sink in a three-hop neighborhood. The slotting of the sink and the medium reservation preamble (MRP) do both increase the packet success rate of Dwarf's alarm forwarding.

slots and wakes up in the middle of each slot. Every connected neighbor (N_{max} at most) gets assigned a dedicated slot for sending a message to the sink. This slotted sink has two advantages: Firstly, the sink node appears like a (set of) regular node(s) with a dedicated wake-up time for each neighbor and no special procedure (i.e., no additional code) is required for contacting the sink. Secondly, every node has a dedicated access time to the sink, which avoids that messages sent to the sink collide. The results of using a slotted sink without the MRP is illustrated in Figure 37(c) and shows a miss rate of 2.7%.

Figure 37(d) shows the impact on Dwarf's alarm forwarding if both schemes are applied. The combination of the MRP and the slotted sink results in a success rate of a 99.9% and only 3 out of 3000 missed alarms. One of the missed alarms was a one-hop alarm, the remaining 2 were triggered at node in a 3-hop distance from the sink.

Having a detailed look at the plots for alarms tested with a non-slotted sink in Figure 37(a) and 37(b), exhibits a burst pattern with a distance of 1.5 s. This is explained by the fact that a failed transmission to the sink will be repeated with a delay of $T_w = 1.5$ s. Even worse, after this delay the transmission may again collide with the (re)transmission to the sink sent by a different node. In the case of the slotted sink approach in Figure 37(c) and 37(d), the retransmission delay is shorter thanks to individual receive slots at the sink for different neighbors and because alarm messages are forwarded over multiple paths. Hence, if the forwarding of an alarm message fails, the same alarm is likely to be forwarded to the sink by a different neighbor before the next retransmission is due after 1.5 s.

5.5 Experimental Evaluation

The refined implementation of the protocol suite with enabled MRP and slotted sink showed very promising results. To gain more insight into the behavior and performance of the protocol suite, the implementation is thoroughly evaluated for the most essential metrics for safety-critical WSNs. First of all, it is of paramount importance that the mandatory real-time demands are met: alarms need to be delivered reliably within 10 s at the sink and failed nodes have to be reported within 5 minutes. Furthermore, one must ensure that the protocols are energy efficient in order to enable a maintenance-free and year-long operation.

In the first evaluation step, a testbed is deployed in a typical office environment. The office environment represents a sparse node setting which, however, is not typical for all fire-alarm deployments. The second common fire-alarm deployment is an open-space scenario, as it can be encountered in factories and airport halls, in which nodes are typically in

line of sight to each other having a high node density. For this reason, the implementation is further tested in a dense tabletop setting.

In both settings all nodes are wired to a central database, collecting performance data from the various tests. It is possible to trigger fire alarms at specific nodes and to trace the path and delay of the alarm message to the sink. The nodes can further be powered off to simulate failure and record the time delay until a node is reported missing.

5.5.1 Office Deployment

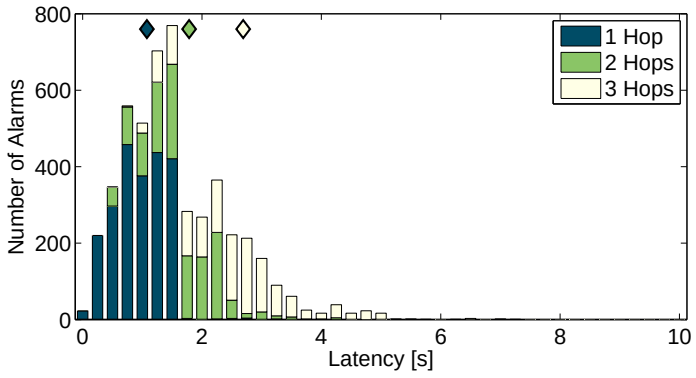
In the distributed office setting, 16 sensor nodes and one sink node are deployed on a floor in an office building, forming a three-hop deep spanning tree to the sink. The nodes are mounted on the ceiling at positions at which wired fire detectors would be placed. Due to the distributed nature of the deployment, the signal quality is degraded by walls and other obstacles. This results in an increased chance for random packet loss, which is counterbalanced by a decreased probability of mutual interference compared to the tabletop deployment with high node density.

5.5.1.1 Alarm Latency

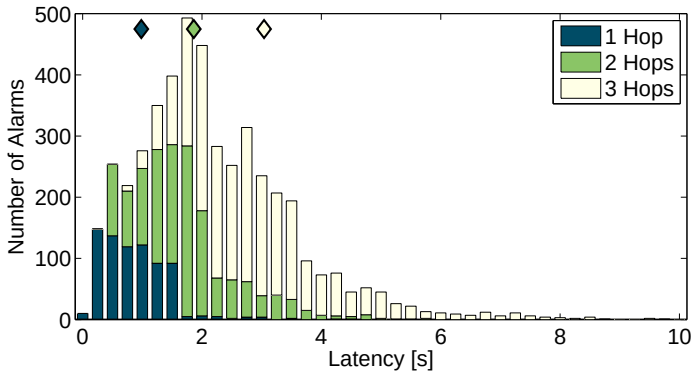
The latency and success rate of the fire-alarm messages are measured in the same way as in the small tabletop scenario (cf. Section 5.4.2). For measuring the alarm delay, 4950 alarm messages are triggered in a round-robin fashion. Altogether, 99.94 % of the messages are successfully delivered to the sink having only 3 messages being lost. Further investigations exhibited that that 2 out of the 3 lost messages are dropped due to a rare error in the implementation (an allocated buffer is overwritten) and cannot be attributed to Dwarf's alarm-forwarding scheme.

The latency distribution of the measured alarms is depicted in Figure 38(a) and divides the alarms according to the minimal hop count to the sink. The message delay clearly depends on the hop count: one-hop nodes deliver their message on average in 1.07 s, two hops in 1.78 s, and three hops in 2.69 s. Notably, three-hop messages take only 0.90 s per hop despite a wake-up period of $T_w = 1.5$ s. Considering that the messages are (1) delayed by the wake-up preamble and in the communication stack, (2) fail every once in a while, and (3) can be blocked by interference, this per-hop delay is very short. It clearly highlights the advantage of the Dwarf alarm-forwarding scheme.

Due to the stringent time bounds on the alarms, the maximal delay of alarm messages is of utmost importance. Figure 38(a) shows that although there is a clear tail in the latency distribution, no message is delayed by more than 7.4 s, which is well within the 10 s requirement.



(a) Office testbed



(b) Tabletop testbed

Fig. 38: Latency of alarm messages on both testbeds. The markers on the top indicate the average latency of the corresponding hops.

5.5.1.2 Status Monitoring

Node failures have to be reported within 300 s, but as status messages are classified as low priority an additional delay of up to 20 s in the backbone network is accounted for. Consequently, the status monitoring protocol is configured to report failed nodes within $T_D = 280$ s at the sink. Assuming a maximum delay of $T_L = 20$ s for reporting missing nodes, the monitoring interval is set at the observer to $T_M = 260$ s. For the monitoring the sending interval is set to $T_S = 240$ s. This leaves $T_R = T_M - T_S = 20$ s for sending several retries or for contacting an alternate parent.

In order to test whether failed nodes are reported on time to the sink, nodes are powered down in a round robin fashion and the delay is measured until they are reported missing at the sink. Afterwards the node is restarted and the network is given time to stabilize before the next failure is triggered. In total 360 node failures are measured as detailed in Figure 39. The maximum delay for detecting a node missing is 271 s,

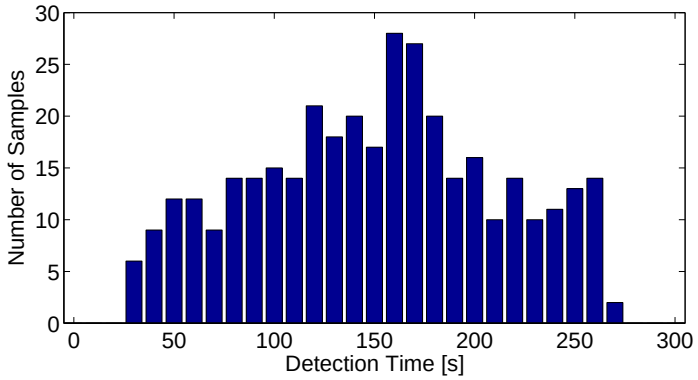


Fig. 39: Status monitoring in office setting: Failed nodes are reported missing at the sink within the required 5 minutes.

which is well within the targeted $T_D = 280\,\text{s}$. It also shown that the status messages from the observer to the sink are delayed up to 11 s, which is longer than the maximum delay for alarm messages (7.4 s). This is a consequence of two effects: First, status messages are sent without a medium reservation preamble and second, to reduce overheads, status messages are sent without any redundancy ($k = 1$) in comparison to alarms ($k = 2$).

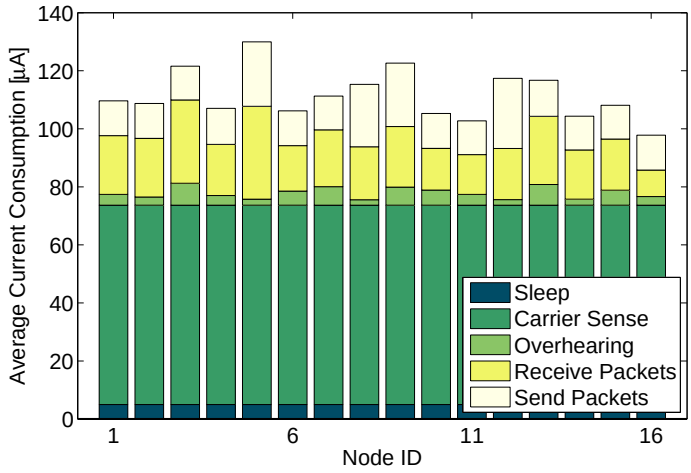
Figure 39 further shows that nodes are not reported missing during the first 20 s. This is a direct consequence of having $T_R = 20\,\text{s}$ for sending possible retries. So even if a node fails just before it is scheduled to contact its observer, the latter one will not report the nodes missing for this time.

Furthermore it should be pointed out that during all monitoring tests, no false positives occurred (i.e., no nodes were erroneously detected as missing). This indicates that the monitoring scheme is robust against random packet loss. However, long-term measurements in the order of several months have to verify these results.

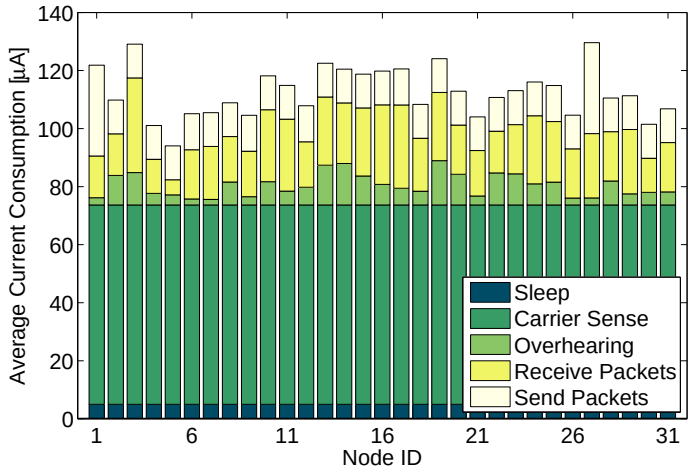
5.5.1.3 Energy Consumption

Alarms are assumed to occur rarely during the lifetime of a network. Energy is thus mainly consumed by the status monitoring and for keeping the network ready to forward an alarm. The average current consumption for this steady state is detailed in Figure 40(a), showing the various sources of energy consumption for the 16 nodes in the office setting.

A node's energy consumption is deduced as follows: First the average energy consumption for the various events in the system (e.g., receiving a message) is measured. Then the number of corresponding events (e.g., number of received messages) is recorded and used to calculate the nodes' overall energy consumption.



(a) Office testbed.



(b) Tabletop testbed.

Fig. 40: Energy consumption without ongoing alarms.

Figure 40(a) shows a well balanced current consumption of up to 130 µA for nodes in the steady state. In particular, they all show the same energy consumption for the regular channel polls. This is due to the fact that all nodes sample the channel with the same polling interval, which adds a constant offset of E_{cs}/T_w to the energy consumption. These regular channel polls are the main cause of the system's energy consumption in the steady state, draining about two-thirds of the energy. This value is rather large given the very low polling interval of $T_w = 1.5\,\text{s}$ and a direct consequence of the narrow-band requirement for alarm messages, which results in the previously discussed long PLL time. These findings match the simulation results as depicted in Figure 34(a).

If a lifetime of three years without maintenance is targeted with an

average energy consumption of 130 µA, a battery capacity of 3400 mAh is required. Hence with two alkaline batteries with a capacity of 2800 mAh each, there is ample energy budget left; this surplus is used for sensor readings, setting up the topology, and forwarding possible alarms.

5.5.2 Tabletop Experiments

For the high-density tabletop experiments, 32 nodes are arranged in a 8 x 4 grid. This arrangement is, on the one hand, a benign scenario due the high signal quality and the absence of obstacles blocking the line of sight. On the other hand, this arrangement also constitutes a worst-case scenario due to the high chance of mutual interference. Similar to the initial tabletop experiments in Section 5.4.2, each node is assigned to a specific level (i.e., distance to the sink): 5 nodes are assigned to level 1 and are allowed to directly communicate with the sink, 10 nodes to level 2, and 16 nodes to level 3. In this dense environment, where all 32 nodes interfere with each other, the alarm latency and the energy consumption are measured and compare the results with those of the office setting. Furthermore the energy demands for sending alarm messages is analyzed.

5.5.2.1 Alarm Latency

For the tabletop experiment, 4609 alarms are triggered in the network. Only one of the 4609 messages is lost and results in a delivery rate of 99.98 %. The maximal delay is with 9.7 s just within the required 10 s.

The latency distribution of the successfully received alarms is depicted in Figure 38(b). Compared to the office experiments, there are two differences: both, the average latency and the tail of the alarms are increased in the tabletop experiments. In particular, the average latency of three-hop messages is 3.03 s compared to the 2.69 s in the office deployment. A closer look at the tail shows that 3.3 % of all 3-hop messages are delayed by more than 6 s in the tabletop experiment as opposed to only 0.5 % in the office setup. The increased delay in the denser deployment is explained by the increased level of interference caused by the network's background traffic for link and node monitoring. Alarm messages are thus delayed if a scheduled transmission is blocked or disturbed. This interference cannot be entirely avoided but is already minimized by the careful orchestration of Dwarf, DiMo and WiseMAC.

In summary, a message delivery rate of more than 99.9 % is achieved in both settings; a delivery rate which most WSN deployments do not achieve, even with end-to-end acknowledgments. However, even with the careful orchestration of the protocol suite, the alarm forwarding remains probabilistic and cannot guarantee that all messages are always

delivered. It is therefore intended to add an end-to-end acknowledgement scheme to ensure that in the rare cases where the deadline cannot be met, alarms are not lost unnoticed.

5.5.2.2 Energy Consumption

The energy measurements in the tabletop setting exhibit a similar energy consumption as the ones in the office setting as highlighted in Figure 40. The major difference is that the level of overhearing is increased in the tabletop setting. However, despite the high number (31) of possible interfering nodes, this increase is not excessive. This can be attributed to the careful design of the node and network monitoring, which minimizes and balances the overall message load. Furthermore, the WiseMAC protocol minimizes overhearing due to its randomly allocated channel access times and the adaptive wake-up preambles.

The maximum channel utilization on the alarm channels is regulated to be less than 1 %. Hence, it must be ensured that none of the nodes exceeds this limit. Node 27 transmitted the most messages (cf. Figure 40(b)), yet occupied the channel for less less than 0.1 %. Consequently, the system is well within the required limit for using the alarm channels.

In the dense tabletop setting, the energy demands for sending alarm messages is measured, even though the contribution to the overall energy consumption is minimal due to alarms being rarely triggered (mainly for testing purposes, not for reporting actual fires). The energy demand for sending 1000 alarm messages is depicted in Figure 41. This number corresponds to sending approximately one fire alarm per day over a period of three years, which is clearly above the rate at which fire alarms are expected to be triggered. For the evaluation, the nodes send 42 alarm messages each in a round robin fashion, resulting in a total of 1302 triggered alarms. The energy consumption for the steady state is subtracted in order to only show the fraction induced by the alarms and the energy consumption is normalized for 1000 alarms. It is worth mentioning that all triggered alarms are received at the sink.

Most remarkably, the 1000 alarm messages reduce the battery capacity by at most 1.76 mAh. This is less than 0.1% of the capacity of a single alkaline AA battery. Hence, neglecting the energy considerations for alarm messages was a reasonable decision. In contrast to the steady state, the energy demand for sending alarms is not well balanced. This is the consequence of having the nodes closer to the sink routing more packets. In particular, the five 1-hop neighbors and some of the 2-hop neighbors show a highly increased energy demand compared to the leave nodes. However, as alarm forwarding only marginally contributes to the overall energy consumption, the imbalance is of no real concern.

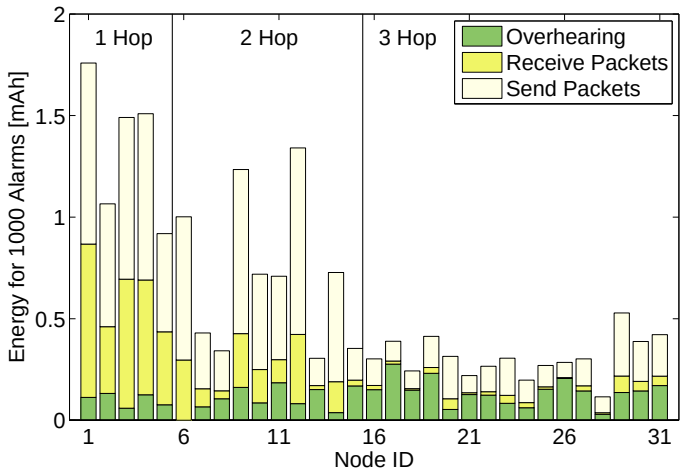


Fig. 41: Energy consumption in the tabletop testbed for sending 1000 alarms.

5.6 Summary

This chapter presents an integrated protocol suite for safety-critical event monitoring, where strict real-time requirements are coupled with a strong demand for reliability, system integrity, and energy efficiency. The protocols are carefully designed and chosen to cooperatively meet these strong demands: (1) the delay-aware alarm-forwarding algorithm Dwarf closely interacts with WiseMAC and allows for a timely delivery of alarm messages. (2) DiMo and the neighbor management allows to monitor the status of the nodes, while keeping the overall traffic to a minimum. (3) This background traffic is well distributed over time, which minimizes collisions and further minimizes contention in the case of an alarm.

The implementation and tuning of the integrated protocol suite for safety-critical event monitoring is guided by initial simulations, and evaluated in a lab setting as well as a realistic office environment. The results show that even in dense deployments where nodes face fierce competition when accessing the wireless channel and consequently overhear a lot of traffic, more than 99.9 % of the alarm messages are delivered in time, *all* failing nodes are promptly reported, and the (extrapolated) network lifetime exceeds 3 years.

Scalability is a major issue when designing a wireless communication stack. The 12-node tabletop testbed shows to build a stable network topology in less than an hour. The 32-node testbed on the other hand requires half a day for building a stable topology and wastes a lot of energy during this time. This motivates the design of NoSE, a dedicated start-up and maintenance scheme. NoSE allows for a fast and energy-efficient start-up of the network as detailed in the next chapter.

6

NoSE: Efficient Maintenance and Initialization

Energy efficiency is of utmost importance for long-term sensor network deployments. Specialized applications and protocols like Dwarf and DiMo are therefore optimized for yearlong operation. However, this usually results in reduced energy efficiency during maintenance tasks and initialization. This chapter proposes the NoSE protocol stack enhancement that enables different *operating modes* for the application: (1) the network can be set back to sleep while the network is being maintained, (2) the network topology can be *initialized* efficiently with respect to time and energy, (3) protocol parameters can be changed at runtime. In particular NoSE is designed for a smooth integration into LPL-based protocol stacks and therefore integrates well into Dwarf and DiMo.

A motivating example analyzes the lifecycle of a safety-critical wireless fire detector network being deployed in a large office setting. At the beginning, nodes need to be installed. This cumbersome task is expected to take several days, possibly interrupted by a weekend. Hence, the first nodes in a network that are being installed and powered on are likely to be without sufficient connectivity for a considerable time. During this time it is essential that the nodes do not search extensively for neighbors not yet installed, wastefully draining the batteries. On the other hand, the responsiveness of starting up the network is important and should not be delayed by a greatly reduced signaling scheme as used in low-power data gathering stacks [BvRW07]. NoSE addresses this task by providing the means for a time and energy-efficient initialization.

During the long lifetime of a sensor network in the order of years,

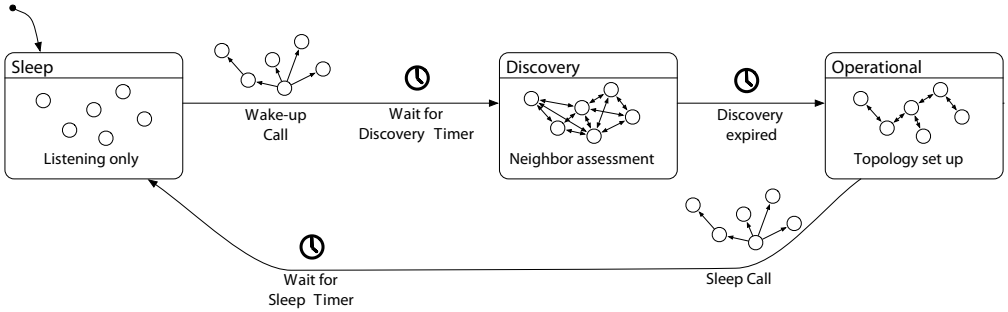


Fig. 42: NoSE state machine: A received network call initiates a timer that triggers the actual state change.

maintenance tasks need to be performed. A common task for a fire-detector network is to replace and to add nodes. This may introduce false alarms due to intermittent connectivity failures. Maintenance can also considerably increase data traffic due to frequent topology changes announced by broadcasting. NoSE provides the functionality for turning off the network temporarily and saving energy. Similar energy savings are possible if the data collected by the WSN is not being used temporarily. In this case the network can be set into deep sleep. When the maintenance task is completed, the network is woken up again and a stable topology is formed.

At the (re)start of the fire-detector network, the maintenance team needs a timely reassurance of a correct system set-up. Are there any partitions? Do nodes need to be relocated for an increased connectivity? Is a certain node completely unavailable? The NoSE protocol enhancement allows to answer such questions in a timely manner. In this context, a detailed link assessment is vital. The low-power radios typically used in a multi-hop deployment result in a large fraction of poorly connected nodes as discussed in Section 2.2. Low-quality links affect the routing protocol especially during the network set-up, where statistical data on the links' performance is not yet available [DHS+07]. The start-up scheme of NoSE (Neighbor Search and Estimation) allows for an exhaustive neighbor search including a detailed link assessment.

This chapter describes NoSE, a protocol enhancement that can be used with the most commonly used WSN protocol stacks. NoSE allows for operational mode changes traversing a defined sequence of states as illustrated in Figure 42. With so called network calls, all nodes in the network can be toggled between an operational and a sleep state. This allows for putting the network in a very energy-efficient sleep mode while a maintenance task is being performed. When (re)starting, NoSE's built-in discovery scheme provides the functionality of an exhaustive neighbor search and link assessment in a short and bounded time. This allows

for a fast and energy-efficient start-up of the WSN and for an immediate feedback about the integrity of the network.

Section 6.1 details maintenance tasks and evaluation metrics. Section 6.2 describes NoSE's network calls and neighbor discovery functionality. Section 6.3 discusses the implementation of NoSE and presents the test setup on a testbed and in simulation. Section 6.4 offers a comprehensive evaluation of NoSE. Section 6.5 presents related work and Section 6.6 concludes this chapter, highlighting the essentials for mode-changes in Wireless Sensor Networks.

6.1 Maintenance and Initialization

NoSE allows for mode changes between the sleep and the operational state. If a maintenance task is to be performed, nodes are set to sleep. This is of particular importance during the initial installation of the nodes, and hence nodes automatically switch to a sleep state when being powered on. During sleep, nodes minimize their energy consumption. As soon as the maintenance task is completed, the nodes are woken up and return to normal operation.

6.1.1 Criteria

NoSE is evaluated on the criteria important for the maintenance and initialization of a WSN:

1. **Energy efficiency versus responsiveness:** Energy consumption is crucial for most sensor network deployments. The maintenance task takes considerable time. During this phase, it is vital that the nodes do not drain a substantial fraction of their battery power. This is particularly likely since the dynamics introduced by the maintenance greatly increase the node's communication overhead and hence its energy consumption. Typically energy consumption and responsiveness are a trade-off: On the one hand, a protocol may spend a lot of energy by aggressively looking for neighbors, which allows for fast topology formation. However, if nodes are not yet ready to participate, this approach is in vain. On the other hand, minimizing radio communication decreases energy consumption, but also responsiveness.

   The time delay for setting the network back to operation is a major concern as a time-bound, distributed assessment of the topology is vital. The responsiveness of sensor nodes needs to be traded off with the requirements on energy efficiency during the deployment

phase. NoSE allows for explicit setting of a suitable trade-off for a given application by adjusting its duty-cycling parameters.

2. **Neighbor discovery and link-quality assessment:** At the beginning of the operational state, responsiveness and energy usage can be improved by providing a well assessed neighborhood allowing for fast topology setup without a large communication overhead. While in sleep mode, a node does not have information about its neighbors. This requires an initial message exchange, which is expensive. Hence, neighbor discovery shall only be performed on a completed deployment in order to avoid repetitive neighborhood searches while more nodes are still joining. At the exit of the sleep state all neighbors should be available. Hence, neighbor discovery is performed after waking up all the nodes in the distinct transitional *discovery* state.

   As Table 13 on page 136 indicates, more than a third of the links available in the network have a packet reception rate of less than 85%. Hence energy is wasted in the operational state if neighbors with a bad link quality are selected, which need to be replaced later on. This necessitates a link-quality assessment, allowing for the selection of high-quality neighbors.

In addition to these evaluation criteria, there is a strict requirement for compatibility. The maintenance and initialization protocol must integrate with a given protocol stack. Although a specialized initialization protocol may benefit from a private second protocol stack, an integration minimizes the resource requirements. NoSE is designed for integration with the prevailing low-power-listening MAC protocols. In particular it can also be used with WiseMAC and therefore complements Dwarf and DiMo. Furthermore, NoSE can also be used with B-MAC and X-MAC available in TinyOS 1.x and 2.x.

In an LPL-based MAC, a node generally has the radio turned off, only switching it on at a regular interval T_P in order to poll the channel for a short time T_{cs}. If a carrier is detected, the node keeps listening, otherwise the radio is switched off immediately. This concept results in a very energy-efficient operation (duty cycle = T_{cs}/T_P), if there is no or little communication in the network. T_P can be tuned for optimized energy consumption if channel utilization is known as discussed in Section 3.4.2.

6.1.2 A Case for a Dedicated Maintenance Protocol

In order to show the need for a protocol stack enhancement, this section investigates the effect of maintenance on a state-of-the-art low-power

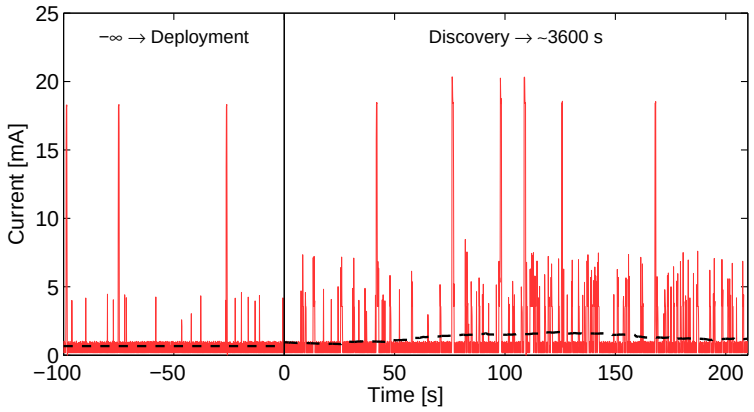


Fig. 43: CTP* does not distinguish between sleep and operation (starting with the discovery). It broadcasts regular beacons during maintenance ($t < 0$ s), even if neighbors are not yet available. The dashed line denotes the average over the last 10 s.

protocol stack. In particular, it looks into the start-up behavior of the TinyOS Collection Tree Protocol (CTP) on top of a LPL MAC protocol is analyzed. The so-called CTP* [LWMB09] has been running efficiently and reliably on a testbed (cf. Section 6.3.3) for months.

CTP* does not feature a dedicated maintenance phase. Hence during the installation of the nodes, repeated neighbor announcements are broadcast long before all nodes are available. This results in an increased current draw of 0.67 mA during the installation of the nodes (cf. Figure 43 for $t < 0$ s). As shown in Section 6.4.1 a dedicated maintenance protocol such as NoSE reduces this amount by 60%.

CTP* is designed and parameterized for an efficient overall operation, at the price of a delayed responsiveness during start. As shown in Figure 44, CTP* shows a lot of parent switches, distributed throughout the network's start up. These switches can mostly be attributed to parent selections with bad link qualities and require a lot of control beacons for announcing the changes in the topology. The impact is two-fold: (1) it takes about an hour to build a stable topology and to conclude about the integrity of the network and (2) each control beacon wastes considerable energy as can be seen in a power trace in Figure 43 for $t \geq 0$ s. Using the NoSE protocol enhancement, the start-up process can be sped up and energy is saved due to the integrated link assessment of the NoSE discovery scheme.

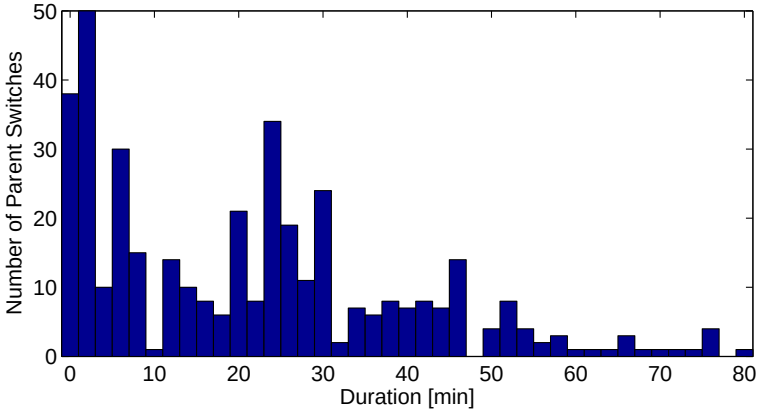


Fig. 44: Dynamics of the network after start-up of CTP* in a 25-node testbed (cf. Section 6.3.3). It takes CTP* about an hour to obtain a stable topology.

6.2 NoSE in Detail

Mode changes in NoSE are supported by the internal state machine depicted in Figure 42. Additional to the sleep and operational states, the transitional discovery state is responsible for the neighbor assessment. A mode change is initiated by *network calls*: a wake-up call in sleep mode initiates an initialization of the network; a sleep call in operational state initiates the transition into a global, low-power-listening mode.

On power-on, a node enters the very energy-efficient sleep state. While sleeping, the nodes only wake up every T_P^{Sleep} for a quick poll of the channel. Since every node in the network follows this paradigm, no message exchange takes place during sleep. The sleep state is introduced in order to save energy after the node is initially powered on and during maintenance. In particular it would be of limited use, to gather neighborhood information when not all surrounding nodes are yet available or nodes are added, removed and relocated.

After installation or when the maintenance task is completed, the network is woken up by a wake-up call originating at a single node, e.g., the sink node. The wake-up call is disseminated by flooding the network. The flood notifies and synchronizes all nodes in the network for the upcoming quasi-synchronous discovery phase by propagating two timer values: (1) the *discovery start timer* triggers the transition to the *discovery* state, (2) the *discovery expiration timer* triggers the exit from discovery (cf. Figure 42). Quasi-synchronous in this respect means that the nodes synchronize on a common time window for the discovery phase, making a rather loose synchronization in the order of a second sufficient.

During discovery, an exhaustive neighbor search and link assessment is performed. All nodes send a predefined number of messages N, making

Algorithm 5 NoSE pseudocode (parameters in Table 12)

```
1: State = StateSleep
2: MACSetPollingInterval(T_P^Sleep)
3:
4: upon ReceivingParameterCall(C)
5:     if OldParameterCall(C.id) then
6:         BroadcastCurrentStatus(H)
7:     else if NewParameterCall(C.id) then
8:         CancelNoseTimers()
9:         BroadcastNewStatus(C)
10:        H = C
11:        SetStateChangeTimer(C.Ts)
12:    end if
13: end upon
14:
15: upon ReceivingStateChangeTimer
16:    MACSetPollingInterval(H.T_P)
17:    if H.type == SleepCall then
18:        State = StateSleep
19:    else if H.type == WakeUpCall then
20:        State = StateDiscovery
21:        ClearNeighborTable()
22:        N = H.N
23:        T_Slot = (H.T_D − H.T_R)/N
24:        SetDiscoveryTimer(0)
25:    end if
26: end upon
27: upon ReceivingDiscoveryTimer
28:    N = N − 1
29:    SendTime = Random(0,T_Slot)
30:    SetDiscoverySendTimer(SendTime)
31:    if N > 0 then
32:        SetDiscoveryTimer(T_Slot)
33:    else
34:        SetDiscoveryEndTimer(T_Slot + H.T_R)
35:    end if
36: end upon
37:
38: upon ReceivingDiscoverySendTimer
39:    BroadcastDiscoveryMsg()
40: end upon
41:
42: upon ReceivingDiscoveryMsg(NodeID)
43:    UpdateNeighborTable(NodeID)
44: end upon
45:
46: upon ReceivingDiscoveryEndTimer
47:    State = StateOperational
48:    MACSetPollingInterval(H.T_P^Op)
49: end upon
```

it beneficial to decrease the channel-polling interval to $T_P^{Disc} \ll T_P^{Sleep}$. This saves considerable energy and shortens the duration of the discovery due to the temporarily increased bandwidth. The discovery phase ends at the same time for all neighbors. It results in complete and well-assessed neighbor information available at every node.

On expiration of the discovery expiration timer, the node enters the operational phase. In particular, the nodes will first set up a network topology based on the well-assessed neighbor information provided by NoSE. For the operational state, the MAC's polling interval is set to T_P^{Op}, which is propagated in the wake-up call. At any time, the operating network can be set back to sleep by flooding a *sleep call* in the network.

The NoSE maintenance scheme is presented in pseudocode in Algorithm 5, not including the parameter call. An overview of all NoSE parameters and suggested setting is provided in Table 12.

6.2.1 Network Calls

Mode changes are initiated by network calls. These calls are triggered at a dedicated node (usually the sink) and flooded in the network. There are three possible network calls: (1) the wake-up call that wakes up the network, (2) the sleep call that puts the nodes back asleep and (3) the optional parameter call, which updates the MAC polling interval. A call comprises all information required for the subsequent mode transition:

Param.	Description	Typical Value
T_S	Remaining time until mode change	60 s
T_D	Duration of mode change (discovery)	2 min
N	Number of discovery packets	30
T_R	Reserve time for a packet queued in the MAC	3 s
T_P^{Sleep}	MAC polling interval during sleep	1.5 s
T_P^{Disc}	MAC polling interval during discovery	50 ms
T_P^{Op}	MAC polling interval when in operation	1.5 s

Tab. 12: Overview of NoSE parameters and suggested setting.

- It contains a call-type identifier, denoting whether the call is a sleep, wake-up or parameter call.
- It contains a call id *C.id*. For every new call initiated by the sink, the id is incremented.
- It contains the countdown-timer T_S, indicating the start time of the mode change, and its duration T_D.
- It includes two polling intervals T_P for the MAC protocol. One is to be applied after the start of the mode change T_S and the other one to be used after the expiration of the mode change $T_S + T_D$.

The calls need to be reliable and fast in order to reach all nodes before the start of the discovery phase. Furthermore it is essential that calls induce only minimal overhead and are easily integrated. NoSE calls are based on a simple flood in the network. If a new call is received for the first time, a message is broadcast to all neighbors. Collisions are minimized by benefiting from the MAC's collision avoidance, i.e., doing a carrier sense before sending. There is no need to use the NoSE specific wake-up call, if the system already provides means for a fast and reliable flood, e.g., Trickle [LPCS04].

A potential problem occurs if a node misses one of the calls flooded in the network. In order to ensure consistency within the nodes in the network, NoSE calls provide a recovery scheme similar to the one of Trickle. Every node stores the most recent call H. Whenever a call C is received containing a newer call ($C.id > H.id$), the call is forwarded and the old call is replaced in the history. Is the received call the node's current call, the call is ignored. If the node however receives a call containing an old id ($C.id < H.id$), the node broadcasts its state H and hence ensures that the outdated node gets synchronized. Similar, if a node A receives a non-NoSE message during sleep from a node B, node A knows that the network is not consistent. It therefore broadcasts its current state

H. Subsequently, node B will get asleep if it had missed the sleep call ($H.id^B < H.id^A$). Otherwise ($H.id^B > H.id^A$) node B will send its current status, which indicates node A should become operational.

6.2.1.1 Wake-up Call

The wake-up call notifies and prepares all nodes in the network for the subsequent discovery phase. All nodes have to know the point in time T_S when the discovery phase begins, its length T_D, the number of packets N being exchanged and the adapted channel-polling interval during discovery T_P^{Disc} and during the subsequent operational state T_P^{Op}.

6.2.1.2 Sleep Call

In the operational state, NoSE allows to set the network back asleep. This is done by flooding the network with a sleep call containing the duty cycle T_P^{Sleep} during the sleep phase and the start time T_S. The sleep phase lasts until receiving a wake-up call and hence has no duration ($T_D = 0$). It is up to the application designer, whether the application should still gather new data and whether potentially buffered messages should be kept in the queue or flushed. However, no messages are forwarded until the next time the operational state is entered.

6.2.1.3 Parameter Call

The parameter call is an optional feature of NoSE. It allows for adapting the MAC's duty cycle. Similar to a wake-up call, the parameter call contains the time T_S and the new duty cycle T_P^{Op*}. The start time T_S ensures that all nodes in the network switch the duty cycle synchronously despite the propagation delay while flooding the call. Furthermore the parameter call allows to define a duration T_D for which the new duty cycle should be applied before returning back to the old duty cycle. If the duration is omitted (i.e., set to zero), the duty cycle is changed to T_P^{Op*} upon further notice. Parameter calls do not trigger a mode state change.

6.2.2 Discovery Phase

The discovery phase starts at the same time for all nodes that have received the wake-up call. During discovery with duration T_D the node sends exactly N broadcast messages containing their node identifier. In parallel to sending the broadcast messages, nodes keep a neighbor entry for all neighbors they receive packets from. In particular, they track the number of received packets and the maximum RSSI. As analyzed in the scenario

without in-network interference in Section 2.4, the number of received packets and the RSSI allows for reliably assessing the link quality.

During the discovery, the network's traffic is increased. This in turn will increase the probability for collisions that occur if broadcasts of different nodes (with length T_P^{Disc}) partially overlap. By limiting the channel utilization C_U, i.e., the fraction in time the channel is busy, the probability for collisions are reduced. The channel utilization C_U depends on the number of neighbors L sending N messages each, the broadcast length T_P and the duration of the discovery T_D:

$$C_U = N(L+1)T_P^{Disc}/T_D. \tag{6.1}$$

It is shown in Section 6.4.4 that a C_U of 0.2 should be chosen as an upper bound for the channel utilization in order to ensure well-assessed link qualities. Collisions are further reduced at the MAC layer, which performs a carrier sense prior to the broadcast. If a carrier is detected, the packet is rescheduled with a short random backoff. Hence a packet may be delayed at the MAC layer for a short time before being transmitted. In order to account for this small possible delay, NoSE reserves a slot of length T_R at the end of the discovery. During this reserved slot, the nodes must not schedule any broadcasts beforehand. This allows the MAC for transmitting broadcasts previously blocked by a positive carrier sense.

The NoSE discovery messages are evenly distributed over the whole discovery phase. This avoids burst failures that are induced due to short-time link failures as they have been commonly observed in Section 2.3.3. Hence, this even distribution of the discovery messages increases the fidelity of the link assessment. For this purpose, the discovery time is partitioned into N subslots. In each of the subslots the node selects independently a random time to send one discovery message.

During discovery, NoSE shortens the channel-polling interval to T_P^{Disc}. According to (6.1) this reduces the channel utilization C_U and therefore allows reducing the discovery time T_D. This shortening of the channel-polling interval increases the responsiveness and also saves sending energy in the order of T_P^{Sleep}/T_P^{Disc}.

6.3 Implementation and Test Setup

6.3.1 Implementation

NoSE has been implemented in TinyOS-2.x on Tmote Sky nodes. It uses the TinyOS-2.x CC2420 radio stack [MHL07] with low power listening (LPL) enabled. NoSE is implemented as an individual layer in the protocol stack. Figure 45 shows NoSE integrated between the MAC and the

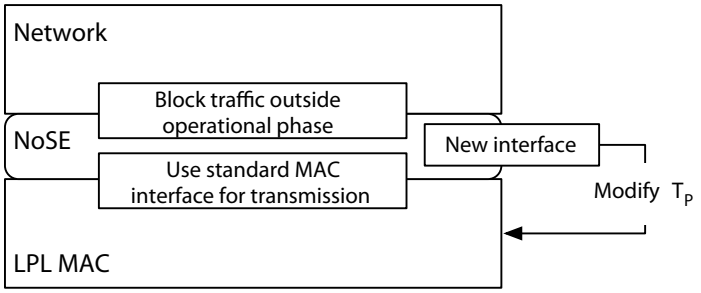


Fig. 45: NoSE stack. An interface allowing for runtime modification of T_P is provided.

Network layer. NoSE uses the standard MAC interface for transmitting its messages. Additionally, an interface to adapt the polling interval T_P on the fly has been added. All NoSE packets (network calls and discovery packets) are identified using a user-defined frame type according to IEEE 802.15.4. Based on a received packet's frame type, NoSE decides whether the packet needs to be internally handled or passed to the network layer.

Internally, NoSE maintains a state machine containing three phases: *sleep*, *discovery* and *operational* (cf. Figure 42). When the node is powered on, it switches into the energy-saving sleep state. As soon as a wake-up call is received, the node sets the discovery start and expiration timers. After the discovery phase, the node is in the operational state.

In order to achieve a better fidelity in the link estimation, Chackeres et al. [CBR02] showed that the discovery messages should have a similar packet size than the data messages being sent during operation. NoSE therefore sends a discovery message with the same length as data messages (e.g., in TinyOS 2.x the standard packet size is 36 bytes). Instead of filling the packet with random data, additional information is added that can be used by the routing protocol, e.g., the node's battery level.

6.3.2 Integration

After the discovery phase, the collected neighbor information has to be transferred to the routing protocol. Thus, it is advisable that NoSE and the routing protocol share a neighbor list.

As an example, the widely used ETX [CABM05] routing scheme is considered, where packets are routed based on the overall link quality of routes. This requires knowledge on each link's packet reception rate, which can be gained directly from the link-assessment information collected during NoSE's discovery phase. This results in a great performance gain compared to the current TinyOS 2.x ETX implementation, where all links are initialized as being perfect (EETX = 0, as of revision 1.4).

6.3.3 Testbed Evaluation

NoSE is designed to improve the effectiveness of real deployments. To this end, NoSE is tested in the realistic environment presented in Section 2.1 (Network $\mathcal{A}$ extended by 7 more nodes). The assumed application scenario is a wireless multi-hop fire-detection network initialized with NoSE. The nodes' neighbor density and location is characteristic for fire-detectors deployed in an office building: 25 nodes deployed over several offices on a single floor. The average neighbor density is 7.4; the maximum density is 12. For the evaluation, 558 test runs were performed. In order to check the quality of NoSE's neighbor search, it is required to have a profound knowledge of the network characteristics, i.e., all neighbors and the according link qualities. This information however is susceptible to change. For this reason, the network characteristics have been measured 18 times over a period of six weeks, alternating with the performance evaluation of the NoSE protocol. For each assessment, every node sent 1000 broadcast messages (with a length in the order of 1 ms) randomly distributed over a period of three hours using CSMA without duty cycling.

Two different metrics are extracted from this reference data, which are used in the evaluation: the *Packet Reception Rate* (PRR) and the *Long Term PRR* (LTPRR). The term PRR reflects a single link assessment being closest in time to the NoSE test. The LTPRR expresses the link performance over the whole six weeks, i.e., over all 18 link measurements. Furthermore, the terminology *High-Quality Links*, refers to links with a (LT)PRR > 95%.

6.3.4 Simulation

In order to show the scalability of the discovery, NoSE has also been simulated in Castalia 1.3, a state-of-the-art WSN simulator based on OMNet++. Castalia provides a realistic wireless channel model that captures the effects of the so called grey area. As with real deployments, this model results in many links that exhibit poor performance. Castalia calculates packet collisions based on the signal-to-interference-plus-noise ratio. Castalia further provides a radio model that features transition times between the radio's states. The radio model specifically uses the characteristics of the CC2420 to match the testbed evaluation.

The simulations use a network containing 160 nodes, arranged in a grid with a small Gaussian-distributed displacement. The network represents an event-detection system where nodes are rather evenly spread. The grid size is varied resulting in node densities ranging from 4 to 45 neighbors. 60 different topologies were analyzed by feeding random seeds to the grid's displacement and the wireless channel model.

6.4 NoSE Evaluation

The metrics used to benchmark NoSE evaluate the major concerns for the maintenance: (1) energy efficiency, (2) responsiveness, (3) integration and (4) the completeness of the neighbor search and the quality of the link assessment. These metrics are evaluated based on the most important maintenance task: the original deployment of the nodes and the subsequent first start of the application. It is essential that the node's energy consumption is minimized during the *deployment*, yet shows a fast and reliable *start up* as soon as all nodes in the network are installed. It should be noted that the analyzed deployment and first start-up show very similar characteristics to other subsequent maintenance tasks.

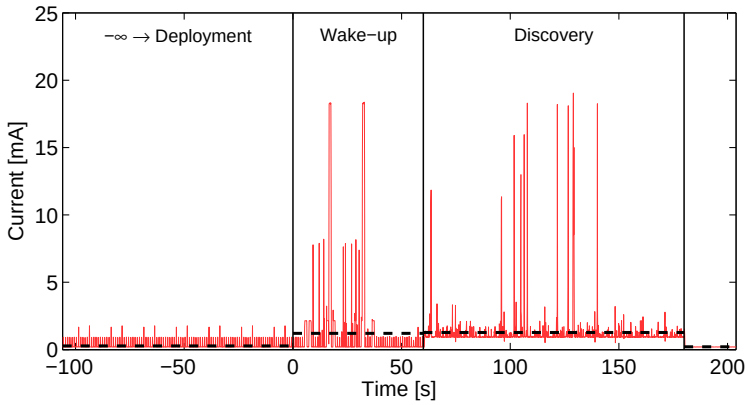
Focusing on the initialization, NoSE can be compared to the Birthday Protocol (BP) [MB01]. While BP neither features operational mode changes nor link assessment during initialization, it can be compared with NoSE with respect to the energy efficiency and the responsiveness during the initialization. BP has been implemented according to the protocol description in [MB01] and evaluated with the suggested parameter allowing to find 95% of the available links.

Concerning integration, NoSE is designed to easily integrate with the predominantly used LPL MAC protocols. BP on the other hand requires a separate, second radio stack for its operation.

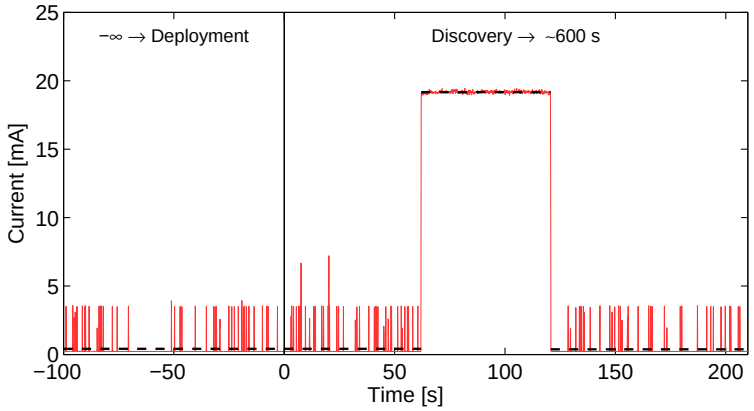
6.4.1 Energy Efficiency During Deployment

The energy consumption is analyzed by measuring the power consumption of a single node. Figure 46 displays the traces of the two protocols' power consumption during the deployment and the start-up phase. Power measurements are taken with an Agilent N6705A power analyzer, with a sample time of 1 ms. For illustration purposes, samples are integrated and plotted over periods of 100 ms each.

During the deployment of the nodes, drastic power savings have a significant impact when the installation time is in the order of days. Figure 46 only presents the last 100 s of the deployment phase, whereas this phase usually takes orders of magnitudes longer for real installations (i.e., days). Nevertheless, longer measurements confirmed that these trace excerpts are representative. NoSE's deployment (sleep) phase shows a very regular and low-power consumption, with an average current drain of 0.28 mA. This is due to NoSE's low duty cycle merely sampling the channel once every second. BP following the same paradigm of listening only, shows an increased current drain of 0.41 mA. This can be attributed to a rather long channel sampling time of 20 ms waiting for a message to be received. Another interesting artifact can be seen around the interval -70



(a) NoSE is energy efficient and finishes within a given time.



(b) BP is less energy efficient and non-deterministic.

Fig. 46: Measured current consumption and responsiveness analysis for NoSE and BP during initialization in the testbed. The dashed line indicates the average energy consumption of the different protocol states.

to -50 s, where BP did not sample the channel for about 20 s. Due to the use of random sampling times, nodes may sporadically stop to listen to the channel for a long time, rendering BP highly non-deterministic.

Figure 47 shows the significant energy savings when using a dedicated maintenance scheme compared to the standard routing protocol CTP*. CTP*'s energy consumption is measured for single node without any neighbors. Hence there is no network traffic biasing the measurement. For the comparison, it should be stressed that both NoSE and CTP* are based on the same LPL MAC protocol. CTP* requires 115 mAh over a period of 7 days and hence 5% of the available energy of a single alkaline AA battery (2200 mAh). The dedicated sleep phase of NoSE allows to reduce this amount by 60%.

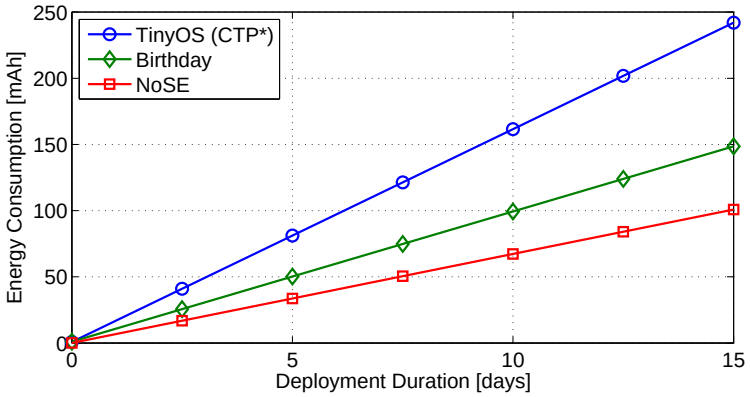


Fig. 47: Extrapolated energy consumption comparison during initialization.

6.4.2 Responsiveness

The second important metric is the responsiveness of the start-up. This is shown in Figure 46, where $t = 0\,\text{s}$ indicates the begin of the start-up. BP's runtime is non deterministic. In particular the start time of the discovery differs for all nodes in the network. As illustrated in Figure 46(b), the node started its discovery at $t = 62\,\text{s}$. Hence the node initiating the discovery at $t = 0\,\text{s}$ had already finished its discovery. NoSE on the other hand features a deterministic start-up time, which is controllable by an application-specific length of wake-up and discovery. Minimizing these two phases directly increases the responsiveness. A detailed discussion about the time requirement for the discovery is provided in the next section.

NoSE's wake-up call is evaluated for reliability and speed. Only in one out of the 558 runs on the testbed, the wake-up call was not received by all nodes. The nodes were always notified within less than 20 s. Nevertheless a pessimistic wake-up period of $T_S = 60\,\text{s}$ is chosen and allows for flexibility in the network size. For the discovery phase, a period of 2 min has shown to be an adequate value, which is further detailed in Section 6.4.4. Overall NoSE provides well-assessed neighborhood information in just 3 minutes starting from the wake-up call.

BP and NoSE both require knowing the end time of the discovery phase, which allows for switching to operational mode, e.g., to set up the routing tables. For BP, this requires estimating the discovery's runtime, which is upper bounded by the network's diameter multiplied by the discovery time. Hence a conservative estimate of the a priori unknown diameter of the network has to be made. NoSE on the other hand features a bounded and deterministic duration of the discovery phase, allowing for a smooth transmission to the subsequent operation.

PRR [%]	≥ 95	$85-95$	$50-85$	<50
#Links	155	45	42	75
NoSE	97.8%	88.8%	80.8%	59.3%
BP	97.0%	91.9%	82.5%	70.4%

Tab. 13: Comparison of the neighbor discovery performance measured in the testbed. Both protocols find almost all high-quality links, but also a substantial number of low-quality links.

6.4.3 Neighbor Discovery: NoSE vs. BP

The initialization schemes of NoSE and BP both provide a neighbor discovery, which is compared in the following. NoSE and BP both aim at finding all available neighbors. Table 13 shows their success rate, each sending 20 discovery broadcasts. The table segments the number of found neighbors according the measured link quality (PRR). Both protocols find almost all of the available high-quality links. The same holds for links with a link quality of 85-95%. However, both protocols also find a substantial number of links with a poor link quality. These links should not be included, emphasizing the necessity of a link assessment prior to setting up of the routing tables.

During discovery, both protocols show an increased activity. Even though discovery time is short, the energy consumption requires considerations: In BP (cf. Figure 46(b) between $t = 62\,\text{s}$ and $t = 122\,\text{s}$) the radio is always turned on, running with a 100% duty cycle. NoSE uses the low-power-listening mechanism of the MAC, allowing for a reduced energy consumption as indicated in Figure 46(a). Despite the increased activity during the discovery, NoSE allows for a reduced duty cycle of less than 20% for the same duration of 1 min as the Birthday's discovery phase, thus being 5 times as efficient.

BP's non-deterministic behavior, the requirement of a second radio stack and the lack of an integrated link-quality estimation show its limited usability when being integrated into a system.

6.4.4 NoSE: Link-Assessment Quality

A unique feature of NoSE is its integrated link assessment. The assessment is based on the knowledge that all nodes in the network send N discovery messages. Based on the number of messages a node receives from a specific neighbor, it assesses the link quality. This assumption implies that discovery messages are lost due to bad links and not due to collisions and requires to limit the channel utilization factor C_U according to (6.1). Subsequently an upper bound for C_U is determined.

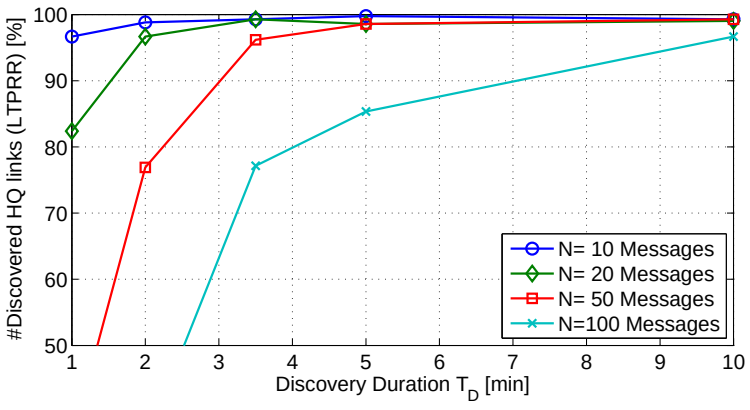


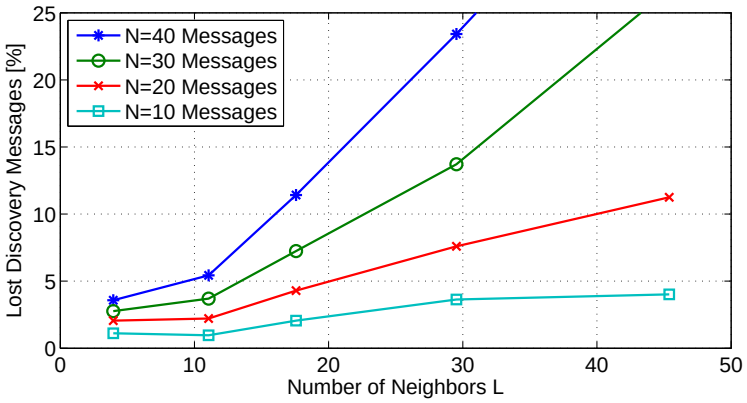
Fig. 48: NoSE discovery phase on testbed: A high channel utilization C_U, i.e., N large and T_D short, jeopardizes the link assessment.

The influence of collisions on the link estimation is analyzed in Figure 48 showing the fraction of high-quality (LTPRR) links, which received at least 90% of the messages. As a rule of thumb, for every 10 messages being sent, the discovery should last an additional minute. For instance, 50 messages require 5 min for finding most LTPRR links. Having a channel-polling interval $T_P^{Disc} = 100\,\text{ms}$ and a maximal number of neighbors of $L = 12$ in the testbed, the maximal channel utilization C_U should be:

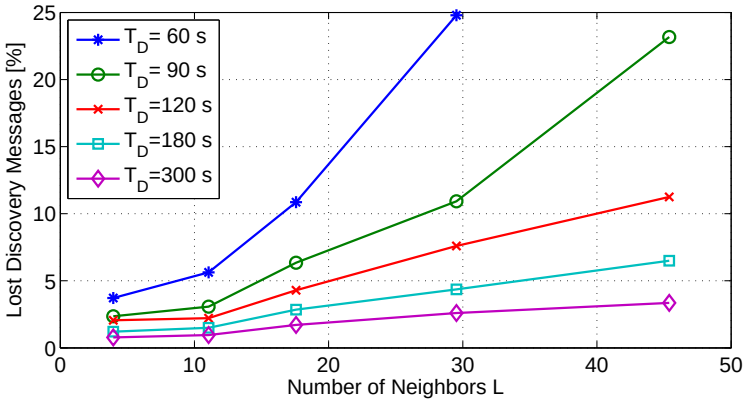
$$C_U \leq N(L+1)T_P^{Disc}/T_D = 10(12+1)0.1/60 \approx 0.2 \tag{6.2}$$

Hence, the discovery duration can be almost arbitrarily reduced by shortening the polling interval T_P^{Disc}. However, this reduction of the discovery phase also shortens channel-assessment time. As shown in Section 2.3.3, the link estimation is susceptible to short-term link fluctuations if the message interval undercuts two seconds. Hence, if the link assessment quality is of paramount importance, the discovery time T_D for this configuration should not fall below $2 \cdot N$ seconds.

In order to show the scalability of NoSE's discovery scheme, the effect of different network densities is simulated. Figure 49 shows the performance of the link assessment in simulation depending on the node density, highlighting the fraction of collisions for high-quality links. Identical to the testbed results, a highly increased bandwidth jeopardizes the link assessment. If the data load is low, e.g., 10 messages in 2 min as seen in Figure 49(a), the number of collisions are limited to about 4%, even with 45 neighbors. More detailed, for a discovery duration of 120 s, Figure 49(b) shows an increased number of lost packets for 17 neighbors compared to 11 neighbors. This is very similar to the implementation results, emphasizing that a channel bandwidth of 20% must not be exceeded during discovery. Thus, the discovery phase can be tuned for an



(a) Fixed discovery duration of $T_D = 2$ min.



(b) Fixed number of $N = 20$ discovery messages.

Fig. 49: NoSE discovery phase in simulation. The parameters need to be adapted to the network density in order to ensure well assessed links ($T_P^{Disc} = 100$ ms).

optimized performance by estimating the deployment's maximal node density.

For parameterizing the discovery, it has to be decided how solid the links should be assessed. Is a link estimation based on $N = 10$ messages sufficient or should rather $N = 50$ messages be sent? The discovery time should then be set to about $T_D = 2 \cdot N$ seconds as discussed above. For setting the appropriate channel polling time, an estimate of the maximal number of neighbors L in the network has to be made and results in $T_P^{Disc} \leq 0.4/(L+1)$ according to (6.2). In the testbed with up to $L = 12$ neighbors this results in a polling interval of $T_P^{Disc} \leq 30$ ms. It should be noted that this polling interval is shorter than the one of the experiments, due to the shorter discovery time (20 s vs. 60 s for $N = 10$).

6.4.5 Long-Term Link Quality

Link assessment provides a snapshot of the link quality during discovery. However the link quality varies over time (cf. Section 2.3.2). This is underlined by an analysis of 40 discovery runs exchanging $N = 20$ packets and assessing a high-quality link if at least 18 of the 20 packets are received. The discovery finds 98.0% of all available long-term high-quality (LTHQ) links. Yet, on the other hand, a substantial 27.3% of the discovered high-quality links are not stable over the long term and exhibit a LTPRR < 85%.

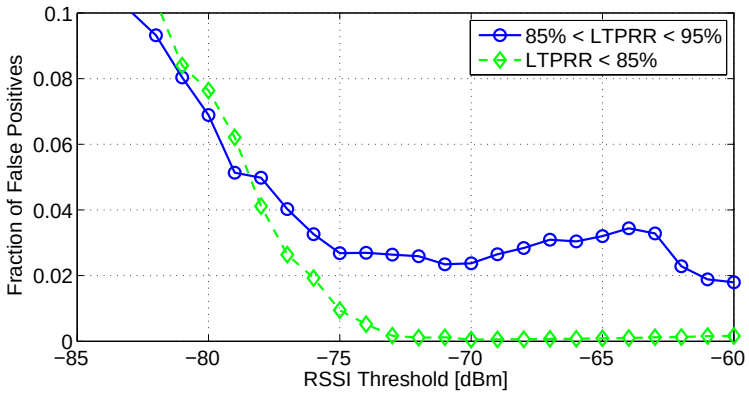
As shown in Section 2.4.3, the RSSI value can increase the accuracy of the link estimation. In the following it is analyzed whether the maximal RSSI value received during the discovery can be used as an additional criteria for detecting LTHQ links: Besides the requirement of receiving at least 19 out of the 20 packets, the maximum RSSI value must exceed a threshold $RSSI_{trs}$. This second requirement greatly influences the fraction of false positives as illustrated in Figure 50(a). The analysis shows that choosing a threshold of $RSSI_{trs} = -73\,\text{dBm}$, allows being almost certain that no bad links (LTPRR $\leq$ 85%) are selected. And there is further a possibility of only 2.6% of selecting a medium link (85% < LTPRR < 95%). However, the RSSI reduces also the fraction of the detected LTHQ links. As shown in Figure 50(b) the previously discussed threshold of $RSSI_{trs} = -73\,\text{dBm}$ will reduce the fraction of detected LTHQ links to only 71.5% compared to the 98.0% without a threshold. Hence a high threshold should only be applied if ample high-quality links can be chosen from.

Trading off the true positives in Figure 50(b) with the false positives in Figure 50(a) is a delicate task and depends a lot on the user's demands. For instance, a threshold of $RSSI_{trs} = -80\,\text{dBm}$ allows finding a good fraction of 92.2% of all LTHQ links. On the other hand, the number of false positives are limited to 7.6% compared to the 27.3% without using the additional RSSI threshold.

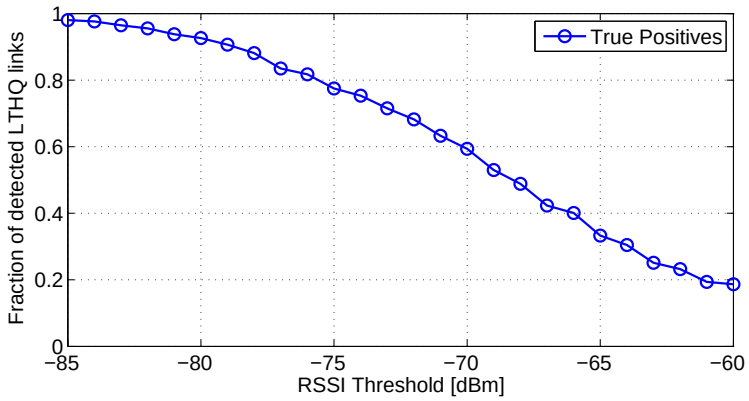
This long-term analysis shows, that the RSSI allows to increase the fraction of detected LTHQ links. This is particularly useful, if ample links are available, and discarding links with a lower RSSI still leaves the network well connected. Another viable option is to rank the links according to their RSSI values and preferably use the high ranked links.

6.5 Related Work

The MAC protocol can greatly influence the energy consumption and responsiveness while maintaining and initializing the network. MAC protocols based on a *global structure* are typically more complex than protocols based on a *random access* scheme.



(a) Having the maximal RSSI as a second criteria minimizes the number of wrongly assessed links.



(b) Choosing a high threshold minimizes the fraction of detected LTHQ links.

Fig. 50: Even though the links show a good quality during the discovery period, many of these links are not stable on the long term. The RSSI value as a second criteria increases the accuracy of detecting long-term high-quality (LTHQ) links.

Protocols that need to maintain a global structure (e.g., slot based [YSH06], TDMA [HH04, HL07]) are likely to show an increased activity while performing maintenance tasks. This is required for keeping the changing topology up to date, i.e., for synchronization and slot arbitration. This requires the nodes to listen intensively and to send numerous neighbor-announcement beacons in order to learn the current channel policy. Moreover, separated clusters can emerge, each having different channel access timings, and taking time to unify. This reduces the system's responsiveness. Hence these globally-structured MAC protocols require a protocol-specific extension for supporting mode changes.

Random-access protocols are commonly based on the *low power listening* (LPL) scheme [HC02] and do not require such an intricate setup.

Protocols, such as B-MAC [PHC04] and WiseMAC [EHD04], do not initiate message transmissions. It is up to the upper layer protocol to decide when the sending of the first message is being initiated. For this class of MAC protocols NoSE is directly applicable.

For maintenance, protocols like Deluge [HC04] and Trickle [LPCS04] provide means for updating the code. Such data-dissemination protocols complement NoSE's operational mode changes ideally. In particular, the parameter call provided by NoSE allows for temporarily decreasing the MAC's polling interval and hence increase the available bandwidth. This allows for a swift and efficient dissemination of a code image.

One of the few protocols that already considers operational mode changes is the Dozer [BvRW07] protocol stack. Dozer provides a combined MAC and network layer which requires to send regular beacons. If the sink node stops sending beacons, the nodes in the network eventually fall asleep. The nodes are gradually woken up again as soon as the sink starts sending beacons again. Since Dozer was not explicitly designed to support mode changes, it misses NoSE's functionality for a fast and time bound (re)start of the network.

Several approaches for initializing WSNs exist. The most prominent solution is the Birthday Protocol (BP) [MB01] by McGlynn and Borbash. As detailed in Section 6.4.3, BP and NoSE are both well suited for discovering neighboring links. BP however, does not feature a deterministic discovery time and requires a separate, second radio stack, making it of limited use when implementing a system.

Kuhn et al. [KMW04] as well as Moscibroda et al. [MvRW06] both suggest setting up a clustered structure for an optimized initialization process. Both these clustered approaches are rather complex and difficult to be implemented on a resource-limited sensor node. Furthermore the clusterhead is less energy efficient than its children, limiting the use for homogeneous networks. In [MvRW06] a second, so-called uniform algorithm is proposed, which is based on an exponentially increasing sending probability. Keshavarzian et al. [KUBHM04] propose different access schemes for combining neighbor discovery and link assessment. Their approach is based on the assumption that all nodes in a network are switched on synchronously, which is impossible to achieve in a real deployment.

Woo et al. [WTC03] investigate and evaluate various estimators for link-quality assessment while running in operational mode. They analyze how a finite, typically quite small neighbor table, providing connectivity and routing information, can be managed. Such an algorithm considerably improves network operation and complements NoSE link assessment performed during initialization.

6.6 Summary

This chapter addresses the issue of maintenance and initialization of WSN deployments. It presents NoSE, a protocol-stack enhancement allowing for mode changes of a network while under operation. NoSE allows switching the network into an energy-efficient sleep state while maintenance is being performed.

NoSE utilizes resources effectively, switching to high-bandwidth phases where necessary, but drastically constraining power consumption otherwise. It manages the trade off between the required responsiveness and an efficient and minimal usage of the energy resources. Furthermore, NoSE offers superior solutions for initialization that outperform standard and specialized approaches known from literature by at least 30% energy savings during maintenance and initialization. A major aspect of NoSE is its bounded operation time, which allows for timely validation of system. Additionally, NoSE's unique integration of link estimation allows for a well-assessed neighborhood. By filtering out mediocre links, NoSE increases the stability and performance of the networking in the operational state.

NoSE is readily usable with LPL MAC protocols commonly used today. In particular it incorporates well into Dwarf and DiMo. Especially during the start-up, great energy saving are possible. This is due to the long channel-polling interval of WiseMAC in conjunction with Dwarf, which makes the initial rendezvous and route updates very expensive. Furthermore, the link assessment during discovery greatly reduces the time to build a stable topology, due to the infrequent neighbor-polling in the order of half an hour. It would therefore require substantial time to gather significant data on a link's quality if no link estimation is performed prior to setting up the topology. Altogether NoSE allows for an energy-efficient maintenance and a swift and dependable initialization.

7

Conclusions

This chapter summarizes the contributions of this thesis and discusses potential directions for future research.

7.1 Contributions

This thesis contributed towards enabling safety-critical wireless sensor networks. It focused on communication aspects and made the following main contributions:

Radio Communication
The wireless communication is known for its erratic behavior. Chapter 2 studied the temporal characteristic of wireless links in an office-like scenario. This is in contrast to most available studies that assume artificial node layouts. It has been shown that total communication losses in the order of a few seconds are much more frequent in real deployments than in these artificial layouts. Furthermore, the novel σ_m metric has been introduced. In contrast to other popular link-quality metrics in WSNs, the σ_m metric analyzes the links' temporal behavior and provides the means for detecting unstable links.

MAC Framework
The analytical MAC framework presented in Chapter 3 provides the first available solution for benchmarking WSN MAC protocols for low data rates. This has not been possible before, due to the large parameter space of the protocols, which does not permit the evaluation in simulation or on a testbed. The framework allows an exhaustive parameter search for

finding the most energy-efficient MAC protocol given a certain data load or latency in the network. The study showed that MAC protocols based on the so called low-power-listening scheme are very energy efficient and well suited for WSNs. In particular WiseMAC proved to be the most suitable candidate for being used in safety-critical applications.

Safety-Critical Protocol Suite
Chapters 4 and 5 proposed the two novel protocols DiMo and Dwarf, which enable in conjunction safety-critical operation of WSNs. A detailed analysis in simulation and in realistic testbeds showed that the tightly integrated protocol stack ensures a timely delivery of more than 99.9 % of the alarm messages. This is a delivery rate most state-of-the-art WSN deployments do not achieve despite their possibility of end-to-end acknowledgements. The protocol suite further proved that failed nodes are promptly reported and that the extrapolated network lifetime exceeds 3 years.

Maintenance and Initialization
Specialized applications and protocols like Dwarf and DiMo are optimized for a yearlong operation. This however results in a reduced energy efficiency during maintenance and initialization due to frequent changes in the topology. Chapter 6 approached this imbalance and suggested the NoSE protocol stack enhancement. NoSE enables different operating modes, which allocates the resources effectively for an energy-efficient maintenance and a swift and dependable initialization. In particular it outperforms standard and specialized approaches known from literature by at least 30% energy savings. Furthermore, the unique link-estimation of NoSE during start-up provides a well assessed neighborhood and therefore a solid basis for the operation.

7.2 Outlook

The research on safety-critical wireless sensor network has disclosed several fields for potential future research:

End-to-End Acknowledgements
It has been shown, that even with a careful orchestration of the protocols, the alarm forwarding remains probabilistic and cannot provide guarantee that all alarms are delivered. The next obvious step is to enhance the alarm forwarding with an end-to-end acknowledgement scheme. Hence in the rare cases where the deadline cannot be met, the alarm will not get lost unnoticed.

Security
This thesis presented a protocol suite that allows safety-critical operation of WSNs. A possible next step is the integration of a security layer in the protocol stack. This would prevent malicious tampering within the network, for instance for raising artificial alarms. Such a secure protocol stack enables the protocol suite to be used in other application domains, in particular for the intrusion detection.

Narrowband vs. Wideband Operation
The implemented protocol suite uses specialized narrowband channels that must only be used for alarm systems. This has the advantage that external interference is greatly minimized, yet comes at the price of increased communication costs due to the low bandwidth and the long radio-switching time. A wideband radio on the other hand provides faster and therefore more energy-efficient operation. This would allow to reduce the wake-up period T_w of the MAC for the same overall communication costs. It would be interesting to analyze whether the increased availability of the nodes counterbalances the raised level of interference. On a similar line, the frequency band (e.g., 433 MHz, 868 MHz, 2.4 GHz) of the operation influences the overall behavior of the system. This raises the question of which band provides for the most reliable and the most energy-efficient operation.

Multichannel and Multiband Operation
Currently the protocol stack operates on a single physical channel, acting as a single point of failure in the case of a (possibly malicious) jammer. This can be avoided by enhancing the protocol stack for multichannel operation. Either a channel-hopping scheme can be used, or the channel is switched on demand. Another viable option is to further enhance to multiband operation, ensuring the operation even if an entire frequency band is jammed.

A

MAC Models – Continued

This appendix details the performance models for the S-MAC, T-MAC, D-MAC, B-MAC, and X-MAC protocols, specifying their synchronization requirements, latency, energy efficiency, and parameter constraints.

A.1 S-MAC

The Sensor-MAC [YHE02] protocol employs a fixed duty cycle for the radio to save energy. This is achieved by synchronizing nodes to a common slot structure of a fixed length T_{Slot}. The slots are divided into a sleep phase T_{sleep} having the radio switched off, and a sync phase T_{sync} and an active phase T_{active} having the radio switched on. This results in a duty cycle of $DC = 1 - (T_{sleep}/T_{slot})$. During the sync phase, nodes broadcast SYNC beacons to keep the network synchronized, i.e., compensating for clock drift. In the active phase the nodes contend for the channel based on a RTS/CTS handshake followed by a DATA and an ACK packet. In order to avoid overhearing, the RTS/CTS control packets include the length of the DATA packet allowing nodes to switch their radios off for the rest of the transfer sequence.

Synchronization. To keep the network synchronized it is assumed that a node has to receive a SYNC beacon from all C neighbors. In order to maximize the efficiency of S-MAC, the sync phase T_{sync} is minimized and fits a single synchronization beacon having the length of a header. The nodes are therefore synchronized every $(C + 1)$ slots and require a clock-drift compensation of $T_{guard} = 2\theta T_{slot}(C + 1)$. This results in a minimal

sync phase time of

$$T_{sync} = T_{guard} + T_{cw} + T_{hdr}. \tag{A.1}$$

Latency. A packet can only be forwarded during the active phase. A newly generated packet is initially delayed by $T_{init} = (T_{sleep} + T_{sync})/2$ on average (assuming a much longer sleep than active phase). In the following active phase, the packet can then be forwarded several hops, whereas the number of hops H_{active} depends on the length of the active phase, the size of the contention window and the time for the message-transfer sequence:

$$H_{active} = \lceil T_{active}/(T_{cw}/2 + T_{msg}) \rceil. \tag{A.2}$$

The message-transfer time $T_{msg} = 4T_{hdr} + P/R$ consists of the control (header-only) packets, namely RTS, CTS and ACK, and the data load with its header.

A packet can only be forwarded H_{active} hops per slot. Therefore $\lfloor h/H_{active} \rfloor$ full slots are required to forward the packet along a h-hop path, while in the last slot the packet is forwarded $\lceil h\%H_{active} \rceil$ slots. Altogether this results in a latency of

$$L(h) = T_{init} + \lfloor h/H_{active} \rfloor \cdot T_{slot} + \lceil h\%H_{active} \rceil \cdot (T_{cw}/2 + T_{msg}). \tag{A.3}$$

Energy Efficiency. The energy efficiency of S-MAC is given by the chosen duty cycle (DC). However, the overhearing avoidance mechanism reduces the duty cycle, especially under heavy load, and has to be accounted for [1]

$$E_{NS} = DC + T_{powerup}/T_{slot} - F_B \cdot (T_{msg} - T_{hdr} - T_{powerup}). \tag{A.4}$$

In addition, S-MAC requires to keep listening during the sleep phase every $T_{discover}$ interval, allowing to receive synchronization beacons of nodes (i.e., discover nodes) that are not synchronized. This results in an overall energy efficiency of

$$E = E_{NS} + T_{sleep}/T_{discover}. \tag{A.5}$$

Parameter Constraints. To avoid hidden terminal collisions, which are especially likely at the sink, the bandwidth at the sink is limited to 25 %. The second constraint is that the active phase needs to accommodate at least one message.

$$F_I^0 \cdot (T_{cw}/2 + T_{msg}) < T_{active}/T_{slot}/4 \tag{A.6}$$

$$T_{active} \geq T_{cw} + T_{msg} \tag{A.7}$$

[1] With the communication model, the overhearing avoidance mechanism for CTS frames cannot be modeled. However, this effect can safely be ignored given that only a minor reduction for RTS frames are observed, because of the focus on worst-case efficiency (nodes on the outside of the network handle little data traffic).

A.2 T-MAC

The Timeout-MAC [vDL03] is an extension to S-MAC. In order to handle traffic fluctuations in time and space T-MAC uses an adaptive duty cycle, implemented as a so called activity timeout $T_a = T_{powerup} + T_{cw} + 2 \cdot T_{hdr}$ that switches off the radio after the last message on the channel.

Synchronization. T-MAC does not have a special sync phase, but sends a synchronization message every T_{sync} in the normal active period, which is guarded for potential clock drift ($T_{guard} = 2\theta T_{sync}$).

Latency. T-MAC does not allow to forward a message more than two hops per active phase, since a node in a three hop neighborhood will switch its radio off preliminary due to the activity timeout. This results in a message latency of

$$L(h) = T_{init} + \lfloor (h-1)/2 \rfloor \cdot T_{slot} + (2 - h\%2) \cdot (T_{cw}/2 + T_{msg}), \tag{A.8}$$

where $T_{init} = T_{slot}/2$ and $T_{msg} = 4 \cdot T_{hdr} + P/R$.

Energy Efficiency. Energy is spent for idle listening (E_{idle}) and for messages that are sent (E_{tx}), received (E_{rx}), and overheard (E_{ovr}). Furthermore, synchronization messages need to be sent and received (E_{sync}), and every discovery interval the channel has to be checked for other nodes.

$$\begin{aligned}
E_{idle} &= (T_{guard} + T_a)/T_{slot} \\
E_{tx} &= F_{out} \cdot (3 \cdot T_{cw}/2 + 2 \cdot T_{hdr} + T_{msg}) \\
E_{rx} &= F_I \cdot (2 \cdot T_{cw}/2 + 2 \cdot T_{hdr} + T_{msg}) \\
E_{ovr} &= F_B \cdot (T_{cw}/2 + T_{hdr}) \\
E_{sync} &= (C+1) \cdot (T_{cw}/2 + T_{hdr})/T_{sync} \\
E &= (E_{idle} + E_{tx} + E_{rx} + E_{ovr} + E_{sync}) + (T_{slot} - T_a)/T_{discover}
\end{aligned} \tag{A.9}$$

Parameter Constraints. Similar to S-MAC, the bandwidth at the sink is at most 25 % in order to avoid hidden terminal collisions.

$$(F_I^0 + (|I^0| + 1)/T_{sync}) \cdot T_{slot} < 1/4 \tag{A.10}$$

A.3 D-MAC

The data-gathering MAC [LKR04] addresses latency overhead for the convergecast (data gathering) communication pattern, by staggering receive and send slots according to the level in the tree. There are N_{sleep} sleep slots between the active receive and send slot, resulting in a frame time of $T_{frame} = (2 + N_{sleep}) \cdot T_{slot}$. D-MAC further uses CSMA with acknowledgments to arbitrate between children, and schedules overflow slots whenever a message is received.

Synchronization. D-MAC needs to send a synchronization message every T_{sync}, if the message rate is too low. There is no dynamically adapted guard time with D-MAC requiring it to guard for a minimal message rate of T_{sync}.

$$\begin{aligned} F_{sync} &= \begin{cases} 0 & \text{if } F_{out} > 1/T_{sync} \\ 1/T_{sync} & \text{otherwise} \end{cases} \\ T_{guard} &= 2\theta T_{sync} \end{aligned}$$

To handle conflicting access to a slot, for example two children sending at the same time to their common parent, D-MAC includes a contention window in every slot, resulting in a slot length of

$$T_{slot} = T_{guard} + T_{cw} + T_{msg}, \quad \text{where } T_{msg} = T_{hdr} + P/R + T_{ack}. \tag{A.11}$$

Latency. The staggered slots allow for fast message forwarding; a message is only initially delayed by $T_{frame}/2$ on average.

$$L(h) = T_{frame}/2 + h \cdot T_{slot} \tag{A.12}$$

Energy Efficiency. D-MAC spends energy listening into the receiving slot (E_{rx}), sending messages (E_{tx}) and listening into an additional slot (for the so called data-prediction scheme) whenever a message is received (E_{dp}).

$$\begin{aligned} E_{rx} &= (T_{powerup} + T_{slot})/T_{frame} \\ E_{tx} &= F_{out} \cdot (T_{cs} + T_{msg}) + F_{sync} \cdot (T_{cs} + T_{hdr}) \\ E_{dp} &= (F_I + |I| \cdot F_{sync}) \cdot (T_{powerup} + T_{slot}) \\ E &= E_{rx} + E_{tx} + E_{dp} \end{aligned} \tag{A.13}$$

Parameter Constraints. To avoid hidden terminal collisions, the sink should receive at most one message in every second slot.

$$(F_I^0 + |I^0| \cdot F_{sync}^1) \cdot T_{frame} < 1/2 \tag{A.14}$$

A.4 B-MAC

In Berkeley MAC [PHC04], nodes periodically check (interval T_w) with a short probe (carrier sense) if the channel is clear, so they can power down immediately. If the channel is busy, the node keeps listening until a start symbol is detected. This reduces the idle-listening overhead at the expense of sending out long preambles (which must be slightly larger than the check interval). By default, B-MAC sends only a data packet after the

preamble, but to be fair to other protocols the optional acknowledgement is included in the model.

Latency. B-MAC allows sending a message right away, but the message transfer time is prolonged by a (long) preamble spanning a complete polling period T_w. B-MAC has an average latency of

$$L(h) = h \cdot (T_{cw}/2 + T_w + T_{msg}), \quad \text{where } T_{msg} = T_{hdr} + P/R + T_{ack}. \tag{A.15}$$

Energy Efficiency. Energy is spent performing regular carrier senses (E_{cs}), sending (E_{tx}), receiving (E_{rx}) and overhearing messages (E_{ovr}).

$$\begin{aligned} E_{cs} &= T_{cs}/T_w \\ E_{tx} &= F_{out} \cdot (T_{cs} + T_w + T_{msg}) \\ E_{rx} &= F_I \cdot (T_w/2 + T_{msg}) \\ E_{ovr} &= F_B \cdot (T_w/2 + T_{hdr}) \\ E &= E_{cs} + E_{tx} + E_{rx} + E_{ovr} \end{aligned} \tag{A.16}$$

Parameter Constraints. In order to avoid hidden terminal collisions at the sink, a maximal bandwidth of 25 % is assumed at the sink.

$$|I^0| \cdot E_{tx}^1 < 1/4 \tag{A.17}$$

A.5 X-MAC

The X-MAC protocol [BYAH06] is a refinement of B-MAC for packet-based radios. Identical to B-MAC, the nodes sample the channel every T_w for a potential packet. The sending node however does not send a long wake-up preamble, but sends a packet strobe instead. The packets in the strobe (with length T_{ps}) contain the receiver's address only and allow overhearing nodes to switch off the radio after receiving a packet out of the strobe. The packets in the preamble strobe are interleaved with short idle times of length T_{al}, in which the sender waits for a so called early acknowledgment, after which the actual message exchange takes place immediately. This early acknowledgement has the benefit that the preamble on average is halved compared to B-MAC, but comes at the price of an increased time for the carrier sense, due to the gaps in the strobe preamble. According to the X-MAC protocol, there is only the early acknowledgement, but no acknowledgment after the data packet is sent. In order to ensure a fair comparison between the protocols, an acknowledgement is added after the packet is sent as done for B-MAC. Furthermore, X-MAC does not provide a functionality for sending broadcasts. However, it can easily be extended following the approach

of B-MAC, i.e., sending a strobe preamble of length T_w, indicating to all receiver to keep listening.

Latency. The latency of X-MAC is similar to that of B-MAC. However the wake-up strobe is cut in half and has an average length of $T_w/2$.

$$L(h) = h \cdot (T_{cw}/2 + T_w/2 + T_{msg}), \quad \text{where } T_{msg} = T_{hdr} + P/R + T_{ack}. \quad \text{(A.18)}$$

Energy Efficiency. Energy is spent performing regular carrier senses (E_{cs}), sending (E_{tx}), receiving (E_{rx}) and overhearing about half (T_{tx}/T_w) of the messages (E_{ovr}). When receiving, the node receives part $(T_{ps} + T_{al})/2$ of the packet strobe before the first strobe packet is received.

$$\begin{aligned}
E_{cs} &= (T_{cs} + T_{al})/T_w \\
T_{tx} &= (\lceil T_w/(T_{ps} + T_{al}) \rceil \cdot (T_{ps} + T_{al})/2 + T_{ack} + T_{msg}) \\
E_{tx} &= F_{out} \cdot (T_{cs} + T_{al} + T_{tx}) \\
E_{rx} &= F_I \cdot (3/2 \cdot T_{ps} + T_{ack} + T_{msg}) \\
E_{ovr} &= F_B \cdot T_{tx}/T_w \cdot 3/2 \cdot T_{ps} \\
E &= E_{cs} + E_{tx} + E_{rx} + E_{ovr} \qquad \text{(A.19)}
\end{aligned}$$

Parameter Constraints. The constraint is the same as for B-MAC (A.17).

B

MAC Models – Special Sink Mode

This appendix explores the optimization of having the sink always listen to the radio. This is the default behavior for Crankshaft and can be easily implemented for B-MAC and WiseMAC.

B.1 Crankshaft*

If Crankshaft does not employ the special sink mode, the sink listens only into one of the unicast slots. This results in the sink being the bottleneck, and hence Equations (3.24) and (3.25) constraining the maximum bandwidth are replaced by

$$F_I^0 \cdot T_{frame} < 1/2. \tag{B.1}$$

B.2 B-MAC*

If B-MAC employs the special sink mode, nodes next to the sink are not required to send a long preamble, which reduces the channel load and further minimizes the energy consumption for sending messages (E_{tx}^*) and overhearing (E_{ovr}^*) with a probability p_{ovr}. The original B-MAC equations for E_{tx} and E_{ovr} in Equation (A.16) therefore change (for nodes next to the sink only) to

$$E_{tx}^1 = F_{out} \cdot (T_{cs} + T_{msg}) \tag{B.2}$$

$$E_{ovr}^1 = E_{ovr}^{\text{B-MAC}}/2 + F_B^1/2 \cdot p_{ovr} \cdot T_{msg}/2, \tag{B.3}$$

where $T_{msg} = T_{hdr} + P/R + T_{ack}$ and $p_{ovr} = T_{msg}/T_w$. For the overhearing it is assumed that half of the overheard nodes are in a one hop distance to the sink. With the special sink mode of B-MAC, the maximal channel load is not necessarily at the sink, but may be at the node next to the sink, or even at a two-hop distance from the sink. Hence Equation (A.17) needs to be generalized to the traffic surrounding any node k in the network

$$\sum_{n \in I^k \cup B^k \cup \{k\}} E^n_{tx} < 1/4. \tag{B.4}$$

This ensures that the channel load is not exceeding 25 % anywhere in the network.

B.3 WiseMAC*

If the sink node is always listening, the resulting energy savings are similar to the one of B-MAC, i.e., the preamble size for nodes next to the sink is minimized and the bottleneck might be shifted. Thus nodes next to the sink will incur the same costs for sending as B-MAC*, and observe a similar reduction in costs for overhearing.

$$E^1_{tx} = F^1_{out} \cdot (T_{cs} + T_{msg}) \tag{B.5}$$

$$E^1_{ovr} = E^{\text{WiseMAC}}_{ovr}/2 + F^1_B/2 \cdot T_{msg}/T_w \cdot T_{msg}/2 \tag{B.6}$$

One of the original parameter constraints of WiseMAC, Equation (3.17), must be adapted to accommodate the new access policy around the sink. The bottleneck is now either at the nodes next to the sink, having a limited number of slots to receive messages (B.7), or (equivalent to B-MAC*) at the channel (B.4).

$$(F^1_I + |I^1| \cdot F^2_{sync}) \cdot T_w < 1/2 \tag{B.7}$$

Bibliography

[Ban96] Anindo Banerjea. Simulation study of the capacity effects of dispersity routing for fault tolerant realtime channels. In *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, pages 194–205, New York, NY, USA, 1996. ACM.

[BBDM05] J. Beutel, P. Blum, M. Dyer, and C. Moser. *BTnode Programming - An Introduction to BTnut Applications*. Computer Engineering and Networks Lab, ETH Zürich, Switzerland, 1.0 edition, May 2005.

[BE02] David Braginsky and Deborah Estrin. Rumor routing algorthim for sensor networks. In *1st ACM Int'l Workshop on Wireless Sensor Networks and Application (WSNA 2002)*, pages 22–31, New York, NY, USA, 2002. ACM.

[Beu06] Jan Beutel. Metrics for sensor network platforms. In *Proc. 2nd Workshop on Real-World Wireless Sensor Networks (REALWSN '06)*, pages 26–30. ACM Press, New York, June 2006.

[BG03] M. Beigl and H. Gellersen. Smart-Its: An embedded platform for smart objects. In *Proc. Smart Objects Conference (SOC 2003)*, Grenoble, France, May 2003.

[BGH+09] Jan Beutel, Stephan Gruber, Andreas Hasler, Roman Lim, Andreas Meier, Christian Plessl, Igor Talzi, Lothar Thiele, Christian Tschudin, Matthias Woehrle, and Mustafa Yuecel. PermaDAQ: A scientific instrument for precision sensing and data recovery in environmental extremes. In *Proc. 8th Int'l Conf. Information Processing Sensor Networks (IPSN '09)*, pages 265–276, San Francisco, CA, USA, April 2009. ACM/IEEE.

[BKM+04] J. Beutel, O. Kasten, F. Mattern, K. Römer, F. Siegemund, and L. Thiele. Prototyping wireless sensor network applications with BTnodes. In *Proc. 1st European Workshop on*

Sensor Networks (EWSN 2004), volume 2920 of *Lecture Notes in Computer Science*, pages 323–338. Springer, Berlin, January 2004.

[BvRW07] Nicolas Burri, Pascal von Rickenbach, and Roger Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *Proc. 6th Int'l Conf. Information Processing Sensor Networks (IPSN '07)*, pages 450–459, New York, NY, USA, April 2007. ACM Press.

[BYAH06] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proc. 4th ACM Conf. Embedded Networked Sensor Systems (SenSys 2006)*, pages 307–320, New York, NY, USA, 2006. ACM Press.

[CABM05] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11(4):419–434, 2005.

[CBR02] I. D. Chakeres and E. M. Belding-Royer. The utility of hello messages for determining link connectivity. In *Proc. 5th Int'l Symp. Personal Wireless Multimedia Communications (WPMC 2002)*, volume 2, pages 504–508, October 2002.

[CvHH04] S. Chatterjea, L. van Hoesel, and P. Havinga. AI-LMAC: An adaptive, information-centric and lightweight MAC protocol for wireless sensor networks. In *Proc. Int'l Conf. on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2004)*, Melbourne, Australia, December 2004.

[CWPE05] Alberto Cerpa, Jennifer L. Wong, Miodrag Potkonjak, and Deborah Estrin. Temporal properties of low power wireless links: modeling and implications on multi-hop routing. In *Proc. 6th ACM Int'l Symp. Mobile Ad Hoc Networking and Computing (MobiHoc 2005)*, pages 414–425, New York, NY, USA, 2005. ACM Press.

[Deb01] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, UK, 2001.

[DFFMM06] Henri Dubois-Ferrière, Laurent Fabre, Roger Meier, and Pierre Metrailler. Tinynode: a comprehensive platform for wireless sensor network applications. In *Proc. 5th Int'l Conf. Information Processing Sensor Networks (IPSN '06)*, pages 358–365, New York, NY, USA, 2006. ACM.

[DGV04] A. Dunkels, B. Grönvall, and T Voigt. Contiki – a lightweight and flexible operating system for tiny networked sensors. In *Proc. 1nd IEEE Workshop on Embedded Networked Sensors (EmNetS-I)*, pages 455 – 462. IEEE, Piscataway, NJ, November 2004.

[DHS+07] Tuan Le Dinh, Wen Hu, Pavan Sikka, Peter Corke, Leslie Overs, and Stephen Brosnan. Design and deployment of a remote robust sensor network: Experiences from an outdoor water quality monitoring network. In *Local Computer Networks, 2007. LCN 2007. 32nd IEEE Conference on*, pages 799–806, October 2007.

[Dun03] Adam Dunkels. Full TCP/IP for 8-bit architectures. In *Proc. 1st ACM/USENIX Conf. Mobile Systems, Applications, and Services (MobiSys 2003)*, pages 85–98, New York, NY, USA, 2003. ACM.

[EEHDP04] C.C. Enz, A. El-Hoiydi, J.D. Decotignie, and V. Peiris. WiseNET: An ultralow-power wireless sensor network solution. *IEEE Computer*, 37(8):62–70, August 2004.

[EH02] A. El-Hoiydi. Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks. In *Proc. IEEE Int'l Conf. on Communications (ICC 2002)*, New York, April 2002.

[EHD04] A. El-Hoiydi and J.D. Decotignie. WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks. In S. Nikoletseas and J.D.P. Rolim, editors, *Proc. 1st Int'l Workshop Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS 2004)*, volume 3121 of *Lecture Notes in Computer Science*, pages 18–31. Springer, Berlin, June 2004.

[ERC08] Relating to the use of short range devices (SRD). ERC recommendation 70-03 (Tromsø1997 and subsequent amendments), February 2008.

[Eur08] Fire detection and fire alarm systems – Part 25: Components using radio links. European Norm (EN) 54-25:2008-06, 2008.

[FGJ+06] Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, Sukun Kim, Philip Levis, and Alec Woo. TinOS 2.x, TEP 123 – The Collection Tree Protocol (CTP). TinyOS Developer List, August 2006.

[FLE06] E. Felemban, C.-G. Lee, and E. Ekici. MMSPEED: Multipath Multi-SPEED protocol for QoS guarantee of reliability and timeliness in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 5(6):738–754, 2006.

[GS06] Lin Gu and John A. Stankovic. t-kernel: providing reliable os support to wireless sensor networks. In *Proc. 4th ACM Conf. Embedded Networked Sensor Systems (SenSys 2006)*, pages 1–14, New York, NY, USA, 2006. ACM.

[HC02] J.L. Hill and D. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, November 2002.

[HC04] J.W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proc. 2nd ACM Conf. Embedded Networked Sensor Systems (SenSys 2004)*, pages 81–94. ACM Press, New York, November 2004.

[HCB02] W. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670, October 2002.

[HDL05] G. Halkes, T. van Dam, and K. Langendoen. Comparing energy-saving MAC protocols for wireless sensor networks. *Mobile Networks and Applications*, 10(5):783–791, October 2005.

[HH04] L. van Hoesel and P. Havinga. A lightweight medium access protocol (LMAC) for wireless sensor networks. In *Proc 1st Int'l Workshop Networked Sensing Systems (INSS04)*, Tokyo, Japan, June 2004.

[HL07] G. Halkes and K. Langendoen. Crankshaft: An energy-efficient MAC-protocol for dense wireless sensor networks. In *Proc. 4th European Workshop on Sensor Networks (EWSN 2007)*, volume 4373 of *Lecture Notes in Computer Science*, pages 276–290, Delft, The Netherlands, January 2007. Springer, Berlin.

[HL09] G. Halkes and K. Langendoen. Experimental evaluation of simulation abstractions for wireless sensor network MAC protocols. In *14th IEEE Int'l Workshop on Computer Aided*

Modeling and Design of Communication Links and Networks (CAMAD '09), Pisa, Italy, June 2009.

[HSW+00] J.L. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proc. 9th Int'l Conf. Architectural Support Programming Languages and Operating Systems (ASPLOS-IX)*, pages 93–104. ACM Press, New York, November 2000.

[IGE00] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. Technical Report 00-732, University of Southern California, March 2000.

[JBT06] K. Jamieson, H. Balakrishnan, and Y.C. Tay. Sift: A MAC protocol for event-driven wireless sensor networks. In *Proc. 3rd European Workshop on Sensor Networks (EWSN 2006)*, volume 3868 of *Lecture Notes in Computer Science*, pages 260–275. Springer, Berlin, February 2006.

[JOW+02] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proc. 10th Int'l Conf. Architectural Support Programming Languages and Operating Systems (ASPLOS-X)*, pages 96–107. ACM Press, New York, October 2002.

[KAH+04] R. Kling, R. Adler, J. Huang, V. Hummel, and L. Nachman. Intel mote: Using Bluetooth in sensor networks. In *Proc. 2nd ACM Conf. Embedded Networked Sensor Systems (SenSys 2004)*, page 318. ACM Press, New York, November 2004.

[KEW02] Bhaskar Krishnamachari, Deborah Estrin, and Stephen B. Wicker. The impact of data aggregation in wireless sensor networks. In *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 575–578, Washington, DC, USA, 2002. IEEE Computer Society.

[KFD+07] Sukun Kim, Rodrigo Fonseca, Prabal Dutta, Arsalan Tavakoli, David Culler, Philip Levis, Scott Shenker, and Ion Stoica. Flush: a reliable bulk transport protocol for multihop wireless networks. In *Proc. 5th ACM Conf. Embedded Networked Sensor Systems (SenSys 2007)*, pages 351–365, New York, NY, USA, 2007. ACM.

[KKP99] J.M. Kahn, R.H. Katz, and K.S.J. Pister. Next century challenges: Mobile networking for smart dust. In *Proc. 5th ACM/IEEE Ann. Int'l Conf. Mobile Computing and Networking (MobiCom '99)*, pages 271–278. ACM Press, New York, August 1999.

[KMW04] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing newly deployed ad hoc and sensor networks. In *Proc. 10th ACM/IEEE Ann. Int'l Conf. Mobile Computing and Networking (MobiCom 2004)*, pages 260–274. ACM Press, New York, 2004.

[KUBHM04] A. Keshavarzian, E. Uysal-Biyikoglu, F. Herrmann, and Arati Manjeshwar. Energy-efficient link assessment in wireless sensor networks. In *Proc. 23rd Ann. Joint IEEE Conf. Computer Communication Soc. (Infocom 2004)*, volume 3, pages 1751–1761, March 2004.

[LBV06] K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture. In *Proc. 20th Int'l Parallel and Distributed Processing Symposium (IPDPS 2006)*, pages 8–15. IEEE, Piscataway, NJ, April 2006. 14th Int'l Workshop Parallel and Distributed Real-Time Systems (WPDRTS 2006).

[LG01] S. J. Lee and M. Gerla. Split multipath routing with maximally disjoint paths in ad hocnetworks. In *Proc. IEEE Int'l Conf. on Communications (ICC 2001)*, volume 10, pages 3201–3205, Helsinki, Finland, 2001.

[LH05] K. Langendoen and G. Halkes. *Embedded Systems Handbook*, chapter Energy-Efficient Medium Access Control. CRC Press, August 2005.

[LKR04] G. Lu, B. Krishnamachari, and C. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering in sensor networks. In *Int'l Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN 2004)*, Santa Fe, NM, April 2004.

[LMH+03] Dhananjay Lal, A. Manjeshwar, F. Herrmann, E. Uysal-Biyikoglu, and A. Keshavarzian. Measurement and characterization of link quality metrics in energy constrained wireless sensor networks. In *Proc. IEEE Global Telecommunications Conf. (GLOBECOM 2003)*, volume 1, pages 446–452, December 2003.

[LMP+05] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. *Ambient Intelligence*, chapter TinyOS: An Operating System for Sensor Networks, pages 115–148. Springer, Berlin, 2005.

[LPCS04] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proc. First Symp. Networked Systems Design and Implementation (NSDI '04)*, pages 15–28. ACM Press, New York, March 2004.

[LWMB09] Roman Lim, Matthias Woehrle, Andreas Meier, and Jan Beutel. Poster abstract: Harvester - low-power environment monitoring out of the box. In *Proc. 6th European Workshop on Sensor Networks (EWSN 2009)*, Cork, Ireland, February 2009. Springer.

[Mar04] M. Maroti. Directed flood-routing framework for wireless sensor networks. In *5th ACM/IFIP/USENIX Int'l Conf. on Middleware*, pages 99–114, 2004.

[MB01] Michael J. McGlynn and Steven A. Borbash. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In *Proc. 2nd ACM Int'l Symp. Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, pages 137–145, New York, NY, USA, 2001. ACM Press.

[MFB99] Muriel Médard, Steven G. Finn, and Richard A. Barry. Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs. *IEEE Transactions on Networking*, 7(5):641–652, 1999.

[MHL07] David Moss, Jonathan Hui, and Philip Levis. TinOS 2.x, TEP 126 – CC2420 Radio Stack. TinyOS Developer List, March 2007.

[MPS+02] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *1st ACM Int'l Workshop on Wireless Sensor Networks and Application (WSNA 2002)*, pages 88–97, Atlanta, GA, September 2002.

[MV04] M. Miller and N. Vaidya. Minimizing energy consumption in sensor networks using a wakeup radio. In *IEEE Wireless Communications and Networking Conference (WCNC04)*, Atlanta, GA, March 2004.

[MvRW06] Thomas Moscibroda, Pascal von Rickenbach, and Roger Wattenhofer. Analyzing the energy-latency trade-off during the deployment of sensor networks. In *Proc. 25th Ann. Joint IEEE Conf. Computer Communication Soc. (Infocom 2006)*, April 2006.

[PG07] Jeongyeup Paek and Ramesh Govindan. RCRT: rate-controlled reliable transport for wireless sensor networks. In *Proc. 5th ACM Conf. Embedded Networked Sensor Systems (SenSys 2007)*, pages 305–319, New York, NY, USA, 2007. ACM.

[PHC04] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proc. 2nd ACM Conf. Embedded Networked Sensor Systems (SenSys 2004)*, pages 95–107. ACM Press, New York, 2004.

[PRW99] Dimitris N. Politis, Joseph P. Romano, and Michael Wolf. *Subsampling*. Springer, 1 edition, August 1999.

[PSC05] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proc. 4th Int'l Conf. Information Processing Sensor Networks (IPSN '05)*, pages 364–369. IEEE, Piscataway, NJ, April 2005.

[RB06] Stanislav Rost and Hari Balakrishnan. Memento: A health monitoring system for wireless sensor networks. In *Proc. 3rd IEEE Communications Society Conf. Sensor, Mesh and Ad Hoc Communications and Networks (IEEE SECON 2006)*. IEEE, Piscataway, NJ, September 2006.

[RBM05] K. Römer, P. Blum, and L. Meier. *Sensor Networks*, chapter Time Synchronization and Calibration in Wireless Sensor Networks. John Wiley & Sons, New York, July 2005.

[RCK+05] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger. In *Proc. 3rd ACM Conf. Embedded Networked Sensor Systems (SenSys 2005)*, pages 255–267. ACM Press, New York, 2005.

[RHL04] N. Reijers, G. Halkes, and K. Langendoen. Link Layer Measurements in Sensor Networks. In *Proc. 1st Int'l Conf. on Mobile Ad-hoc and Sensor Systems (MASS '04)*, October 2004.

[ROGLA03] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for

wireless sensor networks. In *Proc. 1st ACM Conf. Embedded Networked Sensor Systems (SenSys 2003)*, pages 181–192, Los Angeles, CA, November 2003.

[RSF+04] Shad Roundy, Dan Steingart, Luc Frechette, Paul Wright, and Jan Rabaey. Power sources for wireless sensor networks. In *Proc. 1st European Workshop on Sensor Networks (EWSN 2004)*, volume 2920 of *Lecture Notes in Computer Science*, pages 1–17. Springer, Berlin, January 2004.

[RWAM05] I. Rhee, A. Warrier, M. Aia, and J. Min. Z-MAC: a hybrid MAC for wireless sensor networks. In *Proc. 3rd ACM Conf. Embedded Networked Sensor Systems (SenSys 2005)*, pages 90–101, 2005.

[RWR03] S. Roundy, P.K. Wright, and J.M. Rabaey. *Energy Scavenging for Wireless Sensor Networks with Special Focus on Vibrations*. Springer, Berlin, 2003.

[SAA03] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz. ESRT: Event-to-sink reliable transport in wireless sensor networks. In *Proc. 4th ACM Int'l Symp. Mobile Ad Hoc Networking and Computing (MobiHoc 2003)*, pages 177–188, June 2003.

[SDTL06] Kannan Srinivasan, Prabal Dutta, Arsalan Tavakoli, and Philip Levis. Understanding the causes of packet delivery success and failure in dense wireless sensor networks. In *Proc. 4th ACM Conf. Embedded Networked Sensor Systems (SenSys 2006)*, pages 419–420, New York, NY, USA, 2006. ACM Press.

[SGJ08] Y. Sun, O. Gurewitz, and D. Johnson. RI-MAC: A receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In *Proc. 1st ACM Conf. Embedded Networked Sensor Systems (SenSys 2003)*, pages 1–14, Raleigh, NC, November 2008.

[SH03] F. Stann and J. Heidemann. RMST: Reliable data transport in sensor networks. In *First IEEE Int'l Workshop on Sensor Net Protocols and Applications*, pages 102–112, Anchorage, AK, April 2003.

[SKH06] D. Son, B. Krishnamachari, and J. Heidemann. Experimental study of concurrent transmission in wireless sensor networks. In *Proc. 4th ACM Conf. Embedded Networked Sen-*

sor Systems (SenSys 2006), pages 237–250. ACM Press, New York, 2006.

[SL06] Kannan Srinivasan and Phil Levis. RSSI is under appreciated. In *Proc. 3rd IEEE Workshop on Embedded Networked Sensors (EmNetS-III)*, May 2006.

[STGS02] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Topology management for sensor networks: Exploiting latency and density. In *Proc. 3rd ACM Int'l Symp. Mobile Ad Hoc Networking and Computing (MobiHoc 2002)*, pages 135–145. ACM Press, New York, June 2002.

[Sun] Projet Sun SPOT. http://www.sunspotworld.com.

[TC05] G. Tolle and D. Culler. Design of an application-cooperative management system for wireless sensor networks. In *Proc. 2nd European Workshop on Sensor Networks (EWSN 2005)*, pages 121–132, February 2005.

[TG01] K. Tang and M. Gerla. MAC reliable broadcast in ad hoc networks. In *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, volume 2, pages 1008–1013, 2001.

[TPS+05] Gilman Tolle, Joseph Polastre, Robert Szewczyk, David Culler, Neil Turner, Kevin Tu, Stephen Burgess, Todd Dawson, Phil Buonadonna, David Gay, and Wei Hong. A macroscope in the redwoods. In *Proc. 3rd ACM Conf. Embedded Networked Sensor Systems (SenSys 2005)*, pages 51–63, New York, NY, USA, 2005. ACM Press.

[vDL03] T. van Dam and K. Langendoen. An adaptive energy efficient MAC protocol for wireless sensor networks. In *Proc. 1st ACM Conf. Embedded Networked Sensor Systems (SenSys 2003)*, pages 171–180. ACM Press, New York, November 2003.

[WA06] Kai-Juan Wong and D. K. Arvind. SpeckMAC: low-power decentralised MAC protocols for low data rate transmissions in specknets. In *Proc. 2nd international workshop on Multi-hop ad hoc networks: from theory to reality (REALMAN '06)*, pages 71–78, New York, NY, USA, 2006. ACM.

[WCK02] C.-Y. Wan, A. Campbell, and L. Krishnamurthy. PSFQ: A reliable transport protocol for wireless sensor networks. In *1st ACM Int'l Workshop on Wireless Sensor Networks and Application (WSNA 2002)*, pages 1–11, Atlanta, GA, September 2002.

[WK03] Szu-Chi Wang and Sy-Yen Kuo. Communication strategies for heartbeat-style failure detectors in wireless ad hoc networks. In *Proc. Int'l Conf. DependableSystems and Networks (DSN '03)*, pages 361–376, June 2003.

[WM06] Andreas Willig and Robert Mitschke. Results of bit error measurements with sensor nodes and casuistic consequences for design of energy-efficient error control schemes. In *Proc. 3rd European Workshop on Sensor Networks (EWSN 2006)*, volume 3868 of *Lecture Notes in Computer Science*, pages 310–325. Springer, Berlin, February 2006.

[WTC03] Alec Woo, Terence Tong, and David Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proc. 1st ACM Conf. Embedded Networked Sensor Systems (SenSys 2003)*, pages 14–27, New York, NY, USA, 2003. ACM Press.

[YHE02] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proc. 21st Ann. Joint IEEE Conf. Computer Communication Soc. (Infocom 2002)*, volume 3, pages 1567–1576, June 2002.

[YSH06] W. Ye, F. Silva, and J. Heidemann. Ultra-low duty cycle MAC with scheduled channel polling. In *Proc. 4th ACM Conf. Embedded Networked Sensor Systems (SenSys 2006)*, pages 321–334, New York, NY, USA, 2006. ACM Press.

[YZLZ05] F. Ye, G. Zhong, S. Lu, and L. Zhang. GRAdient Broadcast: A robust data delivery protocol for large scale sensor networks. *Wireless Networks*, 11(3):285–298, May 2005.

[ZBG98] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. GloMoSim: a library for parallel simulation of large-scale wireless networks. *SIGSIM Simul. Dig.*, 28(1):154–161, 1998.

[ZG03] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proc. 1st ACM Conf. Embedded Networked Sensor Systems (SenSys 2003)*, pages 1–13, 2003.

[ZK04] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *Proc. 1st IEEE Communications Society Conf. Sensor, Mesh and Ad Hoc Communications and Networks (IEEE SECON 2004)*, pages 517–526, October 2004.

[ZRS05] T. Zheng, S. Radhakrishnan, and V. Sarangan. PMAC: an adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proc. 19th Int'l Parallel and Distributed Processing Symposium (IPDPS 2005)*, pages 65–72, Denver, CO, April 2005.

[ZRW04] L. C. Zhong, J. M. Rabaey, and A. Wolisz. An integrated data-link energy model for wireless sensor networks. In *Proc. IEEE Int'l Conf. on Communications (ICC 2004)*, volume 7, pages 3777–3783, June 2004.

Curriculum Vitae

Name	Andreas Meier
Date of Birth	10 November 1978
Citizen of	Luzern, Oberkirch (LU)
Nationality	Swiss

Education:

11/2005–06/2009	ETH Zurich, Computer Engineering and Networks Laboratory. PhD Thesis under the Supervision of Prof. Dr. Lothar Thiele
10/2002–07/2005	ETH Zurich, MSc ETH in Electrical Engineering and Information Technologies
09/2003–03/2004	University of Edinburgh, Academic Exchange
10/2000–09/2002	ETH Zurich, Basic Studies in Electrical Engineering and Information Technologies
10/1999–09/2000	ETH Zurich, Basic Studies in Mechanical Engineering
08/1994–02/1999	MNG Zurich, Matura, Type C (Scientific type)

Professional Experience:

11/2005–06/2009	ETH Zurich, Research Assistant at the Institute for Computer Engineering and Networks Laboratory
11/2007–04/2008	University of Singapore, Visiting Researcher
02/2005–08/2005	DaimlerChrysler Research and Technology North America, Palo Alto (USA), Internship and Master Thesis
08/1997–09/2003	Inexsys AG, Zurich, Software Engineer