RELIABILITY IN ENERGY EFFICIENT WIRELESS SENSOR NETWORKS

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Markus Anwander

von Untereggen SG

Leiter der Arbeit:
Professor Dr. Torsten Braun
Institut für Informatik und angewandte Mathematik

RELIABILITY IN ENERGY EFFICIENT WIRELESS SENSOR NETWORKS

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Markus Anwander

von Untereggen SG

Leiter der Arbeit:
Professor Dr. Torsten Braun
Institut für Informatik und angewandte Mathematik

Von der Philosophisch-naturwissenschaftlichen Fakultät angenommen.

Bern, 18.12.2012

Der Dekan:
Prof. Dr. Silvio Decurtins

Abstract

Wireless Sensor Networks (WSNs) are characterized by easy installation and adaptive self-organizing with no need for maintenance. The individual sensor nodes of WSNs are usually battery powered, equipped with sensing devices and use a low-power radio module for communication. The limited lifetime of batteries makes energy preserving mechanisms an important topic of current WSN research. Moreover, a WSN network stack has to offer reliable communication services to guarantee the functionality of WSN applications. Unfortunately, reliability techniques require additional energy to recover packet loss. High traffic load involving intra/inter-flow interferences and congestion are especially challenging for energy aware reliability mechanisms. Additionally, WSN network stacks have to support standardized protocol headers to enable communication in heterogeneous WSNs. To solve these problems, this thesis contributes a link layer with two novel mechanisms to support energy efficiency and reliability. Both mechanisms are able to work together with energy efficient and robust packet-oriented radio modules. The first mechanism is a novel traffic load measuring mechanism using traffic prediction. It predicts the expected incoming traffic load during periods of congestion. This enables selecting long sleeping periods to preserve energy while it simultaneously provides sufficient bandwidth during high traffic load or congestion. The second mechanism is a congestion detection mechanism that is able to identify and correctly handle intra-flow and inter-flow interferences as well as congestion. Our contributed WSN stack supports IEEE 802.15.4, IP, UDP and TCP to enhance the network connectivity and offers an energy efficient UDP based end-to-end reliability protocol. Moreover, we analyzed the impact of Forward Error Correction (FEC) codes on energy efficiency and reliability performance. The analyzed FEC codes are able to reduce the required local retransmission attempts but they are neither able to reduce the energy usage nor to enhance the reliability of our real world WSN network stack using a packet-oriented radio module. To evaluate our contributed WSN network stack we compared it to existing real world WSN network stacks. It uses less energy and shows a lower packet loss than the other stacks. Conducted experiments show that our WSN network stack is the most energy efficient reliable among the evaluated stacks. At the same time, our contributed link layer protocols show significantly higher throughput than the other evaluated protocols, i.e. up to 4 times higher throughput. This is because our contributed link layer protocols are able to handle intra-flow and inter-flow interferences as well as congestion.

Contents

List of Figures

List of Tables

Preface

The following PhD thesis is based on work performed during my employment as a research assistant at the Institute of Computer Science and Applied Mathematics (IAM) of the University of Bern, Switzerland. The research conducted has been partially supported by the Hasler Foundation under grant number ManCom 2060 and the Swiss National Science Foundation under grant number (200020-113677/1).

I would like to thank everybody who provided me with support, ideas, and encouragement during my employment at the Communication and Distributed Systems group (CDS). First, I want to express my gratitude to Prof. Dr. Torsten Braun, head of the Communication and Distributed Systems group (CDS), who supervised and encouraged my work. He offered me an interesting and challenging work environment and the opportunity to participate in national and European research and technology transfer projects. I would also like to thank Prof. Dr. Mesut Güneş for reading this work and Prof. Dr. Gerhard Jäger, who was willing to be co-examiner.

Many thanks go to my colleagues at the institute and in our research group for being part of the CDS team. In particular, I want to thank Islam Alyafawi, Carlos Anastasiades, Thomas Bernoulli, Peppo Brambilla, Marc Brogle, Desislava Dimitrova, Kirsten Dolfus, Philipp Hurni, Almerima Jamakovic-Kapic, Zan Li, Dragan Milic, Benjamin Nyffenegger, Thomas Staub, Nikolaos Thomos, Gerald Wagenknecht, Markus Wälchli, Markus Wulff and Zhongliang Zhao.

Special thanks to the secretaries Ruth Bestgen and Daniela Schroth of the CDS research group for their support in the administrative tasks during all these years. I am very grateful to my girlfriend Nadine Nigg for supporting me in many ways and being always patient during the years of my PhD thesis.

Last but not least, I would like to express my thanks to Thomas Staub for proofreading this thesis, correcting language and stylistic errors and pointing out inconsistencies in the original manuscript.

Chapter 1

Introduction

A Wireless Sensor Network (WSN) consists of a number of autonomous sensor nodes. Each sensor node is equipped with different sensing devices, a microcontroller and a radio module for wireless data transmissions. The individual sensor nodes are usually battery driven, which makes energy efficiency a very important field of current WSN research. The main objective of this thesis is to enhance energy efficiency and reliable data transmission in WSNs. Energy efficient support of reliable data transmission in WSNs requires a combination of efficient hardware, appropriate energy preserving algorithms and robust packet loss recovery mechanisms. This chapter briefly introduces different characteristics of WSNs, then describes the related challenges and outlines the contributions of this thesis. Finally, it summarizes the following chapters of the thesis.

1.1 Overview

WSNs have experienced an increasing degree of research interests and a growing number of industrial applications in the last decade. They are characterized by easy installation and adaptive self-configuration with no need for maintenance. Figure 1.1 depicts different common WSN applications:

- **Environmental monitoring:** WSNs are able to monitor environmental conditions such as temperature, humidity, illumination, water flows, air pollution, chemical emissions or radiation. They enable a continuous and unattended remote surveillance of large, inaccessible or protected areas. Applications for environmental monitoring have become an area of growing interest. For example, water flow studies in the Alps help to enhance the water supply during dry summer periods [1].
- **Detection and tracking of objects:** Object detection and tracking enabled WSNs are applied within different fields of applications. Detecting and tracking people is used for security surveillance, crowd detection, people counting or for timekeeping systems to manage employees' working time. Detecting and tracking of objects is used to localize freight containers, cars

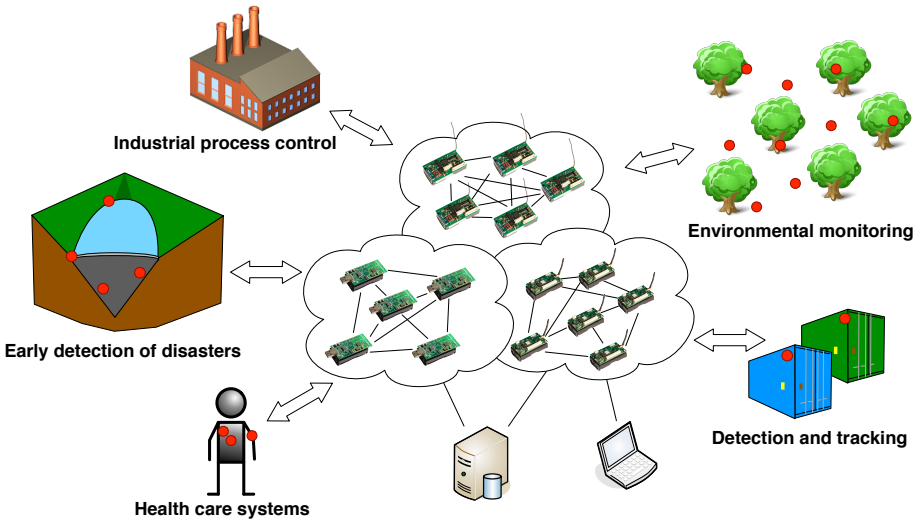


Figure 1.1: WSN Applications.

or equipment in hospitals. For example, car tracking applications are used in urban environments to handle traffic flows or to manage parking lots including automated parking fee accounting.

- **Health care systems:** WSNs are able to continuously monitor vital values of the human body such as its temperature, blood pressure, glucose level or heart and brain activity. This enables real time patient monitoring to rapidly detect clinical deterioration or to improve the life quality of elderly people by prolonging their time living at their own home. Additionally, WSNs are used in medical and psychological large field studies to study chronic diseases or the human behavior.

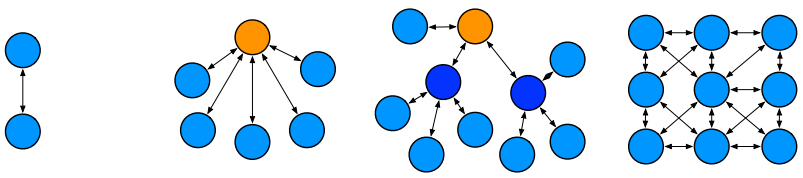
- **Early detection of disasters:** WSNs are able to detect geophysical hazards such as landslides, tsunamis, earthquakes or volcanic eruptions as well as meteorological disasters caused by extreme weather or wildfires. Early warnings for natural or man made disasters are used to reduce the impact of these events on lives and property. For example, wildfire detection is used to plan the evacuation of residential areas or to support firefighters. Additionally, fire detection WSNs are used inside buildings to accelerate evacuation time or active sprinkler systems.

- **Industrial Process Control:** WSNs are used in industrial processes for machine condition monitoring, building automation, predictive maintenance, energy management or vendor managed inventory. The ultimate goals of these WSNs are to decrease the manufacturing costs, to improve the operational reliability and to increase the health and safety of the employees.

The various real world WSNs are individually customized to fulfill their intended purposes. The design of the individual WSNs has to be cost-efficient and to ensure functional reliability during operation. The various application purposes result in various WSNs, which differ in the following characteristics:

- **Hardware components:** Sensor nodes used in WSNs differ in size, attached sensing devices, calculation performance, available memory, battery size and attached radio module. Different WSN applications require different kinds of sensing devices to measure the required environmental characteristics. This results in heterogeneous WSNs, consisting of different types of sensor node platforms equipped with the corresponding sensing devices. Different radio modules attached to the individual sensor node platforms in heterogeneous WSNs represent a challenge to network connectivity. Usually, these different radio modules make use of an individual wireless channel, which prevents a direct communication between the individual sensor nodes.

- **Network topology:** Common WSNs network topologies are using point-to-point, star, tree or mesh topologies. Figure 1.2a depicts a point-to-point topology, where two or more nodes are directly interconnected with each other. A star topology uses a centralized node to coordinate the traffic (see Figure 1.2b). Tree topologies can be considered as a combination of several star topologies (see Figure 1.2c). In mesh topologies the individual nodes are connected to every other node in the neighborhood (see Figure 1.2d).



(a) Point-to-point. **(b)** Star. **(c)** Tree. **(d)** Mesh.

Figure 1.2: Small-scale network topologies for basic evaluations.

- **Deployment:** The positions of the individual sensor nodes inside the WSN can be planned systematically or uniformly random. Sensor nodes placed in a square grid represent a classical systematic deployment of WSNs. Individual sensor nodes inside buildings are sometimes systematically placed to enhance connectivity.

  In a uniform random deployment, individual nodes have an equal probability of being placed at any point in a given deployment area. For example, such a random deployment can result from scattering sensor nodes by an airplane. Uniform random deployment is cost-efficient and easy to apply for large WSNs.

- **Mobility:** Individual sensor nodes in WSNs are generally static. Nevertheless, some WSN applications make use of mobile nodes. The movement of the individual mobile nodes can be constant or sporadic with slow or at high speed. Advantages of mobile WSNs over static WSNs are for instance improved coverage and an enhanced target tracking potential. Mobile WSNs require dynamic routing mechanisms to ensure connectivity between the individual nodes.

- **Connectivity:** Network connectivity between the individual sensor nodes can be permanent, periodic or only sporadic. Network connectivity may include dynamic routing mechanisms to repair broken network links. For example, when a sensor node runs out of power, then a dynamic routing mechanism has to reconfigure routing inside the WSN. Additionally, some WSNs use monitoring applications to observe current connectivity inside the WSN to ensure quality of service.

- **Coverage and network size:** WSNs show different network sizes and coverage strategies. WSN based health care systems attached to the human body usually consist of few individual sensor nodes. Environmental monitoring applications on the other hand may consist of thousands of individual sensor nodes. The coverage of a local area can be sparse, dense or redundant. The coverage of a WSN can refer to the coverage of the sensing devices as well as to the coverage of wireless radio devices.

- **Lifetime:** One common characteristic for most sensor node platforms is, that they are battery powered to feature an autonomous power supply. Unfortunately, the lifetime of battery powered sensor nodes is limited. Therefore, energy preserving mechanisms are an important field of current WSN research. The required lifetime of a WSN depends on its intended purpose. For example, medical applications for patient monitoring have to be operational for several hours or days only, while a volcano monitoring application has to collect data during years.

The multiple characteristics and requirements for real world WSNs result in various research interests in the field of WSNs. Although all WSNs have very different characteristics, some requirements are common. We have identified energy preservation as the major common requirement for real world applications. Therefore, energy preserving mechanisms to increase the lifetime of battery powered sensor nodes are an important field of current WSN research. Two other common requirements are reliable data transmissions and interconnection of heterogeneous nodes equipped with sensing devices. Reliable data transmissions are required by WSN applications for different purposes, e.g. to reliable report critical events or to send configuration messages or code updates. The interconnection of heterogeneous nodes is required to enable communication among different sensor node platforms and to directly connect them to the Internet.

1.2 Problem Statement

In this thesis, we investigate three major problems of current real world WSN applications, namely energy efficiency, reliable data transmissions and network connectivity. Energy efficiency and data reliability are basic requirements for most real world WSNs applications. As sensor nodes are powered by batteries, their lifetime is limited. Data reliability mechanisms are required to ensure successful data forwarding. Therefore, data reliability mechanisms have to be able to detect and recover packet loss. Additionally, a real world WSN stack has to enable communication between individual heterogeneous sensor node platforms and the Internet.

- **Energy efficiency:** A key challenge in the design as well as during the operation of WSNs is the extension of the system lifetime. Battery powered sensor nodes feature only a limited amount of energy, which determines the node and network lifetime. To increase the lifetime of a WSN the energy consumption of all sensor nodes has to be reduced as replacing the batteries is often not feasible. Therefore, energy preserving mechanisms present a major research challenge for WSNs. The most energy consuming parts of a sensor node are the attached sensors, the microcontroller and the radio module. Energy preserving mechanisms use sleep and wake-up scheduling techniques to put the individual components as long as possible into an energy preserving sleep mode. The longer the sleeping period in relation to the wake-up period is, the lower is the resulting duty cycle of individual components. A duty cycle is defined by the ratio of the wake-up period and the duty cycle period. Therefore, reducing the duty cycle of the individual components on a sensor node is, next to using energy efficient hardware, the most effective way to reduce energy consumption. The most effective, but also most challenging way to preserve the energy is to reduce the duty cycle of the radio module, as the radio module is usually the component with the highest energy usage during the operation. Furthermore, a radio module requires 10^3 - 10^6 times less energy in sleeping mode than in listening mode. Therefore, the sleeping period of the radio module should be adapted to the currently forwarded traffic load. Long sleeping periods ensure a low energy usage of the radio module at low traffic load. Short sleeping periods provide short data forwarding delays and offer sufficient bandwidth to forward incoming traffic at high traffic load. The link layer protocol must to be able to determine the expected incoming traffic load to apply an appropriate sleeping period. Current traffic load detection mechanisms are not able to determine if a low traffic load is caused by a low amount of generated packets or by congestion. This represents a major problem, as a low amount of generated packets enables long sleeping periods, whereas congestion requires exactly the opposite, short sleeping periods.

- **Reliable data transmission:** Current WSN protocols only offer weak data reliability support. Usually, they struggle when handling inter-flow and intra-flow interferences caused by concurrently forwarded packets. Especially, high traffic load depicts a challenge for existing link layer protocols. During periods with high traffic load, transmission attempts have to be scheduled carefully in order to not interfere with transmissions of, maybe hidden, other nodes. Otherwise, higher traffic load results in congestion and significant packet loss.

  End-to-end reliability protocols are able to detect packet loss and to trigger a new end-to-end retransmission of corresponding packet. This requires a considerable amount of additional energy to retransmit the lost packet. Additionally, end-to-end reliability protocols have a difficult time when handling packet loss at high traffic load. The extra traffic caused by end-to-end acknowledgments generates additional interferences and packet loss. Therefore, end-to-end reliability mechanisms, such as TCP, even increase the packet loss at high traffic load instead of reducing them. Local hop-to-hop retransmission mechanisms require less energy than end-to-end retransmission mechanisms to retransmit an erroneous packet. Unfortunately, hop-to-hop reliability mechanisms cannot trigger an end-to-end retransmission of a dropped packet.

- **Network connectivity:** The individual sensor node platforms deployed in a real world WSN should to be interconnected among themselves as well as connected to the Internet. An interconnection of different nodes requires a WSN protocol stack using well-established standards released by standardization bodies such as Institute of Electrical and Electronics Engineers (IEEE) and Internet Engineering Task Force (IETF) or at least quasi-standards described by Request for Comments (RFCs). Standardized physical and data link layer protocol (e.g., IEEE 802.15.4) enable direct communication among neighbor nodes. A standardized network layer protocol (e.g., IP) enables addressing and data routing between individual nodes within a network. Standardized transport layer protocols such as Transport Control Protocol (TCP) and User Datagram Protocol (UDP) enable establishing and handling an end-to-end connection or a data flow between different nodes.

Our research into these three major problems showed that the radio module has a significant impact to all three research topics. Additionally, the radio module influences the design of the energy preserving link layer protocol. Therefore, the radio module has to be selected carefully to achieve an energy efficient and reliable WSN stack. Next to that, the limited resources of the microcontroller require to design not only energy preserving but also efficient mechanisms in terms of processing power and available memory.

1.3 Contributions

The main contribution of this thesis is an energy efficient and reliable data transmission in a real world WSN stack, which is compliant to IEEE standards and commonly used RFCs. On the link layer, we contribute with two novel mechanisms to support energy efficiency and data reliability. The first contribution is a novel traffic load measuring mechanism called **traffic prediction mechanism**. The mechanism predicts the expected incoming traffic load, also during periods of congestion. This enables selecting long sleeping periods to preserve energy, while it simultaneously provides sufficient bandwidth during high traffic load or congestion.

The second contribution is a **congestion detection and control mechanism**, which is able to identify and correctly handle intra-flow and inter-flow interferences as well as congestion. Our contributed WSN stack supports IEEE 802.15.4, IP, UDP and TCP to enhance network connectivity. The link layer protocol works with energy efficient and robust packet oriented radio modules. To support energy efficient end-to-end reliability we contribute an application layer overlay protocol providing reliable end-to-end flows over UDP. Additionally, we evaluate the impact of different radio modules as well as the impact of Forward Error Correction (FEC) codes on energy usage and data reliability in WSNs. Our network stack is evaluated by simulation experiments and in real world testbeds.

The individual contributions of this thesis are summarized as follows:

- **Energy and data reliability evaluation of different radio modules.** We evaluated five common sensor node platforms equipped with different radio modules. We analyzed their energy usage and data reliability under different network conditions, e.g., with low SNR or with high interferences. We show that packet oriented radio modules perform significantly better than bit and byte oriented radio modules. Packet oriented radio modules offer a higher robustness against interferences and require significantly less energy for transmissions and channel checks. Additionally, they support IEEE 802.15.4 compliant physical and data link layers to enhance the connectivity in heterogeneous WSNs.

- **Energy preserving, adaptive radio duty cycle protocol supporting traffic load prediction mechanism.** We contribute an energy efficient, adaptive link layer protocol designed for energy efficient and robust packet oriented radio modules. The **traffic prediction mechanism** that we developed is able to predict the incoming traffic even during congestion or periods with high interferences. This enables the protocol to adapt the radio duty cycle according to the incoming traffic load. The developed protocol header of link layer protocol is compliant to IEEE 802.15.4.

- **Hop-to-hop reliability protocol with congestion detection and control mechanism.** We developed a hop-top-hop reliability protocol with a **con-**

gestion detection and control mechanism. This congestion detection and control mechanism is able to detect and properly react on intra-flow and inter-flow interferences by adapting the frequency of the individual transmission attempts. This results in a higher throughput and reduces the number of energy consuming retransmission attempts caused by packet collisions.

- **End-to-end reliability support for UDP.** We developed an application layer overlay protocol supporting end-to-end reliability based on UDP. Using this application layer overlay protocol, a reliable and energy efficient end-to-end transport service between sensor nodes and machines in the Internet may be offered to any application. The application layer overlay protocol employs standard network sockets to support a regular application programming interface.

- **Modular network stack for heterogeneous WSNs.** Our contributed network stack extends the existing modular Contiki network stack [23]. We only use standardized protocol headers for the physical, network and transport layer to enable communication in heterogeneous WSNs.

- **Evaluation of FEC concerning energy consumption and reliability.** FEC codes showed promising results on bit/byte oriented radio modules due to the low reliability performance and low energy efficiency of these modules. Nevertheless, the FEC codes that have been evaluated in this thesis are neither able to reduce the energy usage nor to enhance the reliability of our contributed real world WSN network stack using a packet oriented radio module.

As a next step, we shortly discuss our contributions using an example of a sensor node communicating to the external Internet. Figure 1.3 shows a unicast connection between a sensor node using our WSN stack and a server located in the Internet. The transport layer provides the end-to-end communication services for the applications. Our network stack offers the transport protocols TCP and UDP. TCP already provides services such as end-to-end reliability, flow control and congestion control, while UDP is delegating these functions to the application. UDP requires less energy to forward application payload and offers a higher throughput than TCP. Therefore, we developed a UDP based application layer overlay protocol offering end-to-end reliability to the application. Our network stack is using IP on the network layer. Our link layer is divided into two sublayers. The upper sublayer provides hop-to-hop reliability as well as a congestion control and a back pressure mechanism. The lower sublayer implements the adaptive radio duty cycle mechanism including a traffic prediction mechanism to estimate expected incoming traffic. On the physical layer, we apply the Direct Sequence Spread Spectrum (DSSS) technique for data transmission. DSSS significantly improves robustness against interferences caused by other transmissions or wave propagation effects such as reflections, multi-path fading or diffraction.

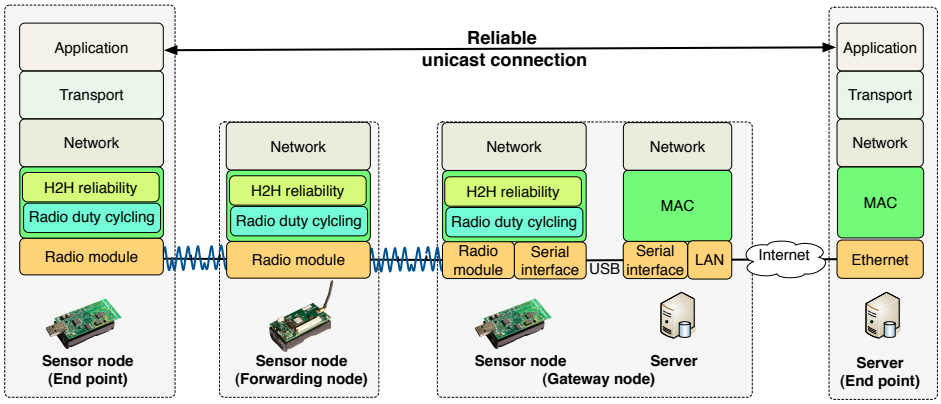


Figure 1.3: Energy efficient and reliable network stack for heterogeneous WSNs.

For our selected scenario, at least one sensor node in the WSN has to be attached (e.g., over USB) to a machine connected to the Internet to work as a gateway node. This gateway node handles all the routing for packets between the WSN and the Internet.

The implementation of the contributed network stack is based on the modular Contiki network stack. The network stack can be compiled for different sensor node platforms. This enables the usage of the same protocol and application implementations on different sensor node platforms. Only the radio driver and the radio duty cycle protocol at the link layer have to be selected according to the used radio module. Our link layer protocols are optimized for the packet oriented radio module CC2420. Therefore, our network stack can be used for every sensor node using such a radio module (e.g, telosB and MicaZ).

Besides the already presented contributions, we evaluated the impact of FEC codes on energy efficiency and data reliability. FEC codes have shown promising results in reducing packet loss on bit/byte-oriented radio modules using a simple On Off Keying (OOK) modulation technique. FEC codes are able to reduce the required local retransmission attempts and, therefore, they possibly are able to reduce the energy consumption in real world WSNs. Bit/byte-oriented radio modules require several times more energy and time to forward data than than packet oriented radio modules. Additionally, packet oriented radio modules use significantly more advanced spectrum spreading and modulation techniques than bit/byte-oriented radio modules. Bit/byte-oriented radio modules show a higher potential for FEC codes to save energy due to the higher probability of packet loss and the higher energy requirements for a retransmission attempt. We evaluated the impact of FEC codes on the energy efficiency and data reliability of our contributed real world WSN stacks using packet oriented radio modules. Figure 1.4 shows a schematic representation of the FEC code evaluations including the impact of spectrum spreading and modulation techniques.

- **FEC mechanism:** The FEC codes are handled by the microcontroller. FEC codes add redundant parity bytes to the outgoing data stream. We evaluated the Hamming(12,8) FEC code and the Reed-Solomon(255,225) FEC code. The Hamming(12,8) algorithms require less calculation time for encoding and decoding the parity information than more complex algorithms of Reed-Solomon(255,225). Therefore, Hamming(12,8) requires less energy than Reed-Solomon(255,225), while Reed-Solomon(255,225) is able to recover more bit errors than Hamming(12,8).

- **Spectrum spreading techniques:** Spectrum spreading techniques are used to enhance the robustness against interferences. Manchester encoding is used to ensure sufficient transitions of the data stream to keep the radio receiver module using an OOK modulation in synchronization. The DSSS technique spreads the incoming data stream by a factor of eight, which enables the receiver to recover bit errors. On the receiver side, frame start detection is performed after spectrum spreading has restored the data stream. Therefore, a spectrum spreading mechanism is able to increase the frame detection rate.

- **Modulation techniques:** The spread data streams have to be modulated on a carrier wave to be transmitted over the air to the receiver. The evaluated bit/byte-oriented radio modules use a simple OOK modulation technique, while the packet oriented radio module is using Offset Quadrature Phase Shift Keying (OQPSK), where two digital data signals are modulated to one carrier wave at the same time.

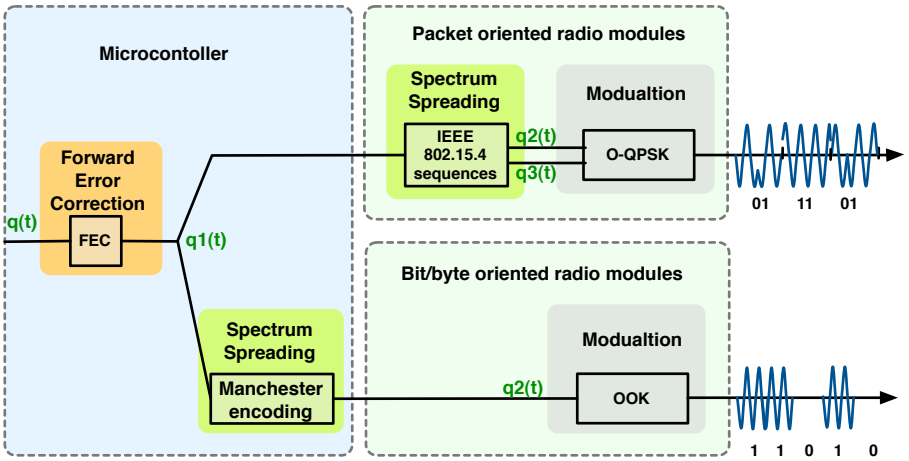


Figure 1.4: FEC mechanisms versus spread spectrum techniques.

Our evaluations show that performance of FEC codes heavily depends on the used radio module. All evaluated FEC codes were able to reduce the required retransmission attempts for both types of radio modules. But when using packet oriented

radio modules, the energy preserved by less transmission attempts is insufficient to compensate the additional energy costs caused by the FEC calculations on the microcontroller and the additional energy required by the radio module to transmit parity data. Therefore, we do not use FEC support in our energy efficient and reliable WSN stack.

1.4 Thesis Outline

The thesis is structured as follows. Chapter 2 provides a discussion of the related work in the area of energy efficient and reliable WSNs. Chapter 3 describes the evaluation of different radio modules. The evaluation results enable us to select the most appropriate radio modules to achieve an energy efficiency and reliable link layer protocol. Chapter 4 introduces the two contributed radio duty cycled link layer protocols and the application layer overlay protocol to support an energy efficient and reliable WSN stack. The design of a radio duty cycled link layer protocol depends on the selected radio module. In Chapter 5, our developed real world WSN stack is evaluated using the WISEBED real world testbed. Additionally, the impact of FEC is evaluated. Finally, Chapter 6 concludes the thesis and discusses potential future work.

Chapter 2

Related Work

This chapter introduces and discusses the related work concerning the main topics of this thesis, which are energy efficiency, reliable data transmission and network connectivity. Section 2.1 describes the characteristics of low-power radio modules for WSNs. Then, Section 2.2 presents the sensor node platforms equipped with the individual radio modules introduced in the Section 2.1. Section 2.3 discusses the operating systems used on sensor node platforms. Section 2.4 describes the used evaluation tools, such as WSN simulators, real world WSN testbeds and the hardware setup for energy measurements based on a digital multimeter. Section 2.5 discusses different reliability mechanisms used by WSNs. Finally, Section 2.6 introduces the most important related work concerning energy efficient link layer protocols for WSNs.

2.1 Radio Modules and Physical Layer Techniques

This section introduces and categorizes different low-power radio modules designed for communication in WSNs. Additionally, this Section describes the physical and link layers used by theses radio modules.

The remainder of this section is structured as follows. Subsection 2.1.1 gives an overview about the different radio modules for WSNs. Subsection 2.1.2 introduces different modulation techniques used by low-power radio modules for WSNs. Subsection 2.1.3 describes frequency speeding techniques used by the radio modules. The following for Subsections 2.1.4, 2.1.5, 2.1.6 and 2.1.7 introduce the four used radio modules.

2.1.1 Survey of Characteristics of Radio Modules for WSNs

Currently available radio modules for WSNs can be roughly divided into two classes, namely bit/byte oriented radio modules and packet oriented radio modules. Bit oriented radios radio modules only provide a digital transmission interface to the microcontroller. This enables a connected microcontroller to transmit individual bits over the radio module to other sensor node platforms. The microcontroller

of the receiving node has to continuously read the signal strength indicated by the connected radio module to estimate the individual received bits.

Byte oriented radio modules enable a connected microcontroller to send and receive individual bytes, instead of individual bits. To transmit a byte stream, the individual bytes have to be written to the radio module. The exact time scheduling of the individual byte transmissions has to be coordinated by the microcontroller. On the receiver side, a byte radio module is able to announce the reception of a single byte by triggering an interrupt of the microcontroller.

The flexibility in the design process of a byte stream, which is provided by bit/byte oriented radio modules, enables a microcontroller to implement almost any protocol design. Physical layer tasks, such as frame start detection and bit rate control, as well as all link layer tasks, such as checksum calculations or acknowledgment handling are controlled and handled directly by the microcontroller. Bit rate control on the physical layer includes the need for an exact scheduling of the individual bit/bytes write operations according to the selected physical bit rate. The accuracy of the used microcontrollers enables a physical transmission bit rate up to 38.4 kbps. Most real world implementations even apply a lower bit rate of 9.6 kbps and Manchester coding (see Section 2.1.3) to achieve a lower bit error rate.

In contrast to bit/byte oriented radio modules, the highly integrated packet oriented radio modules allow the connected microcontroller to only send and receive entire packets. It is not possible for the microcontroller to send and receive individual bytes using these packet oriented radio modules. All tasks of the physical layer are autonomously handled by the radio modules itself. The packet oriented radio modules offer an effective bit rate of 250 kbps on the link layer. Additionally, packet oriented radio modules offer several additional functions on the link layer, such as handling acknowledgments or calculating checksums. Link layer functions executed by the radio module require considerably less time and energy than a corresponding implementation on the microcontroller. Furthermore, packet oriented radio modules require significantly less time and energy for sending a packet du to the higher bit rate. The main drawback of packet oriented radio modules is their limitation in the design process of the physical and link layer protocol header. This decreases the flexibility in designing a suitable WSN communication protocol.

Table 2.1 gives a brief overview about the characteristics of the different radio modules used in this thesis. All operate in the narrowband, which is considered to cover frequencies 300 - 3400 MHz. The next subsection introduces the different modulation techniques used by radio modules presented in Table 2.1.

2.1.2 Modulation Techniques

This subsection briefly introduces different modulation techniques used by low-power radio modules for WSNs. Modulation techniques are used to encode data symbols to an electromagnetic wave, called carrier wave. To encode a data symbol to a carrier wave, either the amplitude, frequency or phase of the carrier wave is modulated according to the data symbol. The frequency of the carrier wave is

Radio	Frequency (MHz)	Modulation technique	Sensitivity (dBm)	Theoretical bit rate (kbps)
CC1000 [89]	$300 - 1000$	BFSK	-110	76.8
TR1001 [71]	$868.15 - 868.55$	OOK, ASK	-104	115.2
CC1020 [90]	402 - 470 804 - 960	OOK, BFSK, GFSK	-118	153.6
CC2420 [91]	2400 - 2484	DSSS with OQPSK	-94	250

Table 2.1: Characteristics of the evaluated radio modules

usually several times higher than the frequency of the individual data symbols into the data signal.

Figure 2.1 shows the two most common **amplitude modulation** techniques used by radio modules for WSNs [18]. The depicted frequency (f_0) of the carrier wave ($z(t)$) is four times higher than the frequency of the data signal ($q(t)$). Every data symbol of the data signal is able to transmit a single bit.

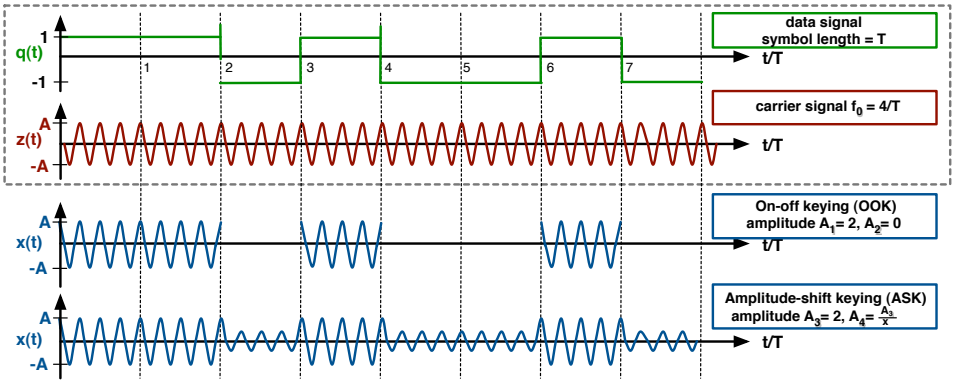


Figure 2.1: Digital amplitude modulation.

The most simple digital amplitude modulation technique is On Off Keying (OOK) [18]. Corresponding to the current data symbol in the data signal, the carrier wave is turned on or off. Another amplitude modulation technique is Amplitude Shift Keying (ASK) [18]. ASK is adapting the height of carrier wave's peak value in response to the data symbol.

Besides amplitude modulation, **frequency modulation** techniques are used. Frequency Shift Keying (FSK) [18] modulation techniques adapt the frequency of the carrier wave according to the data symbol. Binary Frequency Shift Keying (BFSK) [60], depicted in Figure 2.2, is the simplest FSK modulation technique. BFSK uses a pair of discrete frequencies to transmit the individual data symbols. The carrier wave changes the frequency of the carrier wave according to the current data sym-

bol.

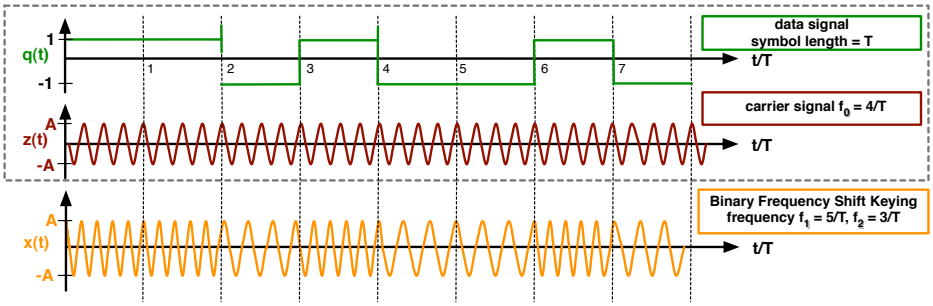


Figure 2.2: Digital frequency modulation.

Gaussian Frequency Shift Keying (GFSK) [60] is basically the same as BFSK. GFSK is offered by the CC1020 radio module for high bit rates. The higher the bit rate of the data signal is, the lower is the distance of the data signal frequency to the frequency of the carrier wave. This results in a problem called pulse shaping [18]. A result of high bit rates and a low frequency of the carrier wave are inter symbol interferences when the waveform of the carrier wave is switching between the individual frequencies. Gaussian filtering is one of the standards for reducing pulse shaping. Therefore, the data signal additionally passes a Gaussian filter before the FSK modulation is applied.
A further modulation technique is Phase Shift Keying (PSK) [18]. PSK is a **phase modulation** technique, where the phase of the carrier wave is changed according to the data symbol. Figure 2.3 shows the Offset Quadrature PSK (OQPSK) modulation technique [82], which is used by IEEE 802.15.4 [58] compliant radio modules.

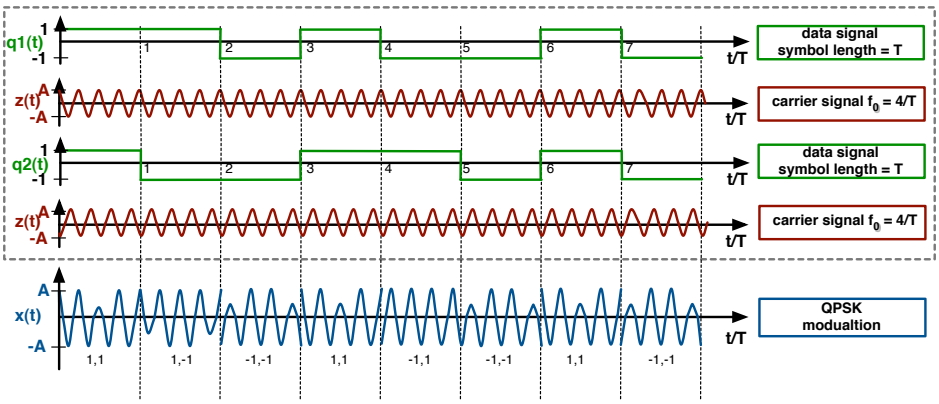


Figure 2.3: Offset quadrature phase-shift keying (OQPSK).

With OQPSK two data symbols can be modulated at the same time to the carrier wave. Therefore, the spectral efficiency [18] of OQPSK is twice the spectral ef-

ficiency of the introduced amplitude and frequency modulation techniques. The receiver compares the phase of the received signal to the carrier wave to retrieve the corresponding data symbol.

2.1.3 Spread Spectrum Techniques

Spread spectrum techniques are used to enhance the robustness of a transmission [18]. The data signal is spread in the frequency domain on purpose. This results in a data signal with a wider bandwidth.

A common spread spectrum technique used for bit/byte oriented radio modules is Manchester coding depicted in Figure 2.4. Manchester coding ensures sufficient transitions of the individual data symbols in the data stream. This is required due to the fact that a receiver is only able to detect a changeover in the waveform when the value of the transmitted data symbol changes. The changeover between two identical data symbols cannot be recognized by the radio receiver. The radio receiver has to measure the time period between the individual changeovers in the waveform to determine the amount of individually sent data symbols. Therefore, the clock drift of the radio modules can result in a synchronization problem if too many of the same data symbols are sent in a row. Manchester coding generates sufficient transitions in the data stream to ensure short periods of identical data symbols.

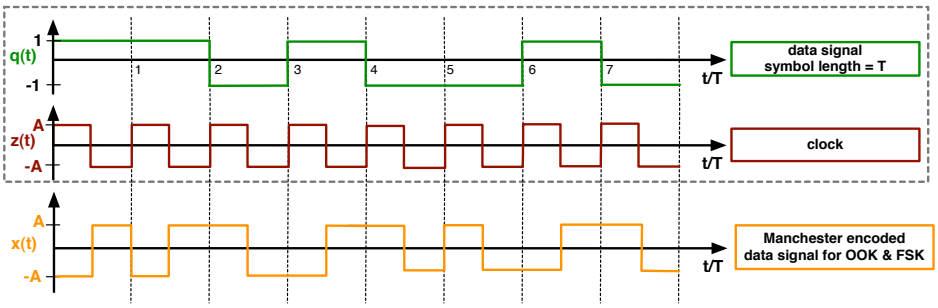


Figure 2.4: Manchester encoded data signal for OOK and FSK.

Packet oriented radio modules such as the CC2420 [91] offer significantly more advanced spread spectrum technique namely Direct Sequence Spread Spectrum (DSSS) [60]. With DSSS, the individual data symbols of the data stream are mapped to longer symbol sequences, e.g. the DSSS technique used by the CC2420 radio module defines 16 different 32 bit chipping sequences. DSSS takes four subsequent data symbols and maps them to the corresponding 32 bit chipping sequences. The 32 bit chipping sequences used by the CC2420 are defined by the IEEE 802.15.4 standard. The receiver maps the received 32-bit chipping sequence back to the four data symbols with the shortest hamming distance. By using the shortest hamming distance, bit errors can be recovered. The DSSS spreading mech-

anism enhances the signal to noise ratio on the channel. The resulting improvement is called **process gain** [60].

2.1.4 Radio Module CC1000

The CC1000 [89] is a low-power radio transceiver manufactured by Chipcon. In the Non Return to Zero (NRZ) mode, the CC1000 supports a bit rate up to 76.8 kbps, in Manchester mode up to 38.4 kbps. In contrast to the other radio modules, the CC1000 just provides digital transmission. The CC1000 enables a microcontroller to send and receive individual bits. Every other functionality, such as frame start detection, bit rate synchronization, or checksum verification is not provided by radio module itself and, therefore, has to be implemented by the microcontroller. This enables researchers to implement almost any protocol [59]. On the BTnode platform [11], using this radio module, the highest functional transmission speed is 38.4 kbps when using the Contiki operating system.

2.1.5 Radio Module TR1001

The TR1001 [71] transceiver is a low-power radio module designed for short-range wireless connections manufactured by RFM. The transmitter supports OOK and ASK modulation. OOK can be used up to a bit rate of 30 kbps. For higher bit rates up to 115.2 kbps, ASK has to be used. The TR1001 enables a microcontroller to send and receive individual bytes. Scheduling for sending the individual bytes has to be handled by the microcontroller. After receiving a byte, the radio module triggers an interrupt of the microcontroller to announce the reception of a single byte. The operating frequency of the TR1001 is between 868.15 and 868.55 MHz. The TR1001 uses an analog implementation of a Finite Impulse Response Filter (FIR) filter [60] to reduce the interfered in the received signal. In Chapter 3 we compare the receiving characteristics of TR1001 using a FIR filter with the other radio modules.

Finite Impulse Response Filter

An electronic filter is able to change the phase and the amplitude of an electric wave depending on the frequency. This enables suppressing or attenuating unwanted frequencies. In mobile applications with a carrier wave frequency below 3 GHz, FIR filters are used. FIR filters enable out-of-band rejection [16] to reduce interferences. The TR1001 uses a Surface Acoustic Wave (SAW) filter [16] as an analog implementation of a FIR filter.

A SAW filter converts an electrical signal to a mechanical wave using a piezoelectric crystal. The mechanical wave shows a delayed propagation within the piezoelectric crystal. The delayed outputs of the mechanical wave are converted back to an electrical signal and recombined to implement the FIR filter. The FIR

filter convolutes the digital input signal with its impulse response. Figure 2.5 shows a FIR filter:

- Input signal: $S_{in}(t)$
- Output signal: $S_{out}(t)$
- Delay elements: τ
- Filter coefficients: b_x (also known as amplitude adapters)
- Filter order: n (with $n + 1$ terms on the right-hand side)

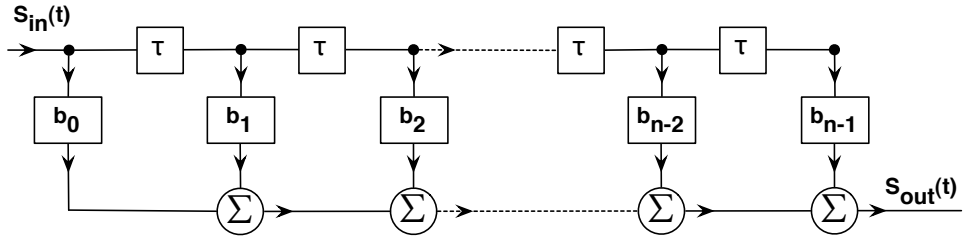


Figure 2.5: Finite impulse response filter.

The output $S_{out}(t)$ is calculated by convolving the input signal $S_{in}(t)$ with the filter coefficients b. The result is a weighted sum of the current and a finite number of previous values of the input. The operation can described by the following equation: $S_{out}(t) = a_0 * S_{in}(t) + a_1 * S_{in}(t - \tau) + a_2 * S_{in}(t - 2 * \tau) + a_{n-1} * S_{in}(t - (n - 1) * \tau) = \sum_{k=0}^{n-1} a_k * s_{in}(t - k * \tau)$.

2.1.6 Radio Module CC1020

The CC1020 [90] is a low-power radio transceiver manufactured by Chipcon. It can be programmed to operate in the $402 - 470$ and $804 - 960$ MHz band. It supports a bit rate up to 153.6 kbps. The CC1020 can be configured to use OOK, BFSK or GFSK modulation. It also offers a digital Received Signal Strength Indicator (RSSI) and a carrier sense indicator to the microcontroller. Similar to the TR1001, the CC1020 enables a microcontroller to send and receive individual bytes. The microcontroller has to schedule the transmission of the individual bytes. After a byte has been received, the CC1020 triggers an interrupt to instruct the microcontroller to read the received byte.

2.1.7 Radio Module CC2420

The CC2420 [91] is an IEEE 802.15.4 compliant low-power radio transceiver. It operates in the 2.4 GHz Industrial, Scientific and Medical (ISM) radio band. It

provides extensive hardware support for services, such as packet handling, autonomous link layer acknowledgments, data buffering, data encryption and authentication, Clear Channel Assessment (CCA), RSSI and Link Quality Indication (LQI). For transmitting data, the microcontroller has to copy a complete packet to the CC2420 *TX buffer*. To utilize the services provided by the CC2420 radio module, the frame format of the packet has to be compliant to IEEE 802.15.4. The following subsections introduce the IEEE 802.15.4 standard and its packet handling within the CC2420.

IEEE 802.15.4

The IEEE 802.15.4 [58] standard specifies the physical and link layers for Wireless Personal Area Networks (WPAN). It is maintained by the IEEE 802.15 working group. In contrast to Wireless Local Area Network (WLAN), WPAN is designed for shorter distances, i.e. distances of centimeters to a few meters to interconnect personal devices located in a person's immediate vicinity. IEEE 802.15.4 is used by protocol stacks, such as ZigBee [103] , ISA100.11a [46], WirelessHART [35], and the MiWi [102] specifications.

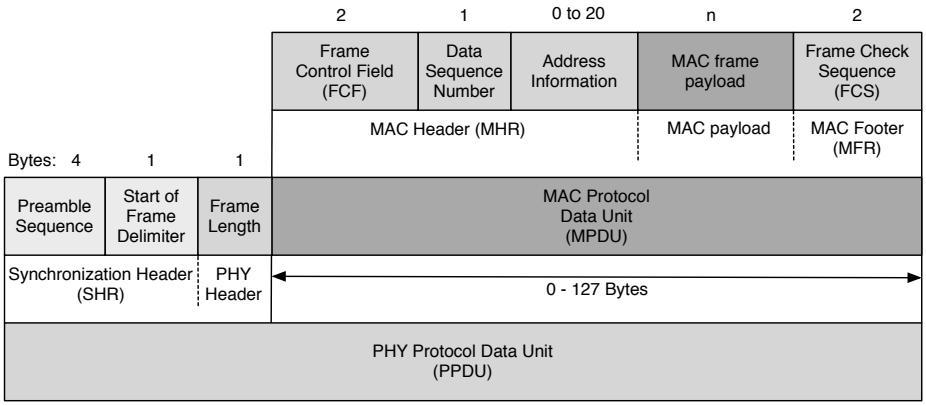


Figure 2.6: IEEE 802.15.4 frame.

Figure 2.6 shows the IEEE 802.15.4 data frame structure. An IEEE 802.15.4 frame starts with a 5 bytes Synchronization Header (SHR) and the PHY Header. The SHR is a 4 bytes preamble sequence and an 1 byte Start of Frame Delimiter (SFD). A receiver requires the preamble sequence bytes (0x00) to synchronize the demodulator and searches for the SFD byte (0x7A) to determine the position of the 1 byte PHY header. The PHY header provides the frame length of the MAC Protocol Data Unit (MPDU). The MPDU consists of the MAC Header (MHR), the MAC payload and the MAC Footer (MFR). Depending on usage of 16-bit or 64-bit addresses, the biggest MAC payload is 118 or 106 bytes, respectively. The MFR includes a 2-bytes Frame Check Sequence (FCS), which is a CRC-CCITT 16-bit checksum of the MPDU (CCITT stands for Comité Consultatif International Téléphonique

et Télégraphique). The maximum frame length, including all the headers is 133 bytes. It requires 4.256 ms for transmitting a frame with maximum length.

The MHR contains the Frame Control Field (FCF), the data sequence number and the address of the sender and receiver nodes. Figure 2.7 shows the FCF in more detail. The FCF provides information about the frame, such as the frame type, usage of 16-bit or 64-bit addresses and frame handling. For example, the acknowledgment request flag informs the receiver if an automatic acknowledgment request has to be sent.

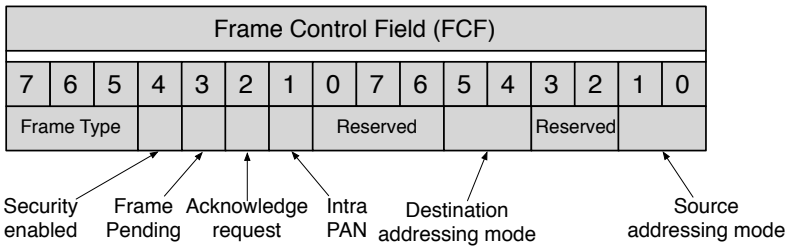


Figure 2.7: Frame Control Field (FCF).

Figure 2.8 shows the IEEE 802.15.4 acknowledgment frame structure. The acknowledgment frame has a fix length of 11 bytes. The first 5 bytes of the acknowledgment frame are part of the physical layer. They are used as synchronization header to enable frame start detection of the receiver. The remaining 6 bytes are part of the link layer. An incoming acknowledgment frame provides neither address information about the sender nor the addressed receiver. Only the data sequence number can be used to allocate the received acknowledgment to the previously sent data frame.

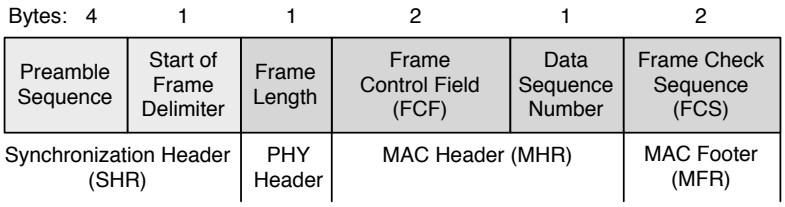


Figure 2.8: IEEE 802.15.4 acknowledgment frame.

CC2420 Packet Receiving

Receiving packets with a packet oriented radio differs from bit- and byte oriented radio modules. Microcontrollers connected to bit or byte oriented radios constantly have to read every received bit respectively byte directly after its reception from the radio to a local buffer. Using packet oriented radio modules, the radio

module autonomously handles the detection, receiving and verification of a frame autonomously without any interaction of the microcontroller. The radio module triggers an interrupt of the microcontroller, after the packet has been received completely. This enables the microcontroller to copy the RX buffer of the radio module to the local buffer.

The CC2420 offers three pins to announce the reception of a packet to the microcontroller. The output pins are depicted in Figure 2.9. One is the SDF pin, which goes high after the SFD field of the IEEE 802.15.4 frame header has been completely received. The SFD pin goes low again after the last byte of the MPDU has been received. This makes it possible to detect ongoing transmissions by the microcontroller. The FIFO pin usually goes high when the length byte of the frame header has been completely received. The FIFOP pin will go high when the last byte of a new packet is received. In receive mode, the FIFOP pin can be used to inform the microcontroller about a received frame in RX buffer. Therefore, the FIFOP pin has to be connected to an interrupt of the microcontroller.

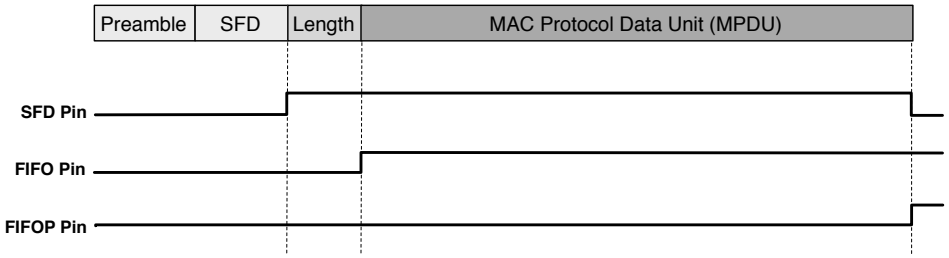


Figure 2.9: Output pin activity during receive.

Figure 2.10 depicts the operations of an energy efficient link layer protocol receiving a frame with the CC2420 radio module. The radio module is turned off most of the time to preserve energy. The link layer protocol periodically turns on the radio module and switches it into listening mode. In listening mode the CC2420 autonomously detects any incoming packet by the predefined synchronization header. After recognizing the synchronization header, the SFD pin is set to high. The SFD pin can be used by microcontroller to recognize an incoming frame if required. The radio module copies every incoming byte to its RX buffer. After the frame

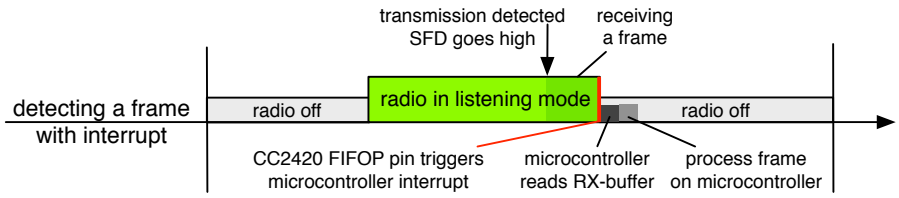


Figure 2.10: Data frame detection by using the FIFOP interrupt.

reception has been completed, the CC2420 FIFOP pin triggers an interrupt of the microcontroller. This enables the microcontroller to turn off the radio and read the received data from the RX buffer for further processing.

CC2420 Packet Transmission

For transmitting a frame, the microcontroller has to copy the frame to the TX buffer of the CC2420 first. Then the microcontroller is able to send a command to the radio module to initiate the transmission of a frame. The command named *STXON* can be used to directly send a frame. It enables and calibrates the frequency synthesizer and directly transmits the content of the TX buffer. The command *STXONCCA* can be used to additionally perform a channel check before sending the frame. If the channel is busy, the frame is not transmitted. Usually *STXONCCA* is used to send a data frame, while *STXON* is used to send a software acknowledgment.

CC2420 Channel Checking Support

A channel check basically fulfills two different requirements. First, before sending a packet, the transmitter has to ensure that the channel is free. Second, the channel has to be scanned for incoming data packets and acknowledgments. To perform a channel check, the CC2420 offers the so-called Clear Channel Assessment (CCA) function. The CCA function supports three different modes:

- **Mode 1:** The channel is free if the currently received energy is below a predefined RSSI threshold.
- **Mode 2:** The channel is free if no valid IEEE 802.15.4 data is currently received.
- **Mode 3:** The channel is free if the energy is below threshold and no valid IEEE 802.15.4 data is currently received.

The output of the CCA function is offered by the digital interface of the CC2420 on the CCA output pin. The value of the pin is valid when the receiver has been enabled for at least a 4 bytes period (128 μs). A sleeping node has to turn on the radio for at least 128 μs before performing a channel check. The CC2420 also shows at the SFD output pin if a synchronization header has been detected. All our protocol implementations use the CCA in Mode 1. CCA in Mode 1 enables a receiver to detect an ongoing transmission even if the radio module has been turned on only after the synchronization header has been already transmitted or if the synchronization header has been corrupted by interferences.

CC2420 Auto Acknowledgment Support

The CC2420 includes hardware support for sending IEEE 802.15 acknowledgment frames. If the so-called AUTOACK function is enabled an acknowledgment is transmitted every time:

1. The incoming frame passed the address recognition.
2. The *acknowledge request* flag in the FCF of the incoming frame is set to 1.
3. The incoming frame passed the cyclic redundancy check (CRC) check.

Then, the radio module creates an IEEE 802.15 compliant acknowledgment in TX buffer by copying the required values from the header of the received frame. Concerning the IEEE 802.15 standard, the transmission of the acknowledgment is started 192 μs after the last byte of the data frame has been received. Figure 2.11 shows the corresponding time periods.

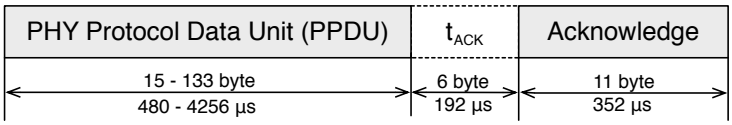


Figure 2.11: AUTOACK timing.

When using the AUTOACK function (hardware acknowledgment) of the CC2420 radio module, the sender of a data frame can rely on an exactly timed incoming acknowledgment. The AUTOACK function releases the microcontroller from creating and copying the acknowledgment frame to the radio module. On the other side, if the acknowledgment is handled by the microcontroller (software acknowledgment), the frame format of the acknowledgment can be customized. For example, additional information about the current network conditions can be added to the acknowledgment. Unfortunately, the software acknowledgment requires significantly more execution time than hardware acknowledgment. Table 2.2 shows the required time periods that we measured for different actions related to sending an acknowledgment with a telosB node. t_{ACK} refers to the time period between the last byte of the data frame has been received and the start of the acknowledgment transmission.

Action	required time (μs)
Microcontroller writes/reads one byte	11.5
Radio module sends/receives one byte	32.0
t_{ACK} hardware acknowledgment	192
t_{ACK} software acknowledgment	335 - 1800 (and more)

Table 2.2: Time periods of different actions on the telosB node

The duration of t_{ACK} when using software acknowledgments depends on the length of the sent frame and the required processing. A hardware acknowledgment on the contrary requires for every transmitted frame length always the same response time. Moreover, the required response time for a hardware acknowledgment is clearly shorter than a software acknowledgment could ever be.

Interferences in the 2.4 GHz ISM band

The IEEE 802.15.4 standard defines several different frequency bands to be used. If using the 2.4 GHz ISM band, IEEE 802.15.4 shares some of its available channels with IEEE 802.11 [43, 44] and IEEE 802.15.1 (Bluetooth) [45]. Using IEEE 802.15.4 side by side with Bluetooth is not a problem, especially since the Bluetooth working group introduced the adaptive frequency-hopping spread spectrum (AFH) in version 1.2 of the Bluetooth specification in November 2003. AFH improves resistance to interferences by avoiding the use of occupied frequencies in the hopping sequence. In contrast to Bluetooth, IEEE 802.11 can cause significant bit errors in co-located IEEE 802.15.4 networks [53]. An IEEE 802.11 radio transmitter uses a transmission power that is up to 14 dBm allocated on 20 MHz bandwidth. IEEE 802.15.4 usually uses 0 dBm on 3 MHz bandwidth. Figure 2.12 shows the used frequencies band of IEEE 802.11 and IEEE 802.15.4

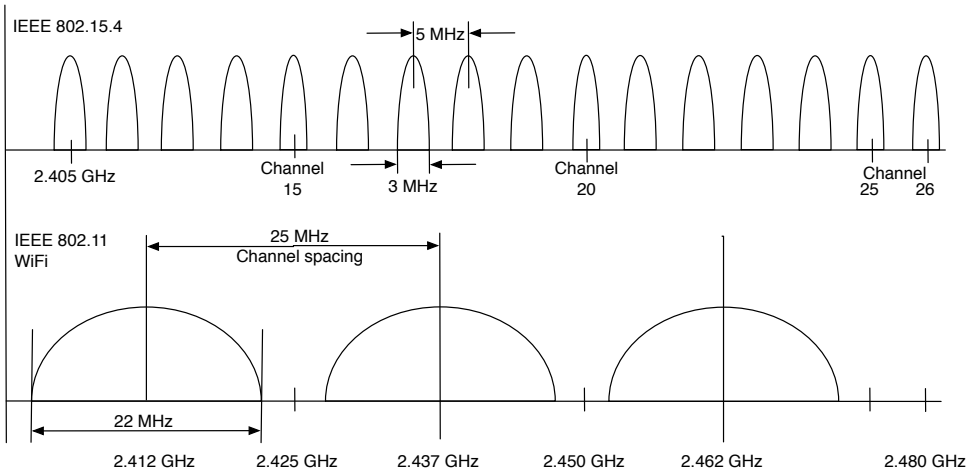


Figure 2.12: 2.4 GHz frequencies of IEEE 802.11 and IEEE 802.15.4.

The IEEE 802.15.4 channels 25 and 26 are outside of the most frequently used IEEE 802.11 access points. Table 2.3 shows different packet lengths of IEEE 802.11 and IEEE 802.15.4 packets. The packets sent by IEEE 802.11 transmitters are considerably shorter than IEEE 802.15.4 packets.

Parameter	802.15.4 [μs]	802.11.b [μs]	802.11.g [μs]
Minimal length	352	202	194
Maximal length	4'256	1'906	542

Table 2.3: Packet length of IEEE 802.11 and IEEE 802.15.4 packets.

The authors of [53] analyzed the IEEE 802.11 interferences in low-power WSNs using a ZigBee network stack. The ZigBee [103] network stack uses IEEE 802.15.4

on the physical and link layer. By using several different optimizations, they improved the ZigBee network delivery rate by 70%. They experimentally found that the location of the IEEE 802.11 transmitter has significant influence on the bit error pattern distribution in an IEEE 802.15.4 frame. Depending on the used IEEE 802.11 transmitter and the distance to an IEEE 802.15.4 transmitter, they split the interference domain into *symmetric* and *asymmetric* regions. In *asymmetric* regions, the IEEE 802.15.4 transmission power is too low to be detected by an IEEE 802.11 transmitter. The observed bit errors are often evenly distributed over the entire IEEE 802.15.4 packet. In this case, the authors of [53] propose to use Forward Error Correction (FEC) to reduce the packet loss. FEC codes do not include any retransmission, they just enhance the probability of a successful transmission. A FEC code will fail in case of a missed frame start or if the received payload contains too many bit errors. Section 2.5.4 introduces the used FEC codes in more detail.

In *symmetric* regions, IEEE 802.15.4 transmissions can be detected by the IEEE 802.11 transmitter. In such a case, an IEEE 802.11 transmitter backs off and schedules a retransmission. The authors of [53] observed in *symmetric* regions that bit errors mainly happen in the IEEE 802.15.4 synchronization header. Therefore, numerous frames are not detected by the receiver. For *symmetric* regions, the authors introduced so-called multi-headers. Figure 2.13 shows an IEEE 802.15.4 frame with using these multi-headers. The MPDU of the IEEE 802.15.4 contains a second synchronization, PHY and MAC header. If the synchronization header has been correctly received, the second header can be removed. If the first synchronization header was corrupted, maybe the second one can be detected successfully.

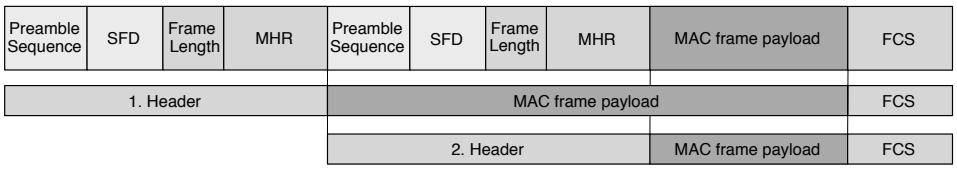


Figure 2.13: IEEE 802.15.4 Multi-Headers.

The CRC checksum in the FCS field is only valid is the first header was correctly received. If the first header was corrupted but the second header could be received successfully, then the CRC in FCS field is not valid for this frame. Therefore, the AUTOACK function of the CC2420 cannot be used to autonomously send acknowledgments if the second header was received successfully.

2.2 Evaluated Sensor Node Platforms

This section briefly introduces the different sensor node platforms equipped with the radio modules that were mentioned in Section 2.1. The highest bit rate of the radio module depends on the used microcontroller. Table 2.4 shows a brief overview

about the five different evaluated sensor node platforms and corresponding radio modules.

Node	Radio	Highest physical bit rate (kbps)	Bit rate including spreading (kbps)
ESB	TR1001	38.4	19.2
MSB430	CC1020	38.4	19.2
BTnode	CC1000	38.4	19.2
telosB	CC2420	2000	250
MicaZ	CC2420	2000	250

Table 2.4: Evaluated sensor node platforms and corresponding radio modules

The column *highest physical bit rate* shows the highest reasonable working transmission speed on the physical layer. The column *bit rate including spreading* shows the bit rate on the link layer before applying spectrum spreading. For the bit/byte oriented radio modules, i.e. TR1001, CC1000 and CC1020, the spreading has to be done by the microcontroller.

In addition to the radio module and microcontroller, the sensor node platforms differ in the attached sensors and components for debugging. The following Subsections 2.2.1 - 2.2.5 introduce the different sensor node platforms.

2.2.1 Embedded Sensor Node Developed at FU Berlin

Figure 2.14 shows an Embedded Sensor Board (ESB) [75], which has been developed and released by the Freie Universität Berlin and the spin-off company Scatter-Web GmbH [76] in 2002.

Figure 2.14: Embedded Sensor Node Developed at FU Berlin.

The ESB consists of an MSP430 F149 low-power microcontroller [88] from Texas Instruments with 2kB RAM and 60kB+256B flash ROM. A TR1001 radio transceiver [71] is used as radio module. On the ESB platform, the highest functional transmission speed of the TR1001 is 38.4 kbps. Most real world implementations for the ESB node use a bit rate of 9.6 kbps.

A RS232 port and a JTAG port are available for connecting the microcontroller to a development environment. Operating elements for diagnostics are a beeper, two buttons and three LEDs. The ESB node offers the following onboard-sensors:

- IR movement PIR sensor with Fresnel lens
- IR transmitting diode
- IR receiving diode
- Microphone
- Movement detector CM 1800
- Temperature sensor
- Potentiometer

With the IR transmitting and receiving diodes, the ESB nodes provide an additional ability of communication. The microphone offers the possibility monitor to environmental noise. The ESB nodes cannot only detect its own movement but can also monitor movements in front of the device with the integrated IR sensor and its Fresnel lens. This makes it possible to implement different movement detection approaches including alarming.

2.2.2 Modular Sensor Board 430 Developed at FU Berlin

Figure 2.15 shows the Modular Sensor Board 430 (MSB430) [8]. It has been developed and released by the Freie Universität Berlin and the spin-off company Scatter-Web GmbH [76] in 2005.

The MSB430 consists of a MSP430 F1612 low-power microcontroller [88] from Texas Instruments with 5kB RAM and 55kB+256B flash ROM. A CC1020 radio transceiver [90] from Chipcon is used as radio module. On the MSB430 platform, the highest functional transmission speed is 38.4 kbps. Most real world implementations for the MSB430 node use a bit rate of 9.6 kbps. The only operating element for feedback is a red LED. The MSB430 node comes with an integrated battery case for 3xAA cells and offers the following onboard-sensors:

- SHT 11 temperature and humidity sensor
- MMA7260QT 3-axis-accelerometer

Figure 2.15: Modular Sensor Board 430.

2.2.3 BTnode Developed at ETH Zurich.

Figure 2.16 shows a BTnode [11]. BTnodes have been developed at the Eidgenössische Technische Hochschule (ETH) Zürich by the Computer Engineering and Networks Laboratory (TIK) and the research Group for distributed systems. The BTnode rev3 is a dual radio device with a CC1000 low-power radio module

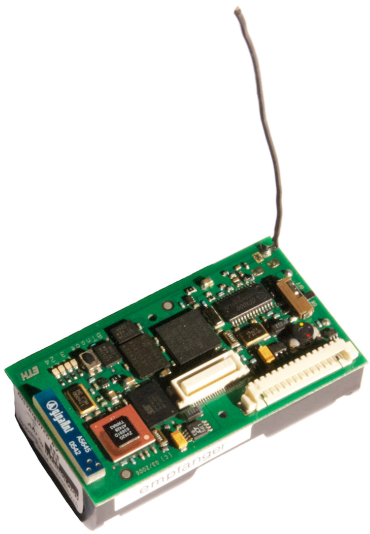


Figure 2.16: BTnode rev3.

and a Zeevo ZV4002 Bluetooth radio module. The Bluetooth system supports up to four independent piconets and seven slaves. The BTnode can operate both radios simultaneously or shut them down independently when not in use. The sensor node platform comes with an integrated battery case for 2xAA cells and extension connectors. Operating elements for diagnostics are four LEDs. The microcontroller is an Atmel ATmega 128L with 64+180 Kbyte RAM, 128 Kbyte FLASH ROM and 4 Kbyte EEPROM.

A brand-new BTnode comes without an attached antenna for the CC1000 radio module. Under this condition the communication range of a BTnode is below 1 m. Therefore we added a $\lambda/4$ monopole antenna to all our BTnodes.

2.2.4 TelosB Mote Platform Developed by UC Berkeley

Figure 2.17 shows a telosB [10] node from Crossbow [17], designed at the University of California, Berkeley. It is identical in construction to the TmoteSky node from Sentilla [78]. The telosB node consists of a MSP430 F1611 low-power microcontroller [88] from Texas Instruments with 10kB RAM, 48kB+256B flash ROM and 1024k serial storage, on-board humidity, temperature and light sensors. A CC2420 radio transceiver [91] from Chipcon is used as radio module. Operating elements for diagnostics are a button and three LEDs.

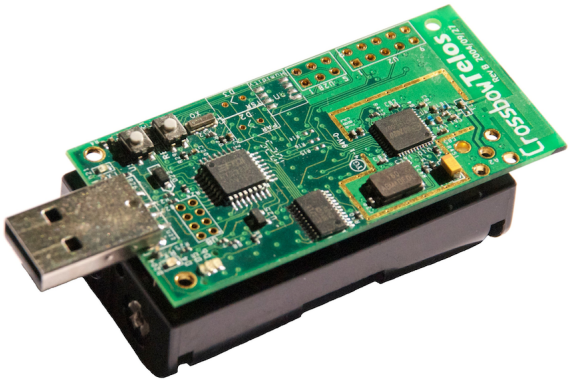


Figure 2.17: TelosB sensor node.

2.2.5 MicaZ Developed by UC Berkeley and Crossbow

Figure 2.18 shows a MicaZ [10] node from Crossbow [17]. The MicaZ node consists of an Atmega 128L microcontroller [6] from Atmega with 4kB RAM and 128kB flash ROM. A CC2420 radio transceiver [91] from Chipcon is used as radio module. Operating elements for diagnostics are three LEDs.

Figure 2.18: MicaZ sensor node.

2.3 Operating Systems for Sensor Nodes

An operating system for sensor nodes handles the resources of a microcontroller and offers useful services to WSN applications. Handling the resources of a microcontroller includes scheduling and prioritizing of processes as well as memory, interrupt and power management functions. Most operating systems for sensor nodes consist of operating system kernel, which includes communication services besides the normal microcontroller resource handling and user space services suitable for WSN applications. The remaining subsections introduce the two operating systems for WSNs used in this thesis, namely Contiki [23] and ScatterWeb [76].

2.3.1 Contiki

Contiki [23] is a free and open source operating system targeting embedded systems. It has been developed by the Swedish Institute of Computer Science. Contiki is a multitasking enabled operating system kernel and communication services. Contiki is written in the standard C programming language. It supports several microcontroller architectures, including the MSP430 from Texas Instruments and the Atmel ATmega. All sensor node platforms introduced in Section 2.2 are equipped with one of these two microcontroller architectures. Additionally, Contiki supports energy preserving mechanisms, such as switching the microcontroller into low-power mode during idle periods.

Figure 2.19 gives an overview about the Contiki network stack located in user space and the communication services of the operating system kernel. The application layer featuring the WSN application is located in user space, while the underlying network layers providing μIP [21] and RIME [25], are part of the communication services. The μIP network stack supports TCP/IP, UDP, ICMP and SLIP. The RIME network stack interconnects the μIP with the link layer and pro-

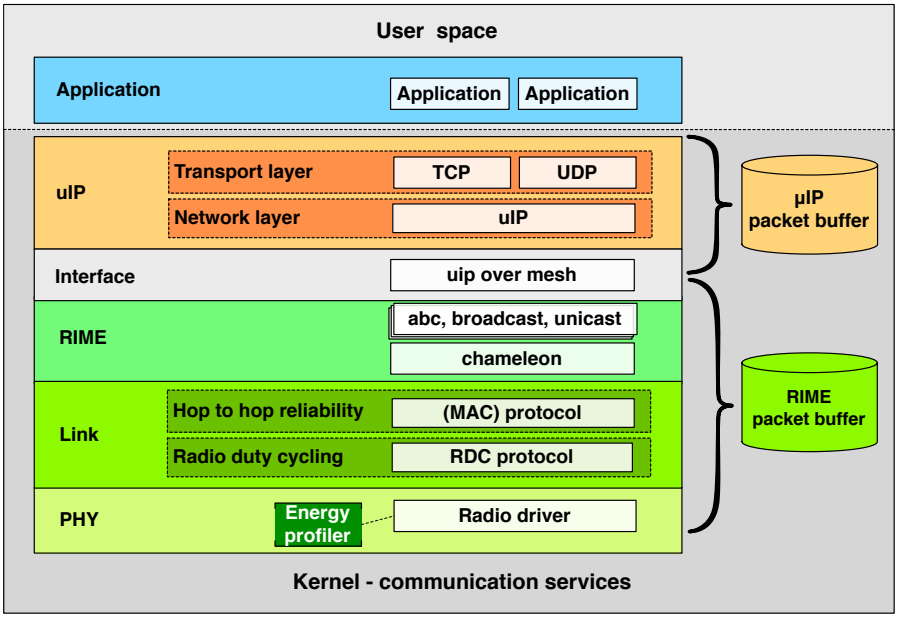


Figure 2.19: Contiki network stack.

vides numerous options to forward a packet to the next hop. The individual layers and functions of the network stack are discussed in more detail in the following:

WSN Applications in Contiki

WSN applications are executed on top of the Contiki network stack in the application layer. With Contiki the individual WSN applications are running in user space, separated from the operating system kernel and communication services. Contiki WSN applications are individually loadable and also removable during runtime. The individual WSN applications are able to communicate with each other by posting events through the operating system kernel.

μIP Network Stack in Contiki

To send messages to other nodes in an IP network, a session socket for TCP/IP respectively a datagram socket for UDP/IP packets can be established by an application. μIP handles all the required TCP/IP and UDP functionality and adds the required protocol headers to the WSN application payload. Additionally, μIP handles the forwarding of IP packets on intermediate nodes.

Interface Layer

The interface layer copies and converts the packets from the μIP packet buffer to the RIME packet buffer and vice versa. If no IP support is required, then the in-

terface layer is used to directly connect the RIME network stack to the application layer. Additionally, the interface layer queries the routing mechanism to get the next address of the next hop.

Contiki RIME Network Stack

The RIME network stack provides the RIME packet buffer to queue multiple packets. Every stored packet consists of a buffered packet and meta-data, such as routing information, reliability requirements or sequence numbers. Modifications to the frame headers are directly executed in the RIME packet buffer.

The RIME network stack provides several very thin layers to build a customized communication stack connecting the overlaying network protocol with the link layer. Every individual RIME layer is responsible to modify one part of the meta-data of the handled packet in the RIME packet buffer. The RIME layers do not directly modify the frame headers, this is done by the link layer protocols. RIME follows an aspect-oriented implementation approach. Different thin layers are solely responsible for single aspects. RIME provides the following thin layers to modify the meta-data in the RIME packet buffer:

- Anonymous best effort single-hop broadcast (abc)
- Identified best effort single-hop broadcast (ibc)
- Stubborn identified best effort single-hop broadcast (sibc)
- Best-effort single-hop unicast (uc)
- Stubborn best effort single-hop unicast (suc)
- Reliable single-hop unicast (ruc)
- Unique anonymous best effort single-hop broadcast (uabc)
- Unique identified best effort single-hop broadcast (uibc)
- Best effort multi-hop unicast (mh)
- Best effort multi-hop flooding (nf)
- Reliable multi-hop flooding (trickle)

For example, with μIP the layers abc, ibc and uc are used. The uc layer copies the address received from the routing mechanism to the RIME packet buffer. The ibc layer copies the address of the sender and abc layer adds the sequence number to the RIME packet buffer. Chameleon adds a two-byte header to the frame to define the used arrangement of the individual RIME layers. The receiving node uses this information to determine the sequence of the individual thin RIME layers required for this packet.

Contiki Link Layer

The link layer of Contiki is split into two sub-layers. The upper one, called *MAC* sub-layer, is responsible for the hop-to-hop reliability by using local retransmissions. The lower one, the so-called *Radio Duty Cycle (RDC)* layer, handles the sleep cycles of the radio module. Both sub-layers make use of the RIME packet buffer to handle the payload and to add the required frame headers. Depending on the addressing of the received packet, the payload is either dropped or forwarded to the RIME network stack.

Radio Driver and Energy Profiler

The radio driver handles all switches of the radio module between sleep, listen and transmission mode. To transmit a packet, the radio driver copies the frame stored in the RIME packet buffer to the radio module and controls its transmission. Received bytes respectively packets are directly written to the RIME buffer and then delegated to the link layer.

The radio driver of Contiki supports a software based energy profiler [28] to estimate the energy consumption of a single sensor node. The total energy consumption of the radio module (E_{radio}) is estimated by the following equation:

$$E_{radio} = V_{bat}\Big((I_{sleep} \cdot t_{sleep}) + (I_{tx} \cdot t_{tx}) + (I_{listen} \cdot t_{listen})\Big)$$

The power supply voltage (V_{bat}) as well as the current draw in sleep mode (I_{sleep}), during transmission (I_{tx}) and listening (I_{listen}) is considered to be static. The individual current draw values corresponding of the evaluated radio module have to be taken either from the radio module manual or determined with an ampere meter, such as the RIGOL multimeter. The time periods t_{sleep}, t_{tx}, t_{listen} are determined by the energy profiler located in the radio driver module. The radio driver module handles the switches between the individual radio modes. The energy profiler updates the length of the individual time periods, every time the radio module is switched into a new state.

Contiki Summary

Contiki is one of the most widely used operating systems for WSNs. The network stack of Contiki supports several communication standards such as TCP/IP, UDP/IP and IEEE 802.15.4 to enhance network connectivity in heterogeneous WSNs. The individual network layers enable a modular development of different protocols supporting energy aware hop-to-top and end-to-end reliability mechanisms. The integrated energy profiler enables the concurrent energy estimation of every sensor node in a WSN.

2.3.2 ScatterWeb

ScatterWeb [76] is a free and open source operating system designed for sensor node platforms developed by Scatter-Web GmbH [76]. ScatterWeb is written in the standard C programming language and divided into two parts. The first part holds the firmware featuring the operating system kernel and rather simple communication services. The second part holds the individual WSN applications, which can neither be loaded nor stopped during runtime in ScatterWeb.

The operating system kernel is purely event driven and does not support preemptive multitasking. ScatterWeb keeps the microcontroller most of the time in a low-power mode to reduce the energy costs. The microcontroller periodically wakes up from low-power mode by using a hardware interrupt. Then, the event handler executes all required tasks. After all tasks have been completed, the microcontroller reenters the low-power mode again. Figure 2.20 depicts the energy consumption of an MSB430 sensor node running ScatterWeb and Contiki. The radio module and attached sensor devices are turned off during measurement. The difference in energy consumption is caused by energy preserving mechanisms used to put the microcontroller into low-power mode. The Figure shows that the power cycling mechanism for the microcontroller used by Scatterweb requires around twice the energy compared to the mechanism used by Contiki.

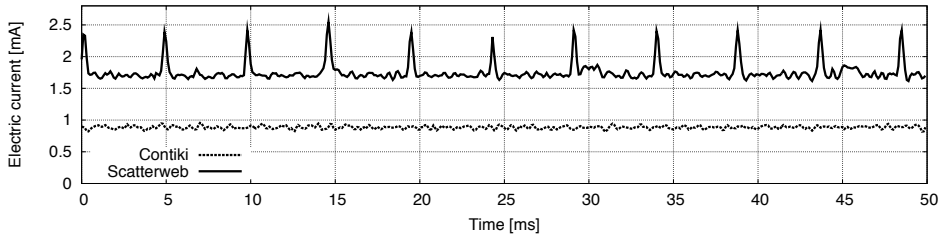


Figure 2.20: Energy consumption of a MSB430 node using Contiki and Scatterweb.

ScatterWeb does not support a layered network stack such as Contiki. The radio driver, link and network layers do neither offer a well-defined interface to each other nor to the WSN applications. The used communication protocols do not comply with any common protocol standard.

2.4 Evaluation Tools

This section introduces the different evaluation tools that we used to analyze and compare the performance of our contributed WSN communication stack to already existing ones. Additionally, the evaluation tools are used during the development phase to detect and analyze common problems and improve the recognized weaknesses. The used evaluation tools can be divided into three classes:

- **Network simulators:** WSNs may consist of hundreds or thousands individual nodes distributed over a large area. Building and maintaining such a large WSN is costly in terms of time and money. Moreover, it is rather impractical to evaluate new mechanisms for WSN applications and communication protocols in such large real world WSNs. For example, reliable distribution of new communication protocols as well as collecting the required evaluation data depict two changeling problems in a real world world WSN testbed. Network simulators provide a solution for this problem by enabling inexpensive and flexible setups of large WSNs. A simulated network can be reseted at any time if the new software results in an erroneous WSN condition. Further advantages of network simulators are, for example, exact repeatability of challenging network conditions, visualization tools showing the behavior of the individual sensor nodes, and tools collecting any kind of evaluation data.

  However, for meaningful results, the network characteristics have to be carefully modeled. For example, the evaluation of network protocols requires realistic radio models to simulate the complex interrelation of individual wireless transmissions, interferences and wave propagation effects.

- **Real world WSN testbeds:** Different research facilities built real world WSN testbeds to evaluate WSN applications and communication protocols in a large real world WSN. Examples for real world WSN testbeds are the different WISEBED WSN testbeds [79] funded by the European-Union, MoteLab [99] the testbed of Harvard University [62], TWIST [34], the testbed of Technical University of Berlin [93] and Kansei [30] at Ohio State University. Usually each sensor node in such a real world WSN testbed is connected over a serial connection to a controlling unit. The controlling unit offers services to upload new software images and to monitor the individual sensor nodes to collect statistical data.

- **Energy measurment equipment:** Energy efficiency is an important field of current WSN research. Digital multimeters [48] supporting digital ammetry [48] are able to frequently measure and record current draw. The individual recorded measurements build an *energy profile* as depicted in Figure 2.21. The used energy (E) is determined by the integral of the recorded electrical current ($I(t)$) multiplied with the voltage (U) over time (t):

$$E = \int_{t_0}^{t_1} I_{measured}(t) \cdot U_{battery} \cdot \mathrm{d}t$$

  The voltage $U_{battery}$ is assumed to be static according to the power supply voltage of the corresponding sensor node.

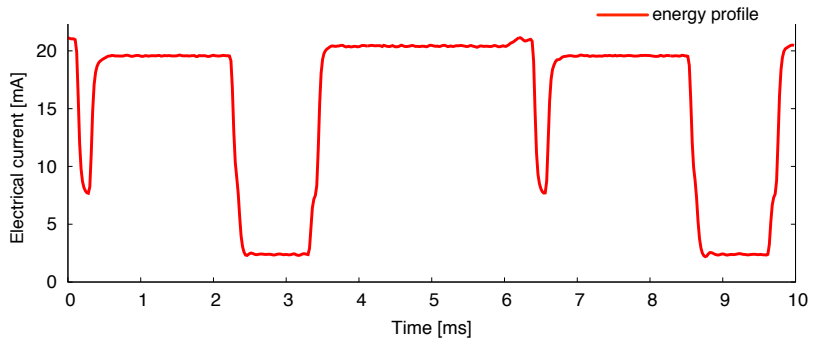


Figure 2.21: Energy profile of a digital amperemeter.

2.4.1 OMNeT++ Network Simulation Framework

As a respective of network simulators, the following subsection describes the Objective Modular Network Testbed (OMNeT++) Network Simulation Framework [2], which we have used in our evaluations. OMNeT++ is a modular open-source network simulator. The OMNeT++ core, the Graphical User Interface (GUI) and analysis tools are completely written in C++ programming language. Individual modules contain core parts written in C++, which are then interconnected by the high-level language Network Description (NED). It is used to define the network topology as well as to attach different network protocol modules to a network stack. Strong points of OMNeT++ are its GUI support, the clean design and the straightforward simulation development process. Additionally, it benefits of several freely available OMNeT++ extensions available such as the INET Framework [95] including UDP, TCP and IP support or the Castalia project [7] supporting IEEE 802.15 protocols.

Radio Wave Propagation Models

We use a free space model for direct waves, parameterized for the physical characteristics of the CC2420 radio module using an OQPSK modulation of a 2.4 GHz carrier wave. We do not consider wave propagation effects such as reflections, multi-path fading or diffraction [81]. In the free space model, the receiving power (P_{Rdb}) in decibel is calculated by using the transmitting power (P_{Tdb}) in decibel and the distance (d_{TR}) between the sender and the receiver:

$$P_{Rdb} = P_{Tdb} + 20log_{10}\left(\frac{\lambda}{4 \cdot \pi \cdot d_{TR}}\right)$$

To calculate the probability of a bit error, one has to be calculated the Signal-to-Noise Ratio (SNR). The signal strength corresponds to the calculated receiving power (P_{Rdb}). The total noise is calculated by summing up the receiving power of all other concurrently ongoing transmissions, the thermal noise and the receiver

noise. The resulting SNR determines the probability of a bit error. The error probability corresponding to the calculated SNR value is taken from Castalia [7] and real world measurements [52].

Figure 2.22 shows a graphical illustration of the bit error probability, without concurrently ongoing transmissions in the simulated area. The depicted values correspond the characteristics of a CC2420 radio module equipped with an omnidirectional antenna.

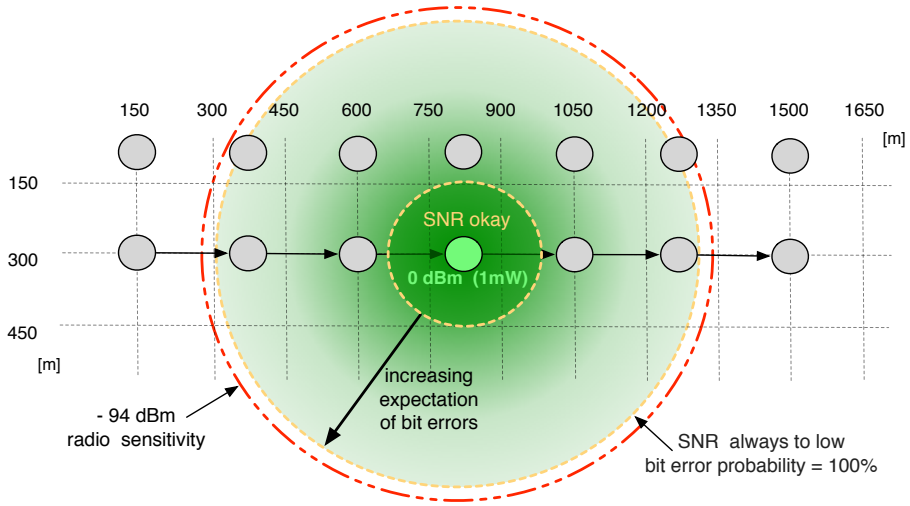


Figure 2.22: Bit error probability with the CC2420 radio module in OMNeT++.

To consider external interfaces, we added a mechanism simulating external interfaces. We configured our mechanism to simulate a probability of 15% for concurrently occurring external interferences. Therefore, in 15% of the time a randomly placed device is transmitting with 1 mW (0 dBm).

CC2420 Radio Module for OMNeT++

We implemented and configured our CC2420 module for OMNeT++ according to specifications found in the manual of the CC2420 radio module [91] and Castalia [7]. The radio module emulates the four communication pins SFD, FIFO, FIFOP and CCA, the TX and RX buffer and the internal state machine of the CC2420 to achieve a realistic behavior of the radio module.

Figure 2.23 shows the implemented internal state machine of the CC2420 radio module for our OMNeT++ module. A transition between the single states is triggered either by an internal timer, an event such as the detection of a frame start or by sending a command to the radio module. To start a transmission of a packet that is stored in the TX buffer, either the command STXON or STXONCCA has to be sent to the radio. Listening mode is represented by the state RX_SFD_SEARCH. In this state, the radio searches for a frame start. Before the

radio module can go into listening mode, the radio has to be calibrated for 192μs in state RX_CALIBRATE. When detecting a frame start in RX_SFD_SEARCH, then the state is changed to RX_FRAME during receiving the frame. Without enabled AUTOACK, the state switches back to RX_SFD_SEARCH after receiving the last byte of the frame. Otherwise, with enabled AUTOACK function, the state is switched to TX_ACK_CALIBRATE in order to initiate the transmission of the acknowledgment.

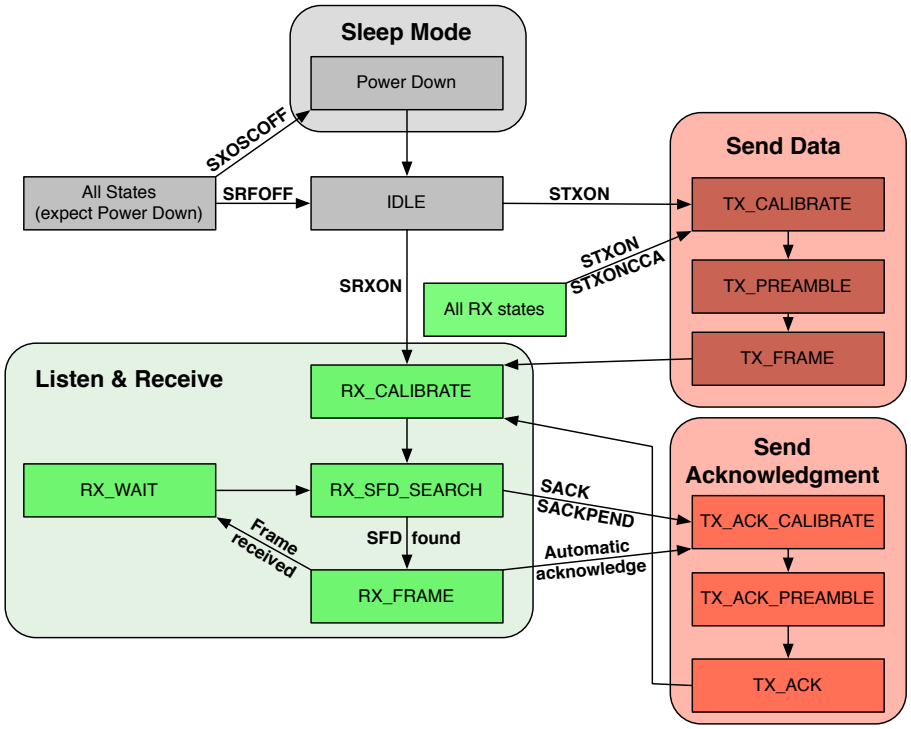


Figure 2.23: Radio control state machine.

2.4.2 WISEBED WSN Testbed Controlled by TARWIS

To verify our developed protocols, we used a real world WSN testbed. We, employed the WISEBED WSN testbed [79], which is controlled by the *Testbed Management Architecture for Wireless Sensor Networks* (TARWIS) software [38]. The target of the WISEBED WSN testbed is establishing and interconnecting several real world WSN testbeds within a federation of testbeds. Interconnecting the individual WISEBED WSN testbeds located at different research facilities enhances the scalability and enables establishing large WSN networks. For our evaluations, we used the WISEBED WSN testbed, which is located in the building of the Institute of Computer Science and Applied Mathematics at the University of Bern (IAM). During our evaluations, the interconnection of individual WSNs on differ-

ent research facilities was not completely finished and, therefore, not available for our tests. Additionally, the interconnection of individual WSNs is very challenging in terms of supporting adequate transmission delays and radio interferences between the individual WISEBED WSN testbeds .

The WISEBED WSN testbed at the IAM is managed by the TARWIS software. Two strong points of the web-based TARWIS user interface are:

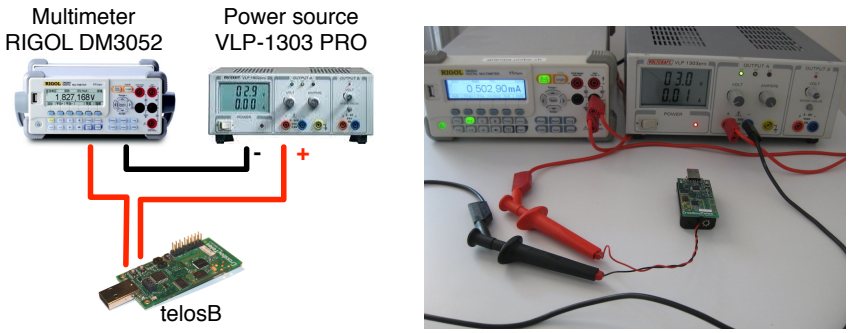
- **TARWIS experiment execution system:** The TARWIS experiment execution system enables the configuration and execution of experiments with the 40 available telosB sensor nodes in the IAM WISEBED WSN testbed. To configure an experiment, the duration, the required nodes and the software images for the individual nodes can be defined in advance. Additionally, the experiment configuration supports so-called commands. A command consists in a string, a time specification in seconds and a node address. The TARWIS experiment execution system sends the defined string at the specified time during the experiment execution over the serial connection to the addressed sensor node. These commands can be used to send instructions to a sensor node during the measurement. For example, instructions can trigger a transmission or to print out statistical data over the serial line.

- **TARWIS event recording system:** TARWIS uses the Wireless Sensor Network Markup Language (WiseML) [20] to store recorded data during the experiment execution. Every string, which a sensor node writes to the serial interface, including a timestamp, is written to a WiseML file.

2.4.3 RIGOL DM3052 Digital Multimeter

The digital multimeter DM3052 from RIGOL [72] enables an accurate recording of the electrical current of a single sensor node with a resolution up to 50'000 samples per second. The RIGOL multimeter is connected over an USB interface to a notebook running the measurement and control software. The measuring accuracy of the RIGOL multimeter is $\pm 2\ \mu A$. At the highest resolution of 50'000 samples per second, the measuring period of the RIGOL multimeter is limited to 42 seconds.

We use the laboratory power supply VOLTCRAFT VLP-1303 PRO [96] to power the sensor node connected to the RIGOL multimeter. The VLP-1303 PRO is a stabilized power supply, supporting a low residual ripple [37] to enable accurate measurements. Figure 2.24 shows the measurement setup to record the energy consumption of a sensor node with the RIGOL multimeter. We soldered copper wires to the sensor node to achieve a stable electric circuit. The copper wires of the sensor node are connected to the power supply, delivering electricity at voltage of at 3.00 V, and the RIGOL multimeter.



(a) Schematic assembly. **(b)** Real-world assembly.

Figure 2.24: Measurement setup to record the energy consumption of a sensor node.

2.5 Reliability Techniques

This section presents different reliability techniques to recover data loss occurring during data transmissions in WSNs. Subsection 2.5.1 introduces different reasons for data loss in WSNs. Subsection 2.5.2 provides an overview and classification of different reliability techniques to recover failed data transmission attempts. Subsection 2.5.3 and 2.5.4 describe two reliability techniques used in WSNs in more details. Subsection 2.5.5 describes a possible combination of these two reliability techniques. Subsection 2.5.6 defines two different metrics used to measure the reliability performance. Finally, Subsection 2.5.7 summarizes this section.

2.5.1 Reasons for Erroneous Data Forwarding in WSNs

This subsection introduces different reasons for data loss in WSNs. Figure 2.25 shows a typical WSN scenario, where two nodes are frequently sending data packets on a network path over multiple hops towards a sink. The intermediate nodes forward the received packets to the next node in the direction of the sink.

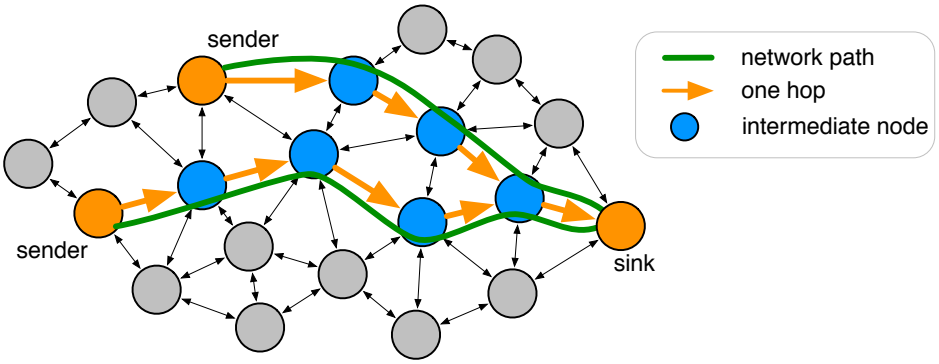


Figure 2.25: Forwarding packets in a WSN.

The authors of [65] identified different reasons for data loss in such a typical WSN scenario. In the following list we divide the discovered reasons for data loss into four groups:

1. **Inter-flow and intra-flow interferences:** These two types of interferences are generated by transmissions of neighboring nodes, which are part of the same WSN and use the same carrier wave frequency. Packets forwarded on neighboring network paths cause inter-flow interferences, while intra-flow interferences are caused by packets forwarded on the same network path. A good example for intra-flow interferences are acknowledgement messages sent by the sink back to the sender to acknowledge successful reception of a data packet. The authors of [83] evaluated the impact of concurrent transmissions with sensor nodes equipped with CC1000 radio modules. They reported that interferences caused by concurrent transmission show a significant impact on link quality and packet loss. They observed that if the SNR exceeds a critical threshold, the packet reception rate is over 90%.

   Moreover, inter-flow and intra-flow interferences are responsible for the hidden node problem [73]. Therefore, handling inter-flow and intra-flow interferences is an important task of WSN network stacks supporting multi hop communication. Mottola et al. [63] analyzed the hidden node problem in real road tunnels where some nodes are able to detect concurrent transmissions of other nodes. They reported that the hidden nodes create significant interferences and generated additional packet loss to other transmissions.

2. **External-interferences:** Any electrical device can emit electromagnetic radiations at different frequencies. Especially, devices using wireless communication systems that operate on similar carrier wave frequencies may cause significant interferences. Many works report that IEEE 802.11 networks can cause significant interferences in WSNs using IEEE 802.15.4 compliant radio modules [69, 80, 100]. For example, the authors of [53, 84] observed that many IEEE 802.11b nodes are not able to detect IEEE 802.15.4 transmissions. Srinivasan et al. [85] discovered that only IEEE 802.15.4 channel 26 is not affected by IEEE 802.11b transmissions.

   Moreover, the frequency of the electromagnetic radiation has not to be in the same range as the frequency of the carrier wave to cause interferences. Any electrically conductive component on a sensor node can receive electromagnetic radiation of specific frequencies which increases the noise in the radio receiver module.

3. **Radio wave propagation effects:** The sender itself generates interferences caused by radio wave propagation effects [81]. Figure 2.26 depicts different radio wave propagation effects such as reflections, diffraction and multi-path fading, which can cause bit errors in the received packet. Radio wave propagation effects are difficult to influence by a link layer protocol. The most

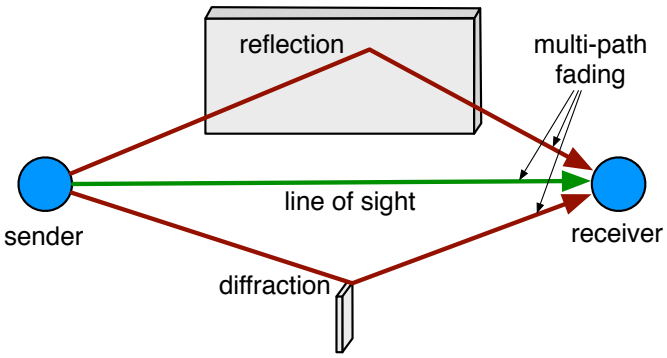


Figure 2.26: Radio wave propagation effects.

promising option is to adapt the transmission power of the radio module. The authors of [92] analyzed radio wave propagation effects in potato field. They report that the radio wave propagation was better at high humidity. This is maybe caused by changes in the reflection of the potato canopy.

4. **Packet buffer overflow:** The available packet buffer space is limited on the individual nodes. During periods with high traffic load or congestion, some of the forwarding nodes may receive more packets than they are able to forward to the next hop. If the difference between incoming and outgoing packets is too high, then the packet buffer overflows. In this case, the receiver has to drop packets, even if there is bit error in the received packet.

The probability of bit errors in received packets depends on the SNR during their reception. Moreover, the distribution of individual bit errors inside the received packets can differ according to the source of interference. For example, interferences generated by IEEE 802.11g networks cause significantly more bit errors in the first bytes of received IEEE 802.15.4 frames than in the rest of frame [53]. This happens due to the following reason. A node communicating by IEEE 802.11g requires a shorter time period to send a frame than a node that uses IEEE 802.15.4 If both nodes perform the channel check at the same time, then the transmissions of both nodes start at the same time. As a result, the received IEEE 802.11g frame and the link header of the received IEEE 802.15.4 frame header are corrupted.

The next subsection introduces different techniques for WSNs to recover failed packet forwarding attempts.

2.5.2 Overview Reliability Techniques

This subsection introduces and classifies different reliability techniques for WSNs to recover packet loss. The authors of [57] present a survey on existing reliability protocols for WSNs. All these reliability techniques require additional energy for recovery. This depicts a fundamental problem for energy preserving WSNs. On

one hand WSNs require a reliable communication to ensure the functionality of the WSN applications. But on the other hand, WSNs require an energy efficient communication to enhance the is lifetime. Therefore, there is a need for energy preserving reliability techniques.

The reliability performance required by WSN applications can either be on the level of **event reliability** or on the level of **packet reliability**. With event reliability, the corresponding WSN application requires only enough information to recognize an event. Packet reliability is required by WSN applications that require all packets containing sensed data or event information. Moreover, packet reliability is required by applications performing software or configuration updates on the sensor nodes.

In general, protocols offering reliability functionality either use **retransmission based** or **redundancy based** packet recovery techniques. Additionally, some hybrid protocols use a combination of both recovery techniques. Retransmission based recovery mechanisms make use of **Automatic Repeat Request (ARQ)** to detect and retransmit lost packets. Redundancy based recovery mechanisms apply **Forward Error Correction (FEC)** codes to increase the probability of a successful transmission. Hybrid protocols combine ARQ schemes with FEC codes to reduce the required retransmission attempts to preserve energy.

Figure 2.27 shows that the individual recovery mechanisms can be applied either on a **hop-to-hop** or an **end-to-end** level. Hop-to-hop reliability mechanisms are implemented by link layer protocols and executed by each node on the multihop network path between sender and sink. End-to-end reliability mechanisms are implemented by transport layer protocols and, therefore, are only applied by sender and sink. With end-to-end reliability, every lost packet has to be retransmitted by the sender node and again forwarded by all intermediate nodes. With hop-to-hop reliability, intermediate nodes can directly retransmit a detected lost packets if packet forwarding has failed. Therefore, hop-to-hop reliability mechanisms generate a lower delay and require less energy than end-to-end reliability mechanisms in general. Unfortunately, hop-to-hop reliability mechanisms are not able to guarantee full packet reliability between a sender and sink as shown in Figure in 2.27. For example, hop-to-hop reliability mechanisms are not able to recover packet loss

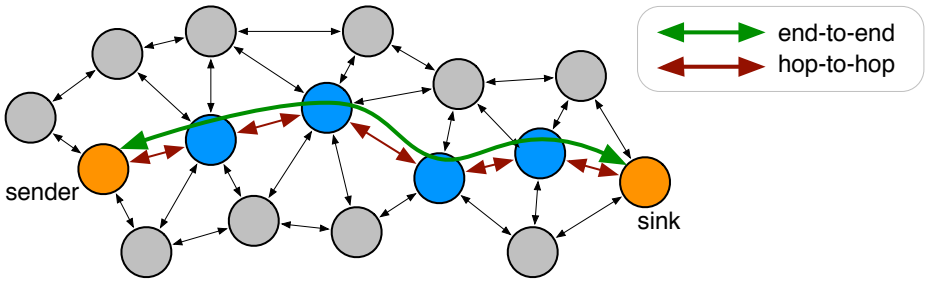


Figure 2.27: Reliability on hop-to-hop and end-to-end level.

caused by a packet buffer overflow. Therefore, a WSN network stack requires an end-to-end reliability protocol on the transport layer using a retransmission based recovery mechanisms to offer full packet reliability. Additionally, a hop-to-hop reliability protocol on the link layer can be used to reduce the end-to-end packet loss rate and, therefore, to preserve energy.

In this thesis, we apply retransmission based recovery techniques on the hop-to-hop and end-to-end level to ensure packet reliability in our WSN network stack. Furthermore, we evaluate the impact of redundancy based and hybrid packet recovery mechanisms to energy efficiency and packet loss. To sum up this subsection, Figure 2.28 shows a rough classification of the reliability techniques used in this thesis and the relation of the introduced terms and definitions.

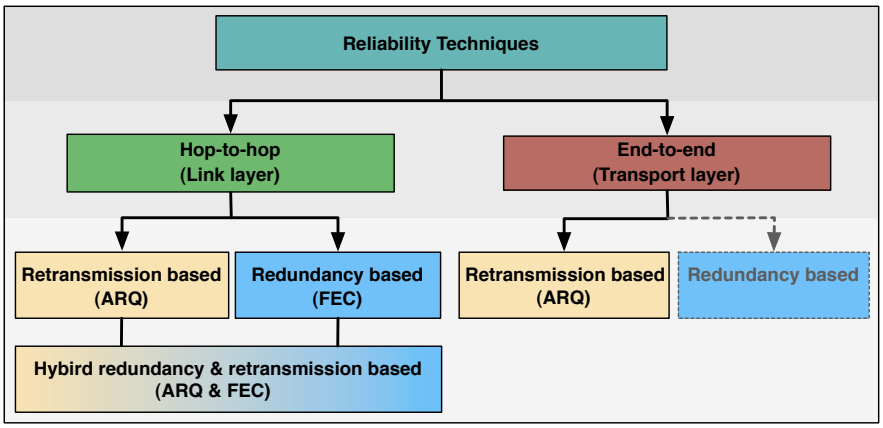


Figure 2.28: Reliability techniques implemented used in this thesis.

2.5.3 Automatic Repeat Request Mechanism

Automatic Repeat Request (ARQ) mechanisms are used in hop-to-hop and end-to-end retransmission based recovery mechanisms. ARQ mechanisms in WSNs make use of three different acknowledgement mechanisms to trigger the retransmission of a lost packet:

1. **Explicit acknowledgements:** Figure 2.29 depicts an example of an explicit acknowledgement for a forwarded data packet. With explicit acknowledgements, every successfully received data packet is directly acknowledged by the receiver with a short notification message. If a sender is not able to recognize the expected implicit notification message, then the packet is retransmitted. This mechanism offers the highest reliability guarantee. It is used by protocols providing packet reliability on hop-to-hop as well as on end-to-end level.

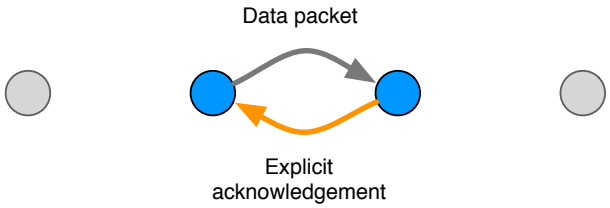


Figure 2.29: Explicit acknowledgement.

2. **Negative acknowledgements:** Negative acknowledgement mechanisms use sequence numbers to detect packet loss. Figure 2.30 depicts an example for a negative acknowledgement. A node detects the failed transmission of the packet with sequence number n after having received the subsequent packet with sequence number $n+1$. Now, this node sends a negative acknowledgement to the sender to indicate the loss of packet n. During periods with low packet loss, this mechanism requires a lower amount of individual notification messages than explicit acknowledgements. Negative acknowledgements are used by protocols providing packet reliability on a hop-to-hop as well as an end-to-end level.

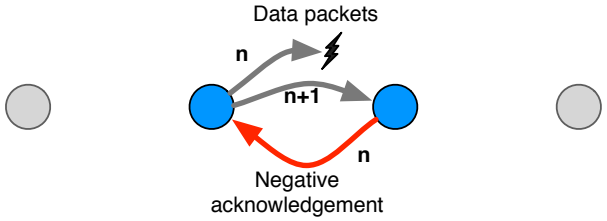


Figure 2.30: Negative acknowledgement.

3. **Implicit acknowledgements:** Figure 2.31 shows how implicit acknowledgements are realized. After transmitting the data packet, the sender overhears the channel to detect the forwarding of the same packet by the next node. This mechanism does not require any additional notification messages.

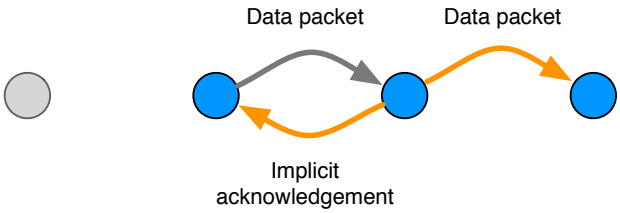


Figure 2.31: Implicit acknowledgement.

Implicit acknowledgements show some drawbacks in WSNs. For example, overhearing of the forwarded packet requires additional energy and overhearing does not work on last hop. This mechanism is only used by protocols providing hop-to-hop reliability.

The next subsection introduces redundancy based packet recovery mechanisms using FEC. In contrast to ARQ, FEC mechanisms do not retransmit lost packets. FEC mechanisms try to recover bit errors in packets which are successfully detected by the physical layer, but incorrectly received, using redundant data.

2.5.4 Forward Error Correction Codes

In this subsection we introduce redundancy based packet recovery mechanisms based on Forward Error Correction (FEC). FEC codes use Error Correction Codes (ECCs) [61, 68] to add redundant information to a packet. The redundant information enables a receiver to detect and correct bit errors in the received packet. The amount of bit errors that can be corrected depends on the ECC used by the FEC code. FEC codes show the following advantages and disadvantages concerning reliability and energy consumption:

- **Advantages:**
  - Some of the corruptly received packets can be restored.
  - The amount of required transmissions attempts can be reduced.
  - Lesser transmission attempts reduce intra-flow and inter-flow interferences.
- **Disadvantages:**
  - Additional time and energy for calculations of FEC codes are required.
  - The redundant data part has to be transmitted additionally.
  - Packets cannot be recovered if there are bit errors in the preamble or the frame start delimiter as well as if received packet has too many bit errors.

Figure 2.32 shows data, which has been encoded by an ECC code. The **data word** represents the original data with a length of m individual **data symbols** . The **parity code** is the redundant data with a length of r data symbols added by the ECC. Both parts together build the **code word** with length of $n = m + r$ data symbols, which is transmitted by the sender. To announce for example a FEC code using a Hamming ECC, we use the notation Hamming(n, m).
The remainder of this subsection is structured as follows. In a first part the two ECCs used for our FEC code evaluation are introduced, namely the linear Hamming ECC [33] and the cyclic Reed-Solomon ECC [70]. Afterwards, we introduce the two ECCs Hamming(12,8) and Reed-Solomon(255,225) as instances of Hamming Reed-Solomon with specific code word lengths and discuss them in detail. Finally, the differentiation between FEC codes and process gain is discussed.

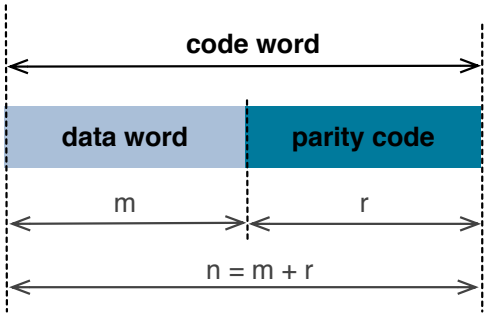


Figure 2.32: Data encoded with an ECC.

Hamming Code

Hamming codes [33] are linear ECCs. The data symbols used in a Hamming code are bits, which have a cardinality (S) of 1. The cardinality of a set of symbols is given by $2^{|S|}$:

$$\text{Bit: } 2^{|S|} = 2^{|1|} = 2 \text{ symbols.}$$
$$\text{Byte: } 2^{|S|} = 2^{|8|} = 256 \text{ symbols.}$$

With Hamming codes, the individual parity bits of the parity code are placed inside a code word at predefined positions. The individual bit positions are the elements 2^x. Therefore, every parity bit has exactly one 1 at its positions represented as binary number. Table 2.5 shows an example of Hamming(12,8) for the data word 00101011.

Position						**Type**	**Example**
Decimal	2^x	Binary					Code word
1	2^0	0	0	0	**1**	Parity bit	0
2	2^1	0	0	**1**	0	Parity bit	0
3		0	0	1	1	Data bit	0
4	2^2	0	**1**	0	0	Parity bit	0
5		0	1	0	1	Data bit	0
6		0	1	1	0	Data bit	1
7		0	1	1	1	Data bit	0
8	2^4	**1**	0	0	0	Parity bit	1
9		1	0	0	1	Data bit	1
10		1	0	1	0	Data bit	0
11		1	0	1	1	Data bit	1
12		1	1	0	0	Data bit	1

Table 2.5: Hamming(12,8) code word

To every parity bit, a set of individual code words bit are associated. The sum modulo 2 of the individual associated bits is the value of the parity bit. The individual associations are based on the parity bit and code word bit positions. To the parity bit at position 2^x, all code word bits with position y are associated where $2^x \cdot y = 2^x$. For example code word bit at position $y = 9 = 1001_2$ is associated to parity bit $2^4 = 1000_2$ (P_{1000}) due to $1000_2 \cdot 1001_2 = 1000_2$.

$P_{1000} = D_{1001} + D_{1010} + D_{1011} + D_{1100} = 1 + 0 + 1 + 1 = 1$
$P_{0100} = D_{0101} + D_{0110} + D_{0111} + D_{1100} = 0 + 1 + 0 + 1 = 0$
$P_{0010} = D_{0011} + D_{0110} + D_{0111} + D_{1010} + D_{1011} = 0 + 1 + 0 + 0 + 1 = 0$
$P_{0001} = D_{0011} + D_{0111} + D_{0111} + D_{1001} + D_{1011} = 0 + 0 + 0 + 1 + 1 = 0$

The final code word to be transmitted is 0000'0101'1011. The receiver is able to decode the code word and check if there is a bit error by summing up all bits that have the value "1" at the corresponding position in the binary number:
$S_{1000} = 1 + 1 + 0 + 1 + 1 = 0$
$S_{0100} = 0 + 0 + 1 + 0 + 1 = 0$
$S_{0010} = 0 + 0 + 1 + 0 + 0 + 1 = 0$
$S_{0001} = 0 + 0 + 0 + 0 + 1 + 1 = 0$

The decoded number is 0000. This number indicates that there is no bit error in the code word. If there is a single bit error, for example at the third bit in the received code word (10'0101'1011), then the decoding delivers a number different from 0000.

$S_{1000} = 1 + 1 + 0 + 1 + 1 = 0$
$S_{0100} = 0 + 0 + 1 + 0 + 1 = 0$
$S_{0010} = 0 + 1 + 1 + 0 + 0 + 1 = 1$
$S_{0001} = 0 + 1 + 0 + 0 + 1 + 1 = 1$

The result 0011 shows that the bit at position 0011 = 3 has flipped its value. Hamming(12,8) is able to repair one single bit error in a data word with a length of one byte.

Reed-Solomon Code

Reed-Solomon codes [70] are widely used in practice, e.g. in audio-CDs, mobile communication, digital video and audio broadcasting systems. A Reed-Solomon code is a subgroup of Bose-Chaudhuri-Hocquenghem (BCH) [68] code. A BCH code is a polynomial code over the Galois field $GF(q)$ [68] with a particularly selected generator polynom. To **encode** the data word, we use the generator polynomial ($g(x)$):

$$g(x) = (x - \alpha)(x - \alpha^2) \cdots (x - \alpha^t) = g_0 + g_1 x + \cdots + g_{t-1} x^{t-1} + x^t$$

The coefficients $g_0, g_1, \ldots, g_{t-1}$ are determined by the length of the data word and the length of the code word (n). The length of a Reed-Solomon code word is determined by the following equation:

$$n = 2^c - 1 = q - 1$$

Thus the code word length of Reed-Solomon codes is one symbol smaller than the cardinality (c) of the symbols. For the Reed-Solomon code we use byte symbols with a cardinality of 8. Therefore, the code word length for Reed-Solomon codes words with byte symbols is:

$$n = 2^8 - 1 = q - 1 = 256 - 1 = 255$$

To **recover** t symbols with a Reed-Solomon code, a parity code with length $2t$ symbols is required. We select a parity code length of 30 bytes to recover up to 15 bytes bit errors in a received code word. The according notation for this code used is Reed-Solomon(255, 225). Usually, data packets in WSN networks are significantly shorter than 225 bytes. Therefore, the remaining symbols in the data word are filled with 0 to calculate the parity code. The added 0 in the data word have not to be transmitted.

FEC Encoded Data Packets

Figure 2.33 shows two packets with 66 byte data payload, once encoded with Hamming(12,8) [33] and once with Reed-Solomon(255, 225) [70]. For Hamming(12,8), the 66 byte data payload has to be split into 66 individual data words of 8 bits resulting in 66 code words of 12 bits with a total length of 99 bytes. With Reed-Solomon(255, 225), the 66 byte data payload fits into one single code word. The resulting code word length is 96 bytes.

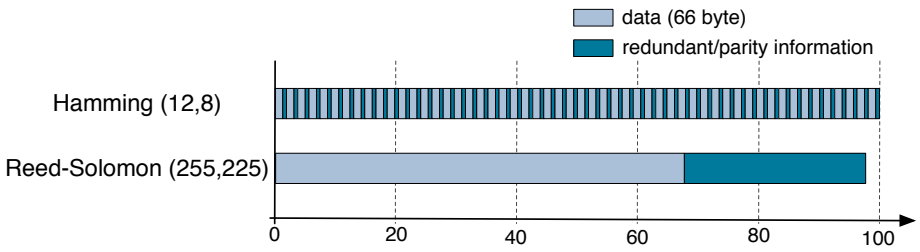


Figure 2.33: Two different FEC codes.

Forward Error Correction Versus Process Gain

Subsection 2.1.3 introduced the process gain provided by the spread spectrum technique DSSS. Both mechanisms, DSSS and FEC, add redundant information to the data stream to enable the receiver recovering bit errors. Nevertheless, both mechanisms differ in some specific characteristics:

- **Position in the network stack layer:** In general, FEC codes are applied to the link layer, while spread spectrum techniques are executed by the physical layer. Spread spectrum techniques add redundant information to all transmitted data symbols including the physical preamble and the frame start delimiter, while FEC codes are only applied to the data payload of the physical layer. Therefore, only spread spectrum techniques enable the detection of a frame with bit errors in the physical frame header. With FEC codes, packets having bit errors in the physical preamble or the frame start delimiter are not detect and, therefore, lost.

- **Execution time:** In general, FEC codes are implemented in software and handled by the microcontroller. Depending on the used FEC code and sensor node platform, the encoding and decoding operations require more time than the transmission of the frame. Spread spectrum techniques are directly handled by the radio module hardware, without generating any additional time delay.

  The required processing time is less important for FEC code evaluations if no energy preserving radio sleep cycles are used [41]. For energy preserving link layer protocols using radio sleep cycles, the execution time for FEC encoding and decoding has a major impact on their energy consumption, i.e. negatively affect their energy efficiency. Section 4.1.5 explains the individual problems in more detail. In the near future, maybe software defined radio modules with Field Programmable Gate Array (FPGA) [47] placed between the radio module and the microcontroller can reduce the required execution time and, therefore, improve the energy efficiency of FEC codes.

- **Application:** FEC codes have to be implemented within a WSN network stack protocol. Moreover, FEC codes require considerable programming skills to realize an efficient implementation. Spread spectrum techniques do neither require any programming skills nor have they to be added to a network stack protocol. They are implemented directly by the radio module hardware that handles the carrier wave modulation.

A detailed performance analysis of both mechanisms is presented in Section 5.4.

2.5.5 ARQ combined with FEC Mechanisms

This subsection briefly introduces a hybrid packet recovery mechanism combining ARQ with FEC mechanisms. ARQ mechanisms are able to detected and retransmit any failed packet transmission attempt, while FEC mechanisms are able to recover bit errors in received packets. If, for example, a received packet contains a single bit error, then the entire packet has to be retransmitted with ARQ. With FEC, a packet with a single bit error can be recovered without any retransmission.

The expectation of a hybrid reliability protocol using ARQ with FEC is that FEC reduces the required numbers of retransmission attempts, while ARQ still

guarantees full packet reliability. The saved retransmission attempts may preserve more energy than the additional overhead of the FEC itself requires. Moreover, the reduced retransmission attempts also reduce the resulting internal interfaces, which can reduce the bit error rate in other transmissions.

The next subsection introduces different reliability metrics used to compare the reliability performance of different protocols.

2.5.6 Reliability Metrics

To evaluate and compare the reliability performance of different protocols, we employ the following two reliability metrics:

- **Expected Transmission Count (ETX):** The ETX metric measures the quality of a direct link between two nodes. The ETX value is a positive integer with a minimal value of 1. The number represents the estimated number of transmission attempts to successfully forward a packet to the next hop. This metric is used in many wireless mesh networking algorithms, i.e. by the routing protocols shown in [19].

- **End-to-end packet loss rate:** The end-to-end packet loss rate specifies how many packets that a sender has generated are not received by the sink. If the end-to-end reliability mechanism is able to recover all packets, the end-to-end packet loss rate is 0%.

Moreover, we measure how many end-to-end retransmissions were executed by the end-to-end reliability protocol to determine how many packets are dropped by our hop-to-hop reliability protocol.

2.5.7 Summary Reliability Techniques

In this section we introduced and categorized different reliability techniques. ARQ schemes are retransmission based recovery mechanisms, while FEC codes are redundancy based recovery mechanisms. ARQ schemes employ different acknowledgment techniques to detect and retransmit failed packet transmission attempts. Redundancy based FEC codes increase the probability of a successful transmission, but they do not retransmit a lost packet. Combinations of FEC and ARQ may reduce the required transmission attempts to support full reliability.

Moreover, we showed that signal spreading techniques able recover bit errors on the physical layer. In this thesis, we evaluate ARQ with implicit acknowledgments and FEC codes in hop-to-hop reliability protocols, implicit and negative acknowledgments in end-to-end reliability protocols and the signal spreading techniques DSSS.

2.6 WSN Network Stack Protocols and Mechanisms

This section introduces different communication protocols for WSN network stacks. Moreover, we introduce common mechanisms used by WSN communication protocols to fulfill tasks, such as energy preserving, packet loss recovery, traffic flow handling. All protocols in a WSN network stack share the fact that they are designed to operate with the limited processing power and memory constraints of sensor nodes. Subsection 2.6.1 introduces different link layer protocols for WSNs. These protocols control the radio sleep cycles to preserve energy and enable reliable hop-to-hop data transmissions. Subsection 2.6.2 describes network and transport protocols for WSNs. Common tasks for these protocols are addressing and routing of packets in multi-hop WSNs as well as ensuring the end-to-end reliability of the transmission. Subsection 2.6.3 presents data aggregation mechanisms for WSNs. These mechanisms are used by different layers to reduce the total amount packets that have to be forwarded in a WSN. Finally, Subsection 2.6.4 discusses back pressure mechanisms for WSNs used to control the traffic flow.

2.6.1 Link Layer Protocol

This subsection introduces different link layer protocols for WSNs. The two major tasks of WSN link layer protocols are preserving energy and ensuring hop-to-hop reliability. Therefore, the link layer is usually divided into two sub-layers to separate these mechanisms and reduce the complexity. The lower layer handles the radio sleep cycles to preserve energy. The upper layer ensures hop-to-hop reliability. Moreover, WSN link layer protocols have to handle interferences and congestion occurring during periods with high traffic.

The remainder of this subsection introduces a classification of energy preserving link layer protocols for WSNs. Furthermore, the WSN link layer protocols shown in Table 2.6 are described in more detail. All these protocols are implemented in real world WSN network stacks.

Protocol	Main function	Available for	Supported sensor nodes
XMAC	Energy saving	Contiki	MicaZ, telosB
ContikiMac	Energy saving	Contiki	MicaZ, telosB
MaxMAC	Energy saving	Scatterweb	MSB430
NullRDC	(RDC reference)	Contiki	ESB, BTnode, MSB430, MicaZ, telosB
Contiki CSMA	Hop-to-hop reliability	Contiki	ESB, BTnode, MSB430, MicaZ, telosB

Table 2.6: WSN link layer protocol overview

Energy Preserving WSN Link Layer Protocols

WSN link layer protocols preserve energy by duty cycling the radio module. A duty cycling mechanism periodically turns off the radio module to reduce energy consumption of the radio module. Depending on the radio module, the energy consumption during sleeping mode is $10^3 - 10^6$ times lower than during listening mode. Between the individual sleeping periods, the radio module wakes up to listen to the channel for detection and reception of packets. Figure 2.34 shows a schematical example of a protocol duty cycling the radio module to define the following terms:

- **Wake-up period:** During the wake-up period, the radio module listens to the channel for incoming packets. The duration of the wake-up period depends on the result of the channel check. If no transmission can be detected, then the radio module goes back to sleep.
- **Sleeping period:** Between the individual wake-up periods, the radio module is turned off to preserve energy within the sleeping period. The longer the applied sleeping period, the less energy is required during time periods with low traffic load.
- **Duty cycle period:** The duty cycle period is the time period required for a wake-up period and the subsequent sleeping period.
- **Duty cycle:** The term duty cycle is used as the ratio of the wake-up period and the duty cycle period.

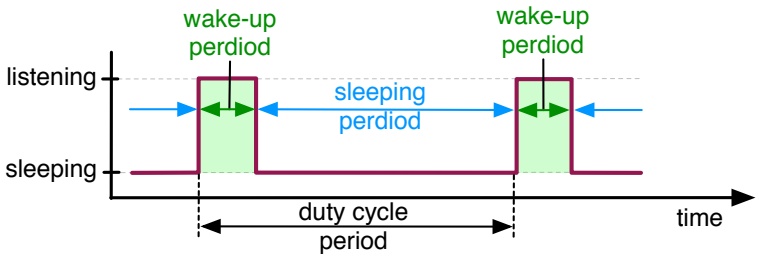


Figure 2.34: Example of a protocol duty cycling the radio module.

A major challenge for radio duty cycling protocols is to determine the point in time to start the next wake-up period to receive a new packet. In the best case the radio module wakes up as soon as another node tries to send a packet to it.

Figure 2.35 shows an overview of existing radio duty cycling mechanisms handling the timing problem of the wake-up period in different ways. Radio duty cycling mechanisms can be divided into **synchronous** and **asynchronous** radio duty cycle protocols. Synchronous radio duty cycle protocols synchronize the

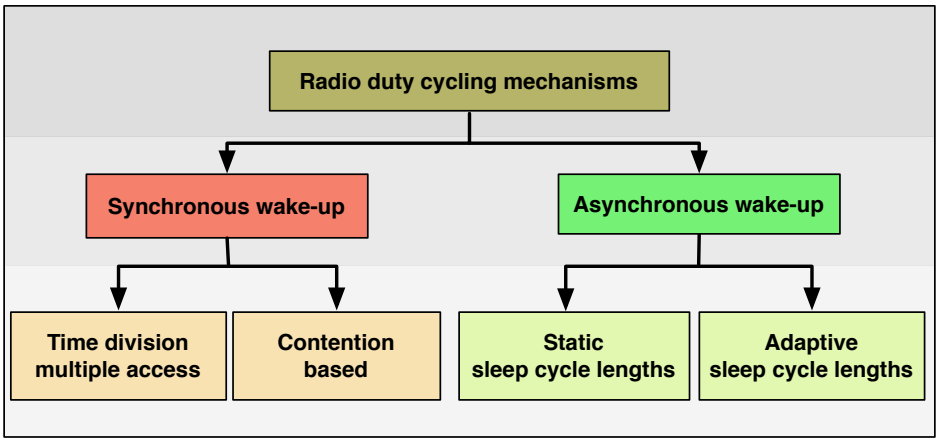


Figure 2.35: Classification of energy preserving link layer protocols.

wake-up periods among neighboring nodes. With these protocols all neighboring nodes are active during the same time period. Synchronous radio duty cycle protocols can be subdivided into **contention based** and **time division multiple access (TDMA) based** protocols. Contention based protocols, such as Sensor-MAC (S-MAC) [101], use carrier sense multiple access (CSMA) mechanisms during the synchronized wake-up period to enable the forwarding of packets between neighbor nodes. With TDMA based protocols such as the Lightweight Medium Access (LMAC) protocol [94], the individual nodes can only transmit packets during their exclusively allocated wake-up periods. In contrast to synchronous, asynchronous radio duty cycle protocols do not synchronize the wake-up periods among neighboring nodes. Every sensor node can perform its wake-up period independently from the schedule of the neighbor nodes. Asynchronous radio duty cycle protocols can be subdivided into protocols with **static** and **adaptive** sleep cycle lengths. With adaptive sleep cycle lengths, the duration of the sleeping period can be adapted, for example, to the current traffic load. A sender using an asynchronous radio duty cycle protocol requires a mechanism to recognize the wake-up periods of its receiver to forward a packet. The following list introduces two common mechanisms used by asynchronous radio duty cycle protocols to detect the wake-up periods:

- **Low Power Probing (LPP):** This mechanism employs the receiver nodes to announce the wake-up periods to a sender (see Figure 2.36). Every sensor node sends a short Hello message called *probe* at the beginning of a wake-up period. A sender that has a packet to forward scans the radio channel for such a probe from the desired receiver. After having received the probe, the sender immediately forwards the data packet to the receiver. An example for a LPP protocol is Koala [64]. This protocol is designed for long term environmental monitoring with low traffic load.

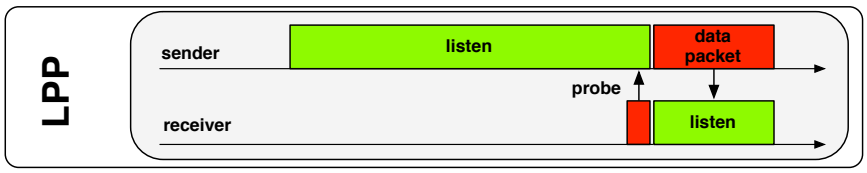


Figure 2.36: Low Power Probing (LPP) mechanism.

- **Low Power Listening (LPL):** This mechanism employs the sender nodes to announce the forwarding of a packet to the receiver node (see Figure 2.37). Therefore, a sender transmits a long physical preamble. If a neighbor node detects such a preamble during the wake-up period, then it waits until the data part is transmitted. Usually, the preamble includes the target address of the data part. This enables non-involved neighbor nodes to quickly return back to sleep. Wireless Sensor MAC (WiseMAC) [29] uses LPL with a long preamble including the receiver address of the data part. WiseMAC is designed to send data from a management node to the sensor nodes.

Figure 2.37: Low Power Listening (LPL) mechanism.

The authors of A Real-Time and Energy Efficient MAC (AREA-MAC) [50, 51] propose an asynchronous and adaptive LPL protocol using several short preambles including the destination address instead of one single long preamble. Replacing the long physical preamble with short preamble packets, so called beacon strobes, additionally enables the support of packet oriented radio modules. Figure 2.38 shows an LPL approach, with short preamble packets modified for packet oriented radio modules. Single packets, so called beacon strobes, replace the long physical preamble. A receiver that gets a beacon strobe addressed to itself sends back an

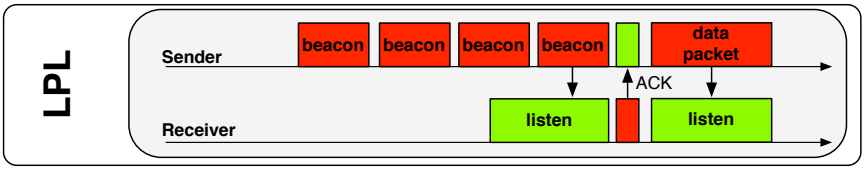


Figure 2.38: LPL mechanism for packet oriented radios.

acknowledgment to announce the wake-up period to the sender node. Then, the sender transmits a data packet to the receiver.

XMAC

XMAC [13] is an LPL based link layer protocol using beacon strobes instead of a long preamble. Therefore, XMAC is capable of supporting packet-oriented radio modules such as the CC2420 radio module. XMAC adapts the duty cycle period according to current traffic load. Current traffic load is determined by counting the wake-up periods in which a packet has been received. If more than a certain percentage of wake-up periods having received packets, then the duty cycle period is decreased to offer more bandwidth. If the percentage of wake-up periods having received packets is too low, then the duty cycle period is increased to save energy. The used percentages of wake-up periods triggering a change in the duty cycle period is not defined by the authors of XMAC. Such traffic monitoring mechanisms work quite well in simple network topologies with low internal interferences. Network topologies including multiple connections and simultaneously forwarded packets cannot be handled efficiently by this simplistic mechanism. Resulting internal interferences may trigger congestion. Congestion results in lower generated traffic. This causes a lower number of counted packets and, therefore, traffic monitoring decreases the duty cycle. Lower duty cycles in case of congestion and high traffic load further increases packet loss due to buffer overflow. Congestion rather requires high duty cycle to increase network bandwidth for high traffic load. XMAC has been implemented on real sensor nodes using the Contiki operating system [87]. The Contiki implementation of XMAC does not support adaptive duty cycles due to the simplistic traffic monitoring mechanism, which is not able to handle internal interferences and congestion in real world WSN.

Figure 2.39 explains how XMAC forwards a pending frame. The numbering of the following list refers to the numbers depicted in Figure 2.39:

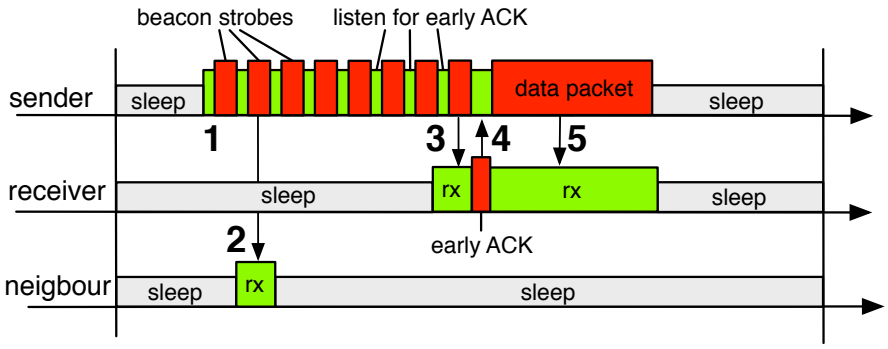


Figure 2.39: X-MAC protocol design.

1. The sender turns on the radio to forward a pending packet. After having initially checked the channel, the sender starts repeatedly transmitting beacon strobes including the address of the receiver node.
2. A neighbor node wakes up to check the channel and receives the next transmitted beacon strobe. Since the packet has not been addressed to the node, the radio module is still turned off.
3. The addressed receiver node wakes up and receives a beacon strobe.
4. The receiver instantly returns an *early acknowledgment* to announce wake-up period to the sender.
5. The sender receives the early acknowledgment and starts transmitting the data packet.

ContikiMac

ContikiMac [24, 26] is an improvement of the Contiki XMAC implementation. ContikiMac is an LPL based link layer protocol using beacon strobes and applying static duty cycle periods.

Figure 2.40 shows the protocol design of ContikiMac. The numbering of the following list refers to the numbers depicted in Figure 2.40:

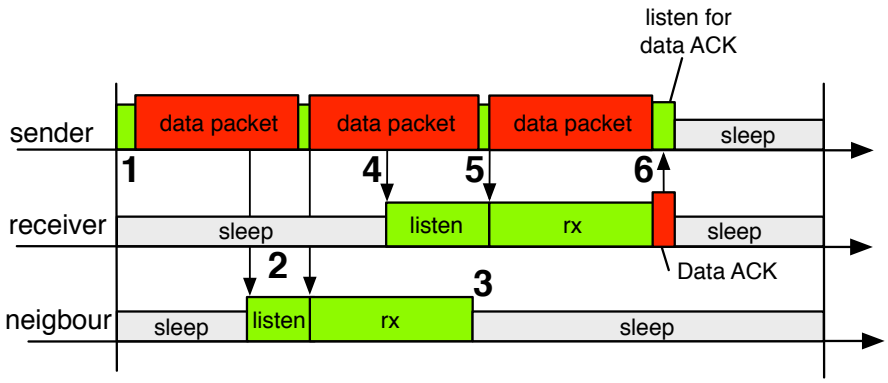


Figure 2.40: ContikiMac protocol design.

1. A sender with a pending packet repeatedly transmits IEEE 802.15.4 conform beacon strobes including the data payload.
2. A neighbor node wakes up and detects an ongoing transmission.
3. Since the target address of the next beacon strobe does not belong to this neighbor node, no acknowledgment has to be sent. The radio module is turned off until the next wake-up period.

4. The channel check of a receiver node detects an ongoing transmission. Therefore, the radio module remains in listening mode to receive the next beacon strobe.

5. The radio module detects and receives the next beacon strobe.

6. The beacon strobe has been transmitted completely and is acknowledged by the receiver node. After having processed the beacon frame, the receiver node turns off the radio module. The transmitter goes back to sleep after the data acknowledgment has been received.

In contrast to XMAC, ContikiMac is able to make use of the energy efficient link layer functions provided by the CC2420 radio module, such as handling acknowledgments or calculating checksums directly by the radio module. Moreover, ContikiMac introduces the so-called *transmission phase-lock* mechanism to reduce the total amount of beacon strobes required to forward a packet. This mechanism tries to learn the wake-up period phases of the receiver nodes. Once the sender learned these wake-up periods, the beacon strobe transmissions are delayed until the receiver node switches to the wake-up period. Evaluation in [26] showed that the wake-up period of ContikiMAC requires 11 times less energy than the wake-up period of XMAC.

ContikiMac does not support adaptive duty cycles, i.e. does not adapt the duty cycle duration to the current traffic load. The duration of the static duty cycle periods has to be defined during the network setup. By default ContikiMac applies a duty cycle period of 125ms.

MaxMAC

MaxMAC [39, 66] is an LPL based link layer protocol with adaptive duty cycles using long preambles. Figure 2.41 shows protocol design of MaxMAC. The numbering of the following list refers to the numbers depicted in this figure:

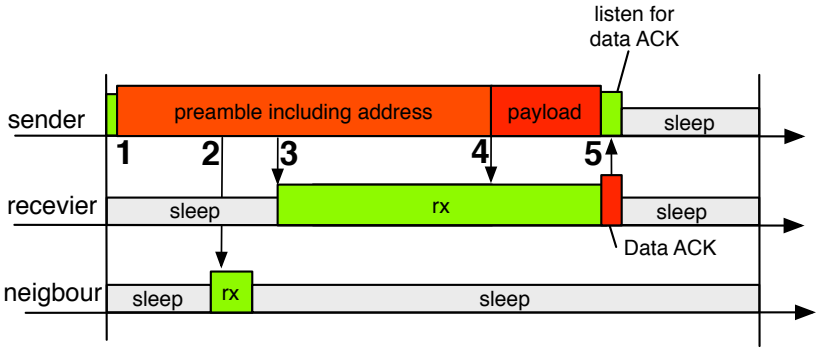


Figure 2.41: MaxMAC.

1. A sender with a pending packet transmits a long preamble including the address of the receiver. The preamble is transmitted for the length of one duty cycle.

2. A neighbor node wakes up and receives the preamble including the receiver address. Since the target address belongs to another node, the radio is turned off.

3. A receiver node wakes up and receives the preamble including the receiver address.

4. After having the length of one duty cycle, the sender assumes that the receiver is listening to the channel. Therefore, the frame start and payload can now be transmitted.

5. After having received the payload, a data acknowledgment is sent to the sender. Sender and receiver turn off their radio modules.

MaxMAC uses adaptive duty cycles to adapt the duty cycle periods to the current traffic load. Moreover, MaxMAC tries to learn the wake-up periods of the receiver nodes to reduce the length of the preamble. Once a sender learned these wake-up cycles, it delays the transmission of the preamble until the expected wake-up of that receiver node. This mechanism reduces the energy required by the sender to forward a packet.

MaxMAC determines the current traffic load, like XMAC, by counting the wake-up periods in which a packet has been received. As described before, this mechanism works quite well in simple network topologies with low internal interferences, but network topologies including multiple connections and simultaneously forwarded packets cannot be handled by MaxMAC.

MaxMAC does not work with packet oriented radio modules. The long preambles used by the LPL based MaxMAC requires a bit/byte-oriented radio module. Therefore, MaxMAC has been implemented in Scatterweb for MSB430 sensor nodes equipped with a CC1020 radio module.

NullRDC

The Contiki NullRDC protocol does not apply any sleep periods to the radio module. Therefore, NullRDC does not preserve energy. NullRDC is used as reference protocol for other *RDC* protocols.

Figure 2.42 shows the protocol design of NullRDC. The numbering of the following list refers to the numbers depicted in Figure 2.42:

1. All nodes are in listening mode.

2. A sender transmits a pending packet to the receiver. The receiver as well as all neighbor nodes receive this packet.

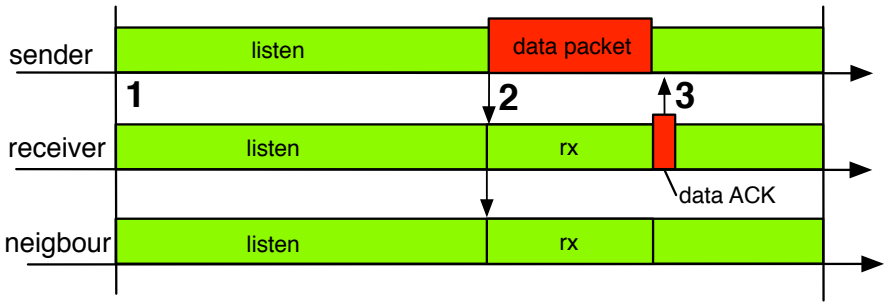


Figure 2.42: NullRDC.

3. The designated receiver sends back an acknowledgment to the sender. All other neighbor nodes drop the packet.

Contiki CSMA

The Contiki CSMA protocol is currently the only hop-to-hop reliability protocol provided by Contiki. It is located on the link layer in the *MAC* sub-layer, which is on top of the *RDC* sub-layer. If the *RDC* protocol, e.g., ContikiMAC, detects a packet loss, then the Contiki CSMA protocol may retransmit the packet at a later point in time.

Figure 2.43 shows the Contiki CSMA protocol within the Contiki network stack. The numbering of the following list refers to the numbers depicted in Figure 2.43.

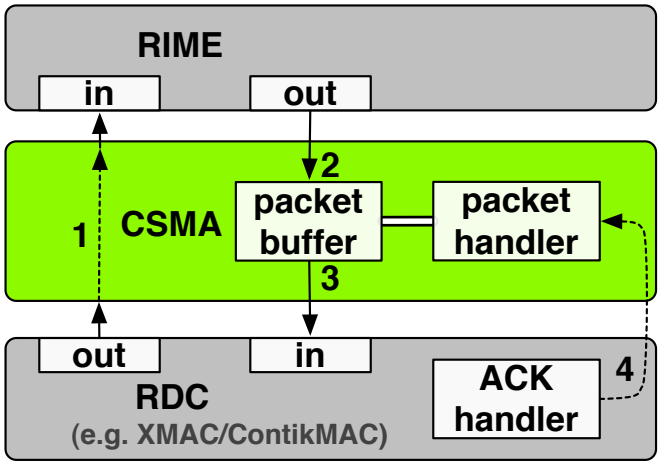


Figure 2.43: Contiki hop-to-hop reliability protocol CSMA.

1. Packets received by the *RDC* protocol are forwarded to the upper layer (e.g.,

RIME) without any interaction of the CSMA protocol.

2. Packets received from the upper layer are cached in the packet buffer. The packet handler controls the queuing of the packets in the packet buffer.

3. The packet handler forwards the next packet of the packet buffer queue to the *RDC* protocol.

4. After the *RDC* protocol has finished the transmission attempt of the packet, it sends a *transmission report* to the CSMA packet handler. This *transmission report* informs the packet handler if the transmission attempt was successful or if it failed. If the transmission was successful, then the packet is removed from the packet buffer. Otherwise, the packet is retransmitted with a certain delay. The length of the retransmission delay depends on the number of retransmission attempts.

2.6.2 Network and Transport Protocols

This subsection introduces existing network and transport protocols for WSNs. The network layer enables forwarding of data packets from a sender in one network to a receiver in another network. The transport layer provides end-to-end connections including reliability and flow control mechanisms. The remaining of this subsection introduces the TCP/IP protocol suite and TCP improvements for WSNs. Moreover, we introduce the Reliable Multi-Segment Transport (RMST) protocol supporting end-to-end reliability functionality based on negative acknowledgments.

TCP/IP

TCP/IP is the de facto standard protocol suite for wired communication. By running TCP/IP in the WSN, it is possible to directly connect the WSN to a wired network infrastructure, without proxies or middle-boxes [27]. While UDP can be used to transmit sensor data to a sink, TCP can be used for administrative tasks such as sensor node configuration and updating program code [4, 98].

Due to limited resources of the sensor nodes, high packet loss, and inefficiency in memory and energy consumption of TCP [56], it is rather difficult to implement TCP/IP on sensor nodes. μIP [21] is an existing TCP/IP implementation for sensor nodes equiped with 8-bit microcontrollers. Usually, μIP is using a very small *TCP Receive Window Size* to trigger an acknowledgement for every sent segment. This prevents the WSN from too many collisions and reduces the memory overhead. A big drawback of a small *TCP Receive Window Size* is the excessive additional signaling traffic. It requires additional energy and network bandwidth. Furthermore, handling traffic in both directions of a connection is a significantly higher challenge for the link layer protocols. This leads to many end-to-end retransmissions. These extra packets reduce throughput and increase Round-Trip-Time (RTT). The

extra energy for retransmissions also reduces the lifetime of the individual sensor nodes. Optimizations, e.g. distributed caching of TCP data packets, local retransmissions, and regeneration of TCP acknowledgment packets can reduce these problems [12, 22].

TCP Support for Sensor Networks

TCP Support for Sensor Networks (TSS) [12] is located between TCP and IP. It controls the packet forwarding on every intermediate node part of a TCP connection. TSS tries to improve the performance of TCP in WSNs with the following three mechanisms:

- Caching and local retransmission of TCP data packets (basic mechanism).
- Improved the TCP acknowledgment mechanism.
- Flow and congestion control.

Figure 2.44 depicts the basic idea of TSS. TSS caches a TCP-DATA packet forwarded in the WSN until the packet has been acknowledged. The numbering of the following list refers to the numbers depicted in Figure 2.44:

1. Node 4 caches the TCP-DATA packet n and forwards it.
2. The receiver acknowledges TCP-DATA n by transmitting TCP-ACK n+1.
3. The intermediate node receives TCP-ACK n+1 and deletes TCP-DATA n.

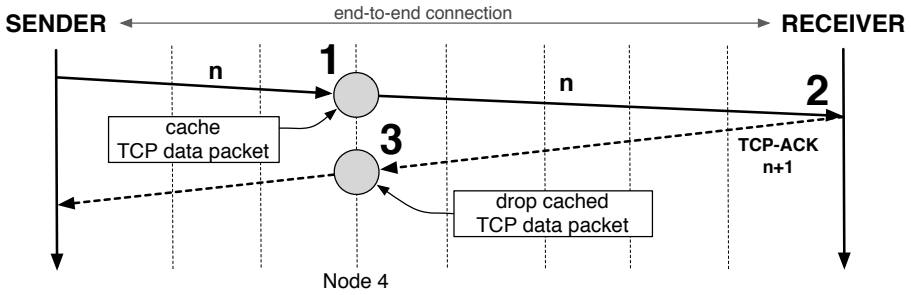


Figure 2.44: Basic idea of TSS: Caching of TCP data packets.

Figure 2.45 shows the case of a packet loss. The numbering of the following list refers to the numbers depicted in Figure 2.45:

1. Node 4 caches the TCP-DATA packet n and forwards it.
2. Due to the missing TCP-ACK, the cached TCP-DATA packet n is retransmitted after $1.5 * RTT$.

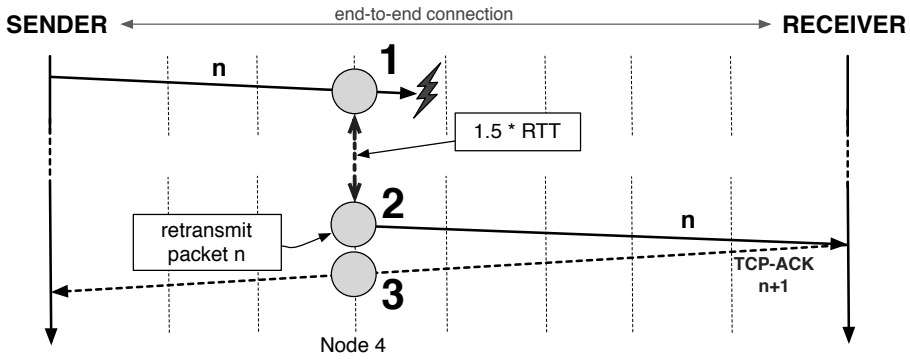


Figure 2.45: Basic idea of TSS: Retransmission of lost TCP data packets.

3. After receiving TCP-ACK n+1 it deletes TCP-DATA n from the buffer.

Too long retransmission timeouts cause retransmissions by the sender. Simulations in [12] show that a retransmission timeout of 1.5 * RTT is adequate to also retransmit multiple losses of TCP-DATA packets without triggering end-to-end retransmission. The RTT is measured between the node and the destination.

Because of the limited memory, only a few packets can be cached and an efficient caching strategy is required. Using cross layer support of the link layer protocol, TSS can discover a successful forwarding of the TCP-DATA packet n to the next hop. This enables a node to drop a cached TCP-DATA packet as soon as it knows that the successor node has successfully received the packet. Figure 2.46 shows an example of TSS using cross layer support of a link layer protocol with implicit acknowledgments:

1. Node 5 receives, caches, and forwards TCP-DATA n and n+1. The link layer protocol informs the TSS protocol that forwarding has failed.

2. Now node 5 tries to retransmit TCP-DATA n. After confirmation, it deletes TCP-DATA n from the buffer and transmits TCP-DATA n+1.

Experiments in [12] show that the loss of TCP-ACKs may have an impact on the amount of TCP-DATA packet transmissions. Figure 2.47 shows that the following two mechanisms reduce the consequences of lost TCP-ACKs:

1. **Local TCP acknowledgment regeneration:** The local TCP acknowledgment regeneration is used to drop duplicated TCP data packets, which are already acknowledged by the receiver. Due to the history list, an intermediate node can discover an already acknowledged packet. The duplicated packet is dropped and a TCP-ACK with the highest acknowledgment number seen so far is regenerated and transmitted.

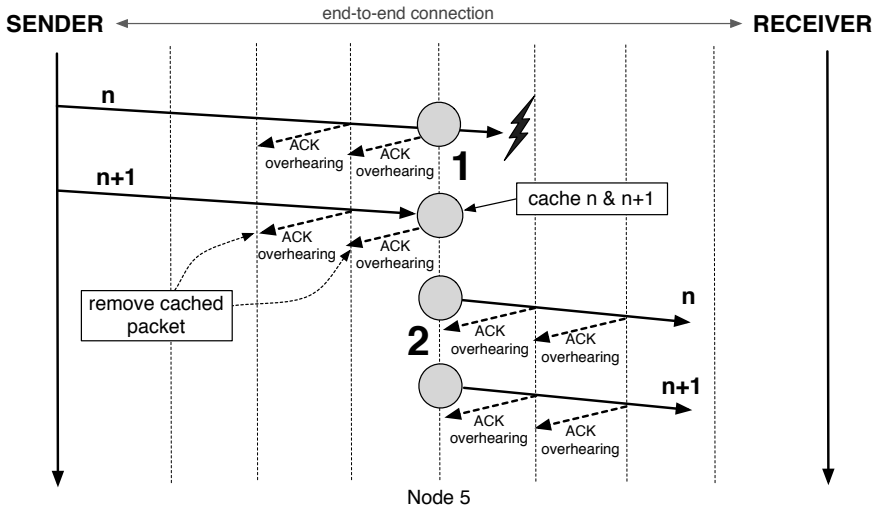


Figure 2.46: Cross layer support of the link layer protocol.

2. **Aggressive TCP acknowledgment recovery:** The aggressive TCP acknowledgment recovery becomes active when a sensor node cannot ensure by link layer acknowledgment that the TCP-ACK has been successfully transmitted to the next hop. Using link layer acknowledgments, the retransmission can be enforced directly.

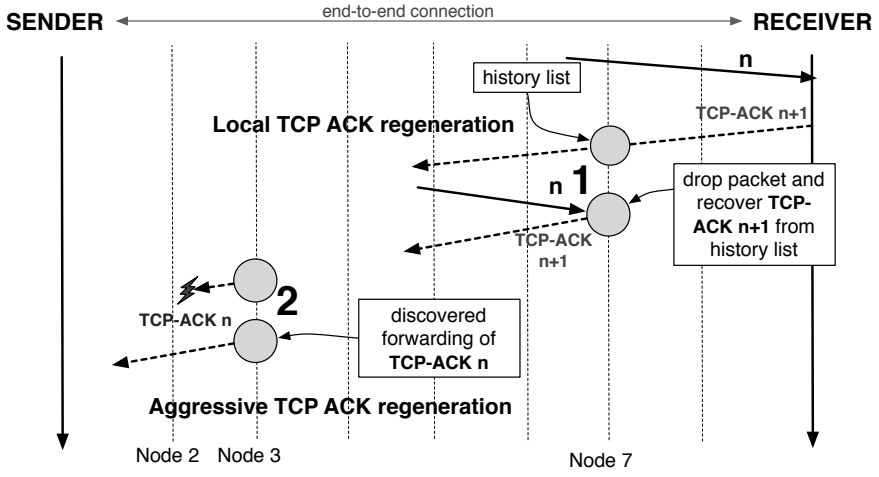


Figure 2.47: Two mechanisms to reduce the consequences of lost TCP acknowledgments.

The authors of [42] implemented the caching and retransmission mechanisms of TSS for μIP in Contiki. The evaluations are performed with telosB nodes in the WISEBED WSN testbed at the University of Bern. The evaluated network topol-

ogy is a single line of nodes with 2, 3, 4, 5 or 6 hops. Every experiment was executed for 10 minutes. During this time period, as many TCP segments as possible are forwarded to measure the throughput. On the link layer NullRDC, ContikiMAC and XMAC are used. They report that μIP with TSS and NullRDC was able to increase the throughput about 84% and μIP with TSS and XMAC able able to increases the throughput about 18% compared to unmodified μIP. When using ContikiMAC on the link layer, the throughput of TSS with μIP was up to 68% lower than with unmodified μIP.

Reliable Multi-Segment Transport Protocol

The Reliable Multi-Segment Transport (RMST) [86] protocol is a reliable transport layer protocol for Directed Diffusion [15, 36]. RMST uses selective negative acknowledgments to guarantee data delivery from different sensor nodes to a single sink node. Moreover, RMST provides a caching and repair mechanisms within the intermediate nodes.

Figure 2.48 shows the RMST frame format. RMST adds an incremented sequence number (FragNo) to each frame received from the application layer. Large application data frames are split into several smaller fragments before adding the incrementing sequence number. To these fragments, RMST adds the total number

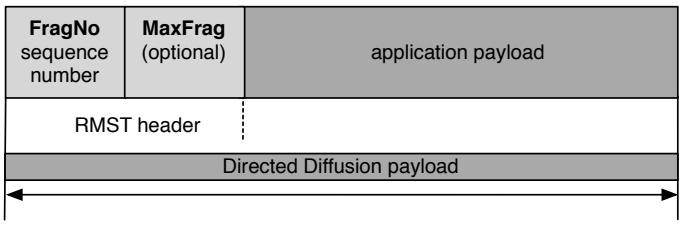


Figure 2.48: RMST frame header.

of fragments (MaxFrag) to enable reassembly of the application data frame at the sink.
RMST supports two different modes for recovering lost frames. The first mode is called **non-caching mode**. In this mode, only the sink is sending selective negative acknowledgment to request missing fragments according to their sequence numbers. The second mode is called **caching mode**. In this mode, each node in the network that caches frames sends selective negative acknowledgments if it detects missing packets. The detection of packet loss is timer driven. A watchdog periodically scans for missing sequence numbers within the packet history. If one or more sequence numbers are missing for a too long time period, a negative acknowledgment is sent to request the missing frames.

Figure 2.49 depicts a small example with three sensor nodes using RMST on top of Directed Diffusion. Every nodes that receives a negative acknowledgment checks its local cache for the missing packet. If the packet cannot be found in the

local cache, then the negative acknowledgment is forwarded on the reinforced path towards the source. Otherwise, the missed frame is retransmitted towards the sink.

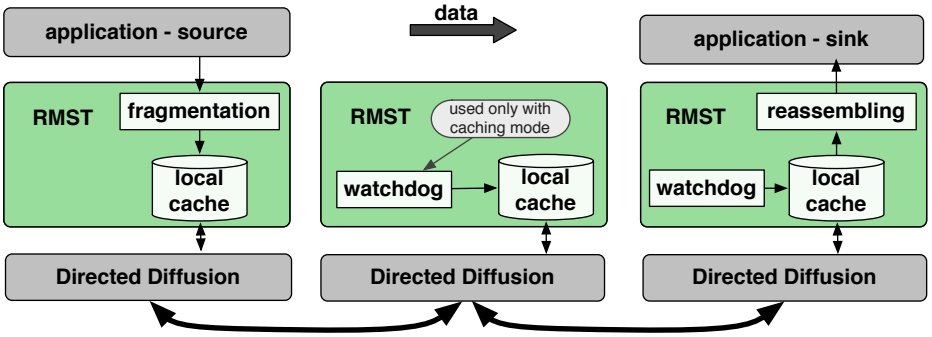


Figure 2.49: RMST in a WSN network stack.

The protocol designers assume a low number of bytes in flight and that the intermediate nodes can completely cache this amount of data. The authors experienced and stated that for packet loss rates below 10%, the caching and negative acknowledgment mechanism is more efficient than a reliable link layer approach based on ARQ due to the overhead caused by link level acknowledgments [86].

2.6.3 Packet Aggregation Mechanism

Packet aggregation mechanisms combine packets from different sources to reduce the total number of packets forwarded in the network. Reducing the number of required packets may preserve energy and increase the throughput by reducing the overhead by physical preambles, headers, CRC, etc. Moreover, at high traffic load, packet aggregation can prevent or at least decrease congestion by reducing internal interferences and the number of concurrently forwarded packets. Figure 2.50 shows a small WSN using a data centric routing protocol and packet aggregation to reduce the number of forwarded data packets to the minimum.
Generating an optimal packet aggregation routing tree in a WSN is an NP-hard problem. Krishnamachari et al. [49] introduced the following three suboptimal aggregation heuristics to solve the problem of establishing an aggregation routing tree:

- **Center at Nearest Source (CNS):** The source node nearest to the sink performs packet aggregation. All other source nodes send their packets directly to this source node.

- **Shortest Paths Tree (SPT):** All source nodes send their packets on the shortest path to the sink. Packet aggregation is performed on the nodes where two or more individual path are overlapping.

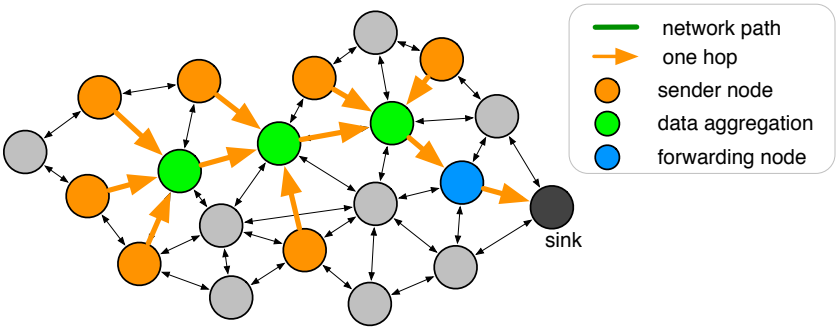


Figure 2.50: Packet aggregation with a data centric routing protocol.

- **Greedy Incremental Tree (GIT):** With GIR, the packet aggregation routing tree is constructed sequentially. Initially, the sink and the source node with the shortest path to it build the initial tree. In the next step, the source node that is closest to the existing tree is connected and builds a new tree. This step is repeated until all the source nodes are part of the tree.

In IEEE 802.11 networks, packet aggregation is a subject of ongoing research [54, 55, 74, 77]. A main target of packet aggregation in IEEE 802.11 networks is achieving a high throughput. Packet aggregation can be handled on different levels in the IEEE 802.11 network stack. The following list introduces three common aggregated IEEE 802.11 frame types:

- **Aggregated MAC Service Data Units (A-MSDUs):** Multiple MSDUs are aggregated into a single MAC Protocol Data Units (MPDU). A-MSDUs show the lowest total length of the physical frame, due to the lowest link layer header overhead. An A-MSDU is protected by only one link layer CRC.
- **Aggregated MAC Protocol Data Units (A-MPDUs):** Multiple MPDUs are aggregated into a single Physical Service Data Unit (PSDU). With A-MPDUs every individual aggregated packet keeps an own CRC.
- **Aggregated Physical Protocol Data Units (A-PPDUs):** Multiple PSDUs are aggregated into a single Physical Service Data Unit (PSDU). With A-PSDUs every individual aggregated packet keeps its own CRC. Moreover, A-PSDUs enable aggregation of packets with different destination addresses.

Table 2.7 compares the three IEEE 802.11 frame aggregation mechanisms:

2.6.4 Back Pressure Mechanisms

Back pressure mechanisms adapt the traffic load in a network to avoid a network collapse caused by congestion. The high number of individual source nodes in

Characteristic	A-MSDU	A-MPDU	A-PPDU
Link layer computing overhead	High	Middle	Low
Overhead on the physical layer	Low	Middle	High
Impact of single bit errors	High	Low	Low

Table 2.7: Characteristics of different IEEE 802.11 frame aggregation mechanisms

large scale WSNs may generate too high traffic load near the sink node. Figure 2.51 depicts the problem by an example. Five source nodes are sending traffic towards a sink. The closer to the sink a node is, the higher is its traffic load on a single hop.

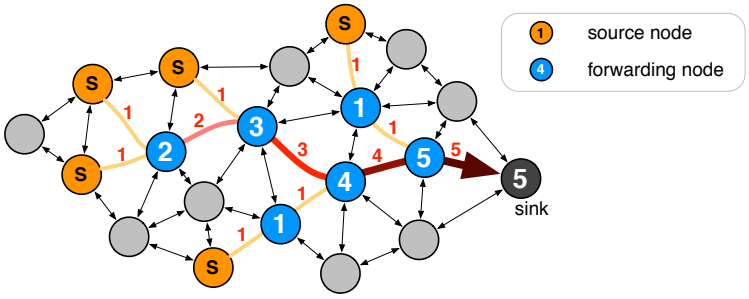


Figure 2.51: Increasing traffic load in a WSN.

Back pressure mechanisms try to control the output of the individual source nodes to keep the traffic load on every intermediate link between the source and the sink node at a working level. Back pressure mechanisms can either be realized on an end-to-end or a hop-by-hop bases. For example, the TCP flow control mechanism tries to adapt the traffic load of a TCP connection on an end-to-end bases. Therefore, the receiver announces to the sender a TCP window size no larger than it can buffer. Unfortunately, this mechanism does not scale well with a large number of simultaneous TCP connections [67].

2.7 Conclusions

In this chapter, we introduced the relevant related work in the area of energy preserving mechanisms and reliability techniques for WSNs. We described different sensor node platforms including the radio module characteristics for real world WSNs. Moreover, we introduced different evaluation tools and existing WSN communication protocols to analyze and compare the contributed protocols of this thesis. In the next chapter, we present the pre-evaluation of the available radio modules presented in Section 2.1. With the help of the pre-evaluation we identify the most suitable radio module to design and implement an energy preserving real

world WSN communication stack supporting reliable data transmissions in heterogeneous networks.

Chapter 3

Hardware Pre-Evaluation

The pre-evaluation of the available radio modules is an essential part in the design process of an energy efficient and reliable communication protocol for real world sensor nodes. The selected radio module determines the basic energy consumption as well as the basic conditions for the protocol design. Some radio modules require a compliant link layer header to autonomously execute link layer tasks such as sending acknowledgments. This enables time and energy efficient link layer operations, but requires using a given format of the physical and link layer header. Some radio modules enhance the robustness of the physical channel by using spread-spectrum techniques. Therefore, the evaluation and selection of the radio module is an essential task for reaching the major targets of this thesis.

To evaluate the real world performance of a radio module, the used communication interface to the microcontroller has to be taken into account. Using bit/byte-oriented radio modules, the microcontroller has to exactly schedule every individual bit/byte read and write operation to the radio module. Otherwise, transmissions may fail. Therefore, we have used existing sensor node platforms equipped with the individual radio modules for the evaluation. To determine the performance of the different radio modules, we analyzed the following characteristics:

1. **Required energy and time to forward data:** The required energy for sending and receiving data should be as low as possible. The shorter the required time for a transmission, the longer the radio module can be turned off.

2. **Required time and energy for a channel check:** A standard operation of all radio duty cycling MAC protocols is the channel check. If no traffic has to be forwarded, the radio module is used for periodically checking the channel.

3. **Bit error rate and robustness against interferences:** Packets received with bit errors have to be recovered either by error correction codes or retransmission mechanisms. Both mechanisms require additional time and energy.

4. **Connectivity between different sensor node platforms:** To support heterogeneous networks, the physical and the link layer should meet a well-

established standard.

We used the sensor nodes introduced in Section 2.2 to evaluate the radio modules introduced in Section 2.1. Table 3.1 presents a brief summary of the different sensor nodes characteristics. We use two different sensor node platforms featuring the packet oriented CC2420 radio module and three sensor node platforms featuring different bit/byte oriented radio modules. Using two sensor node platforms featuring a CC2420 radio module allows testing the direct communication between two heterogeneous sensor node platforms. We configured the bit/byte-oriented radio modules with a physical data rate of 38.4 kbps to ensure an optimal energy consumption. Our experiments have shown that higher transmission rates are not working properly on the used sensor nodes. This is due to the limited performance of the microcontroller and the required accuracy of the write and read operations on the radio module. Most of the available real world implementations made for the BTnode, MSB430 and ESB sensor nodes use even lower physical radio transmission rates of 9.6 kbps or 19.2 kbps. The lower the transmission rate is, the higher is the required energy to send and receive the same amount of data. We used Manchester coding (see Section 2.1.3) to ensure the synchronization of the radio module receiver clock. When using Manchester coding, the link layer data rate is half the physical transmission rate. The packet-oriented radio module CC2420 uses a physical transmission rate of 2'000 kbps. The direct sequence spreading modulation technique used by IEEE 802.15.4 reduces the link layer data rate to 250 kbps.

Sensor node	**Radio module**	**Modulation**	**Physical data rate (kbps)**	**Link layer data rate (kbps)**
telosB	CC2420	DSSS	2000	250
MicaZ	CC2420	DSSS	2000	250
MSB430	CC1020	OOK	38.4	19.2
ESB	TR1001	OOK	38.4	19.2
BTnode	CC1000	BFSK	38.4	19.2

Table 3.1: Evaluated sensor node platforms and corresponding radio modules

The remainder of this chapter is structured as follows. Section 3.1 analyzes the required time and energy for forwarding data. Then Section 3.2 evaluates the required time and energy to perform a channel check. Section 3.3 compares robustness against interferences. Section 3.4 shows which radio modules are compatible to meet a standard.

3.1 Required Energy and Time to Forward Data

This section describes the evaluation of the energy required for forwarding data. First, we determined the energy that is required to send a single byte. Then, we

measured the minimal energy required by a duty-cycled radio module to send a complete frame. Finally, we evaluated the minimal energy required for forwarding a single frame. The results allow comparing the energy efficiency of the different radio modules. Additionally, they make it possible to calculate the energy requirements for a hypothetical link layer protocol featuring the smallest possible energy consumption for every radio module. The hypothetical link layer protocol is used to verify the efficiency of our final real world implementation.

Figure 3.1 shows the experimental setup to measure the transmission time and energy consumption of a radio module. The experimental setup enables the recording of the electric current of one sensor node by the RIGOL DM3052 digital multimeter introduced in Section 2.4.3. We soldered copper wires to one sensor node of each individual platform. The copper wires are used to connect them to the RIGOL multimeter and the VOLTCRAFT VLP-1303 PRO power supply, as described in Section 2.4.3. A second sensor node is connected over a serial connection to a desktop computer. This enables monitoring the traffic during a running experiment. All sensor node platforms are running Contiki as operating system. The RIGOL multimeter is connected over a USB interface to a notebook, which runs the measurement application. We used the maximal recording resolution of 50'000 samples per second. The measuring accuracy of the RIGOL multimeter is $\pm 2\ \mu A$. The sensor nodes are placed in distance of 0.5m to ensure high SNR and to minimize the impact of external interferences. Internal interferences are excluded by sending one packet after each other.

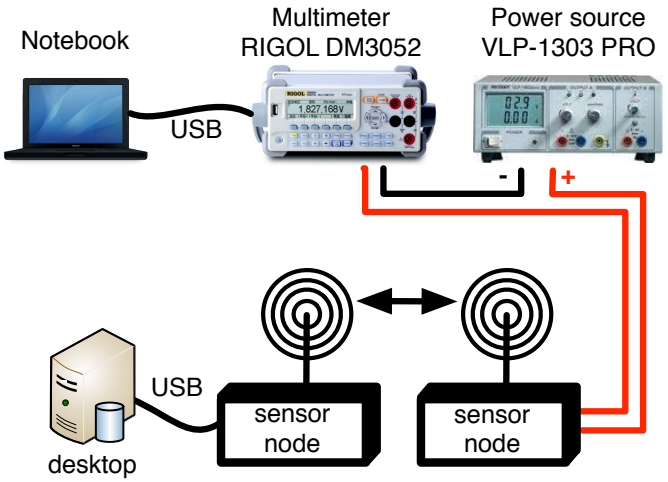


Figure 3.1: Testbed setup to measure the electric current.

3.1.1 Energy Required to Send a Single Byte

First, we used the testbed depicted in Figure 3.1 to determine the required energy for sending one single byte. For every sensor node type, we performed 20 record-

ings with the RIGOL multimeter. During each recording period the node connected to the RIGOL multimeter sent 30 packets within an interval between packets of 500 ms and a payload of 100 bytes. This results in 600 analyzed packets for each sensor node type.

Between the individual transmissions, the radio module is turned off for two reasons. First, the switch-off is used to determine the amount of energy used by the radio module. To determine the energy used by the radio module, the energy costs during the sleeping period are subtracted from the energy measured during packet transmission. Second, turning off the radio helps to determine the time required by the individual radio modules to transmit the packets. We compared the measured time period to the expected time period for verifying the experiment setup.

Figure 3.2 shows an example of such an energy profile recorded with the RIGOL multimeter. The radio module is only turned on to transmit 100 bytes. This results in an energy profile with a distinguishable transmission period. During the transmission, a total of 19.5 mA is recorded by the RIGOL multimeter. Turning off the radio between the individual transmissions reduces the energy consumption to 0.48 mA.

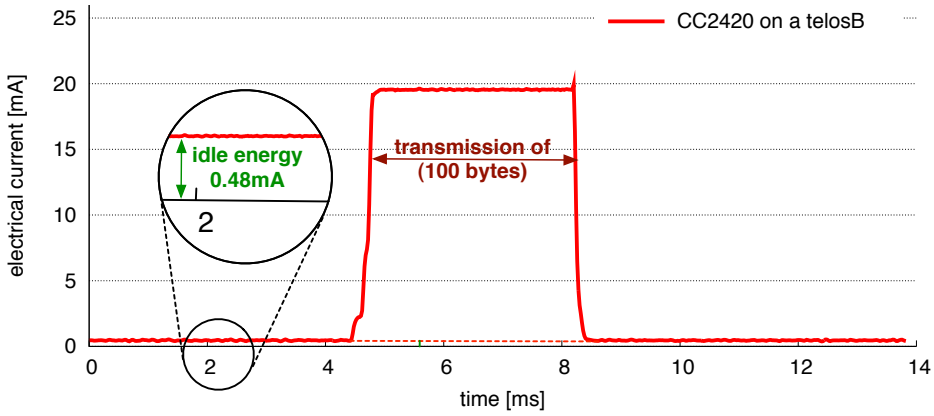


Figure 3.2: Sending 100 bytes with a CC2420 radio module.

Figure 3.3 shows the energy for **sending one single byte** by the different radio modules. The corresponding energy costs are calculated using the energy measurements for sending 100 bytes. All additional energy costs caused by other considered sensor node components as well as switching the radio module are excluded from the calculation. The CC2420 radio module is most energy efficient when transmitting only one single byte. It requires several times less energy and time than every other radio module needs for this operation due to the higher bit rate of the CC2420 radio module. There is no difference in the energy consumption of the CC2420 radio module on a telosB or a MicaZ sensor node. In sleep mode, i.e. with turned off radio modules, the MicaZ requires more energy than the telosB. This difference is due to different electrical components included in these two nodes. The

TR1001 radio module shows the lowest energy usage for sending a single byte with a byte-oriented radio module. The energy requirement for sending a single byte provides a basic indication about the energy efficiency of the different radio modules.

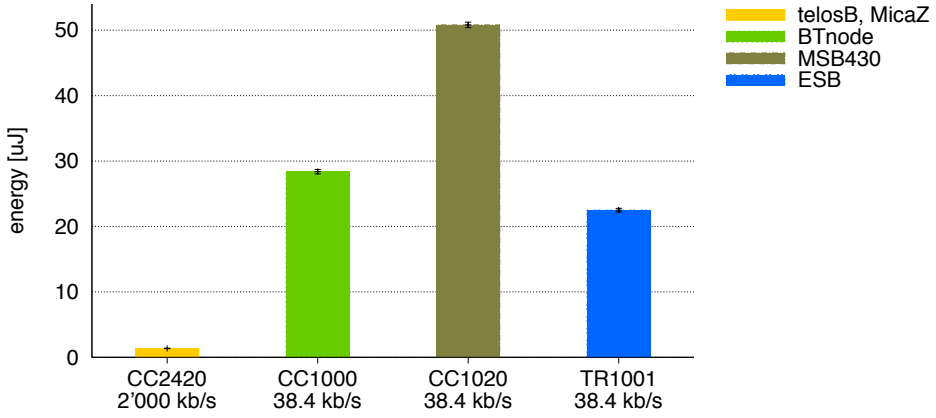


Figure 3.3: Energy required by different radio modules for sending one byte.

3.1.2 Minimal Energy Required to Send a Single Frame

Next, we measured the minimal energy required by a duty-cycled radio module for sending a complete frame. The operation process of a radio duty cycle starts with the wake-up of the radio module followed by a channel check to ensure a free radio channel. Then, a single frame with 50 bytes link layer payload is sent. After the last byte has been transmitted, either the radio module is switched to listening mode or immediately turned off. The radio module switched to listening mode waits for an incoming acknowledgment before it is turned off. If no acknowledgment is expected, the radio module is immediately turned off. This setup determines the smallest possible energy consumption to forward a single frame.

We used the testbed depicted in Figure 3.1 to measure the required energy. We performed 25 measurements with and without acknowledgments. In each measurement, the node connected to the RIGOL multimeter sent 30 frames with a sleep interval of 500ms between the individual transmissions. The second node remains always in listening mode to receive the packets and to send back an acknowledgment if required. We dropped all measurements where the channel was busy during the channel check to exclude the effects of interferences. For the evaluations with acknowledgment, we additionally dropped all the measurements where no acknowledgment was received.

Figure 3.4 shows the minimal energy required by these radio modules to **send a single frame** with 50 bytes payload. The percentage on top of the bar shows how much of the total energy is actually used to send the payload. The rest of the energy

is required for checking the radio channel and switching the radio module on and off. The CC2420 radio module requires 10 additional link layer header bytes to fulfill the IEEE 802.15.4 requirements. Again, the CC2420 radio module is the most energy efficient when transmitting a single frame. Using acknowledgments requires around 15%-20% additional energy.

When comparing Figure 3.4 to previous Figure 3.3 we see that the TR1001 radio module requires comparatively more energy to transmit a frame. In Figure 3.3, depicting the required energy to transmit a single byte, the TR1001 radio module requires 56% less energy than the CC1020 and 21% less than the CC1000. However, in Figure 3.4, the TR1001 radio module requires 43% less energy than the CC1020 and even 7% more than the CC1000 without acknowledgment. The difference in energy consumption is basically caused by the ESB sensor node, which requires disproportionately more time to power on the radio module then the other sensor nodes.

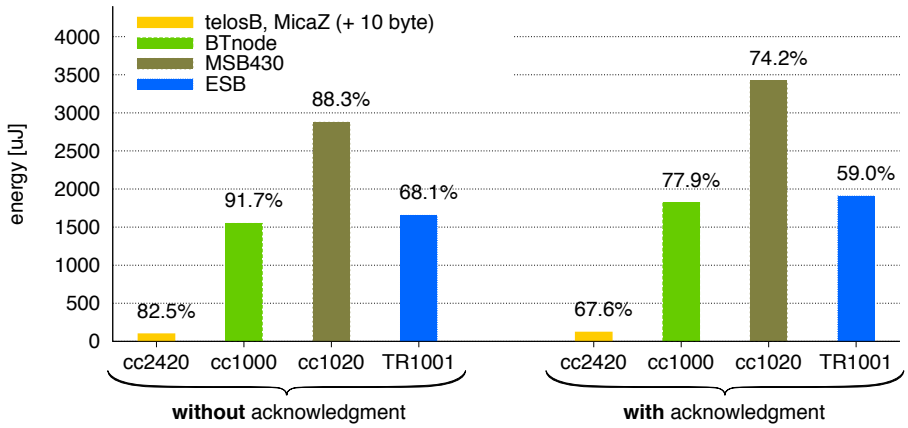


Figure 3.4: Energy required by different radio modules for sending a 50 bytes payload.

3.1.3 Minimal Energy Required to Forward a Single Frame

Finally, we determined the minimal energy required by a duty-cycled radio module to forward a packet with 50 bytes payload. Forwarding a single frame includes the energy required to receive and handle routing of a packet in the network stack. We reuse the testbed to measure the electric current as depicted in Figure 3.1. The sensor node connected to the desktop PC sends a packet every 500 ms to the sensor node connected to the RIGOL multimeter. This sensor node immediately forwards every incoming packet back to the source node. As in the previous experiment, presented in Section 3.1.4, the radio module is immediately turned off after the frame has been forwarded or the acknowledgment has been received. The radio module is turned on 100ms before the next transmission of the packet is expected. The energy used during idle listening is removed from the energy costs to forward

a single frame. We performed 25 measurements with and without acknowledgment. In each measurement, 30 frames with a link layer payload of 50 bytes are forwarded. We dropped all failed forwarding attempts.

Figure 3.5 shows the minimal energy required to **forward a single frame** with 50 bytes payload. The percentage on top of the bar shows how much of the total energy is actually used to send the payload. The remaining energy is required for checking the radio channel and switching the radio module on and off. It requires around twice the energy to forward a packet than just transmitting it. The CC2420 is the most energy efficient radio module when forwarding a single frame. The most inefficient radio module is the CC1020 with almost twice the energy consumption of the CC1000 and TR1001.

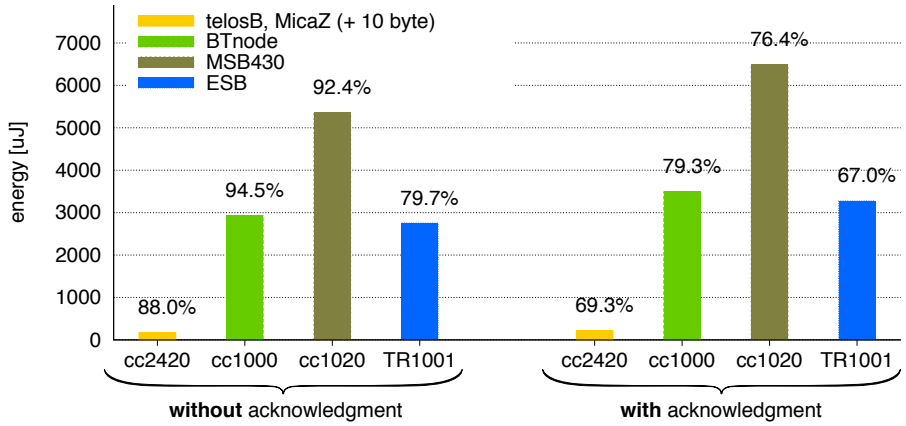


Figure 3.5: Energy required by different radio modules for forwarding a 50 bytes payload.

3.1.4 Hypothetical Reference Link Layer Protocol

We defined a hypothetical link layer protocol featuring global knowledge of the entire WSN. This protocol has the lowest possible energy consumption for every of the evaluated radio modules. The authors of [29] introduced a MAC protocol called ideal protocol, which serves the same purpose. Our ideal protocol can predict the time period of every packet transmission and any other source of interference. Every sender knows the exact point in time to transmit a packet to a receiver without interfering with any source of interference. The receiver knows the exact point in time to wake up for receiving a frame. There are no bit errors and therefore no packet retransmissions. Moreover, no acknowledgements and no channel checks are required by this protocol. The protocol only turns on the radio module to send or receive a frame without any idle listening. The hypothetical link layer protocol depicts the most energy efficient and reliable protocol design that is feasible. We use the hypothetical link layer protocol in Section 5.6 for an energy benchmark and verification of our real world implementation. The energy required by the differ-

ent radio modules to forward 50 bytes with this hypothetical protocol is shown in Figure 3.6.

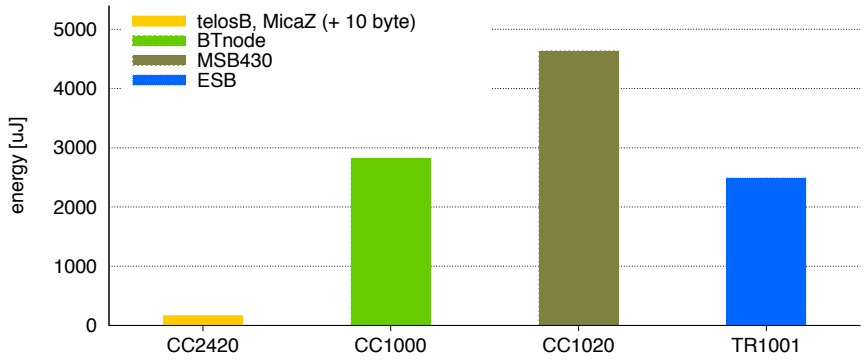


Figure 3.6: Energy for forwarding a 50 bytes payload with a hypothetical MAC protocol.

3.2 Energy Required to Check the Radio Channel

A sensor node is waiting for incoming data most of the time. During this time, a duty cycling MAC protocol periodically performs channel checks. For a channel check, the radio module is turned on and switched to listening mode. If no transmission can be detected, the radio module is switched off again. The periodical channel check is one of the main energy consumers during time periods without traffic or in case of very low traffic rate. Therefore, we determined the minimal energy required by the individual radio modules to perform a channel check. We connected one sensor node to the RIGOL multimeter to record the energy profile during wake-up, checking the channel and going back to sleep. The radio modules perform 10 channel checks per second for a time period of 20 seconds. We performed 20 recordings for each radio module. Sometimes interferences are misinterpreted as an ongoing transmission. We ignored these wrongly detected frames in our evaluation.

A radio channel check starts by turning on the radio module and switching it into listening mode. Then, a channel check is performed before the radio module goes directly back to sleep again. For each radio module we used the most efficient available detection technique. Therefore, the detection mechanism used for the bit/byte-oriented radio module searches for an ongoing transmission of a specific physical preamble. The CC2420 uses the built-in channel check support mechanism introduced in Section 2.1.7.

Figure 3.7 shows the measured energy to perform a single channel check during periods without traffic. Energy used by other components of the sensor node platform has been excluded. Therefore, the major reason for the differences in the

energy consumption is the time required to perform the channel check. The time periods required by the different radio modules to perform a channel check is written to the legend in Figure 3.7. The radio modules require different time periods after wake up to achieve a state of the receiving circuit to correctly demodulate the incoming signal. The CC2420 is the fastest and most energy efficient of the evaluated radio modules. The TR1001 requires a very long time to wake up on the ESB node. This time is several magnitudes longer than specified in the manual of the TR1001.

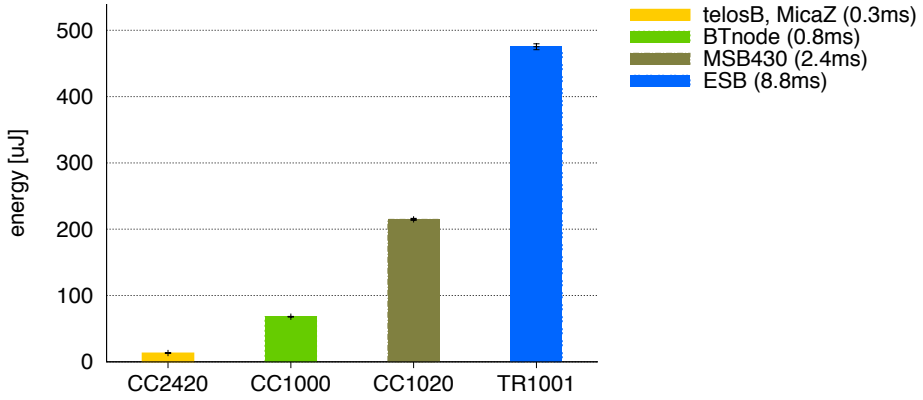


Figure 3.7: Energy for a channel check without traffic.

3.3 Robustness against Interferences

The radio modules should be as robust as possible against interferences to minimize bit errors. Dealing with interferences is one of the most challenging tasks for real world WSN implementations. As long as only one packet is forwarded in a WSN, no internal interferences have to be handled. Then packet loss is basically caused by wave propagation effects and external interferences. With increasing traffic load, inter-flow and intra-flow interferences have to be handled as well. Only interferences with a signal strength higher than the receiver sensitivity limit can be directly detected by the radio modules. Lower signal levels cannot be directly detected and are therefore harder to handle. They can cause, for example, a hidden node problem. The lower the actual SNR during the reception is, the higher is the probability of bit errors caused by interferences.

3.3.1 Testbed Setup

The environment shows an impact on the wave propagation effects. Walls and furniture in typical indoor environments cause wave propagation effects such as

reflections, multi-path fading or diffraction. These effects are also occurring in outdoor environments by buildings or trees, but not to this extent. Additionally, indoor topologies, for example an office building, show a high density of electronic devices generating interferences. Table 3.2 shows four different setups to evaluate the impact of the environment, the SNR and internal-interferences.

Scenario	**Topology**	**Interferences**		**SNR**	**Section**
		External	Internal		
1	Indoor	Yes	No	4 Levels	3.3.2
2	Outdoor	Yes	No	4 Levels	3.3.2
3	Indoor	Yes	Yes	4 Levels	3.3.3
4	Outdoor	Yes	Yes	4 Levels	3.3.3

Table 3.2: Different evaluated interference scenarios

We use an indoor and an outdoor testbed to evaluate the influence of the environment. To analyze the impact of the SNR we used four different distances between the sender and the receiver. We used a third node to simulate internal interferences. The outdoor testbed has been established on a plane meadow. The nodes were placed 1m above ground on tripods as depicted in Figure 3.8. The selected four distances between the outdoor nodes are 10m, 50m, 100m and 250m.

Figure 3.8: TelosB node on tripod in outdoor testbed.

The indoor testbed has been established in the building of the Institute of Computer Science and Applied Mathematics (IAM). Figure 3.9 shows setup of the indoor testbed. A notebook is connected to the receiver node to record the sequence numbers of the received packets (number 1 in Figure 3.9). The sender node is placed in different distances to the receiver node (numbers 2-5). Both nodes are placed 1m

above ground on tripods. The shortest distance with the highest SNR was 0.5m. For the second distance, the same two nodes were placed in neighboring rooms. The connecting door was closed. For the next larger distance two rooms at the opposite side of a corridor were used. The largest distance, with the lowest SNR, was made by choosing two rooms that are two floor levels apart and on the opposite side of the corridors. The SNR decreases with increasing distance and amount of obstacles. The lower the SNR is, the higher is the bit error rate. An additional node is able to generate internal interferences if required (number 6).

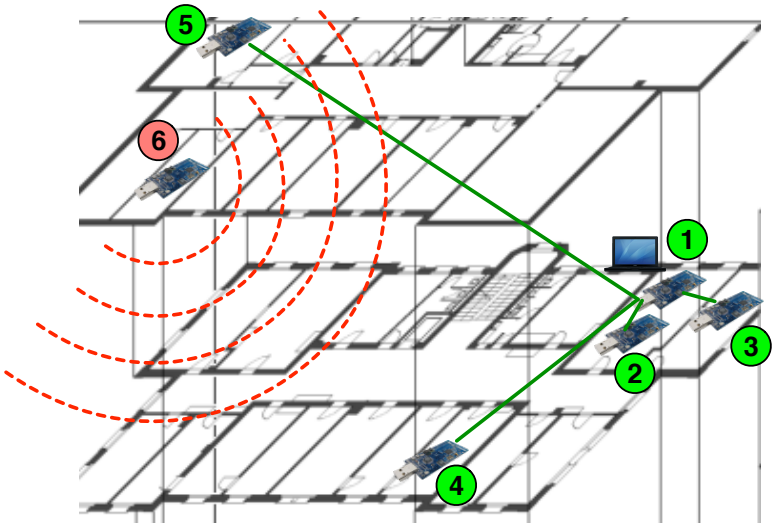


Figure 3.9: Indoor evaluation testbed setup.

We used the Contiki operating system version 2.4 using an out-of-the-box configuration. The network stack uses μIP and nullRDC link layer protocol without duty cycling the radio module. For the different bit/byte-oriented radio modules, namely C1000, CC1020 and TR1001, we use a physical transmission speed of 38.4 kbps with a Manchester encoding to enhance reliability. All other reliability mechanisms are deactivated. If a packet gets lost, it is not retransmitted. We operate all radio modules with the maximal available transmission power. For every distance, we performed a set of 32 measurements. Each of the 500 packets was sent with a data payload of 50 bytes. The time between two packets was 250ms. We use packet loss as metric to express the robustness of the different radio modules against interferences. A packet is lost if the frame start was missed or the received frame contained a bit error.

3.3.2 Packet Loss Caused by External Interferences

Figure 3.10 shows the packet loss for the different SNRs inside the IAM building without internal interferences. The nodes equipped with the CC2420 (telosB and

MicaZ) show significantly better robustness against interferences than all other nodes. The micaZ node shows a slightly lower packet loss rate than the telosB. This might be caused by a higher gain of the external micaZ antenna. The BTnode, equipped with a CC1000, showed the most bit errors. A brand-new BTnode comes without any antenna. Under this condition, BTnodes are not able to send farther than 1 m distance. Therefore, we added a $\lambda/4$ monopole antenna to all the BTnodes. The results additionally show that a high transmission power does not necessarily result in a lower packet loss rate.

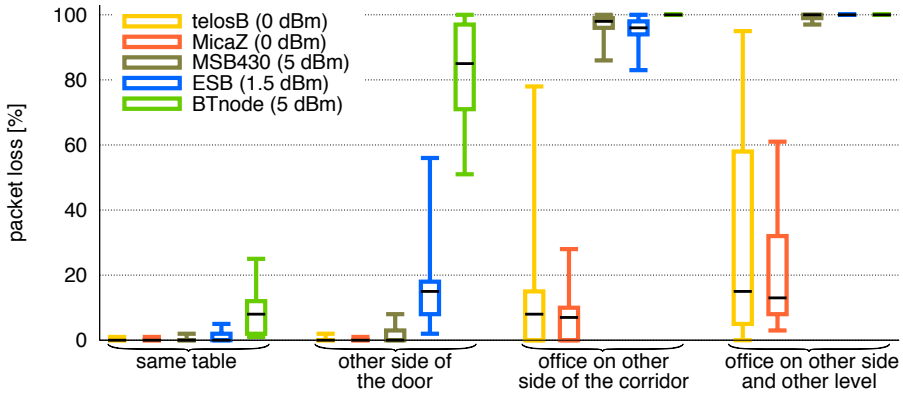


Figure 3.10: Indoor packet loss with external interferences.

Figure 3.11 shows the packet loss for the different SNRs, without internal interferences in the outdoor testbed. In the outdoor testbed, the packet loss is significantly lower than indoor, as expected. Especially, the MSB430 shows outdoor a significantly lower packet loss than indoor. This may be an indicator that the CC1020 radio module has some effort with wave propagation effects caused by reflections,

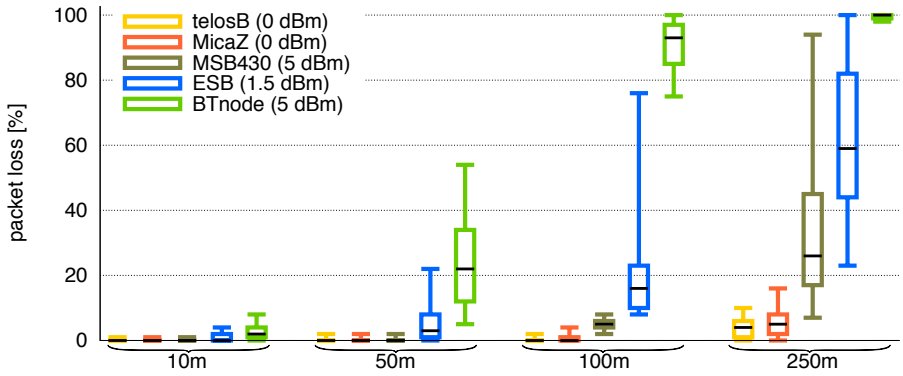


Figure 3.11: Outdoor packet loss with external interferences.

multi-path fading or diffraction. The CC2420 has a significantly lower packet loss rate than the other radio modules. The CC1000 on the BTnode still shows high packet loss.

3.3.3 Packet Loss Caused by External and Inter-flow Interferences

Transmission by neighbor nodes can cause massive intra-flow or inter-flow interferences. In this section, we perform some basic tests, to estimate the performance of the radio modules with inter-flow interferences. The same testbeds like in the preceding sections are used. An additional neighbor node produces traffic with transmission power and distance adapted to the sensing range of the receiver node. The additional node continuously sends single packets with 50 bytes payload. Between packet transmissions, it uses a random back-off window of two to five times of the time required to transmit a frame. The chosen distance and transmission power makes it hard to detect the ongoing transmission, but still generates significant interferences at the receiving node. More complex measurements including intra-flow interferences are made in the evaluation part of Chapter 5.

Figure 3.12 shows the packet loss with inter-flow interferences inside the IAM building. The sending node tries to send a packet every 250 ms. It tries to retransmit the packet twice if the channel was busy. As expected, all radio modules show an explicitly higher packet error rate than indoor without inter-flow interferences (see Figure 3.10). The CC2420 radio module still shows the lowest packet loss rate. The interferences caused by the third BTnode are limited.

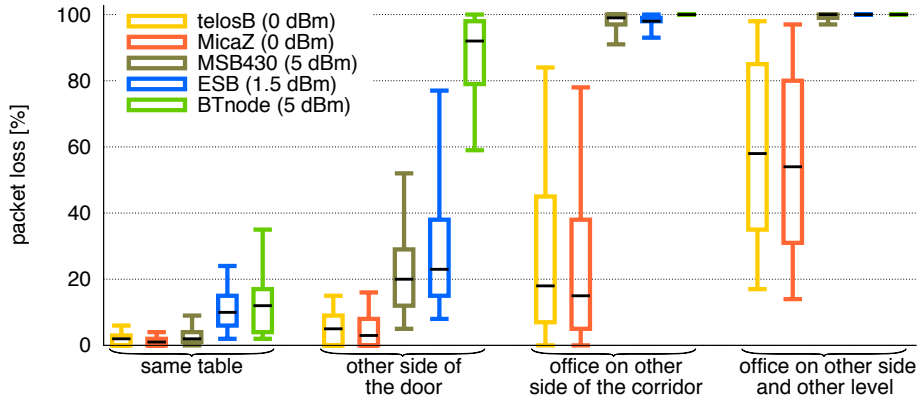


Figure 3.12: Indoor packet loss with external and inter-flow interferences.

For the outdoor measurements, the third node is placed in 300m distance to the sender and receiver node. Figure 3.13 shows the packet loss for different distances between sender and receiver in the outdoor testbed. The additional interferences generated by the third node result in higher packet loss. Under external interferences, the CC1020 showed a lower packet loss than the TR1001. With inter-flow

interferences the TR1001 now has a similar packet loss rate than the CC1020. The results show that the CC2420 radio module is the most robust radio module. This is basically caused by significantly higher process gain of the DSSS mechanism defined by IEEE 802.15.4. Manchester coding used for the byte-oriented radio modules shows only a low process gain.

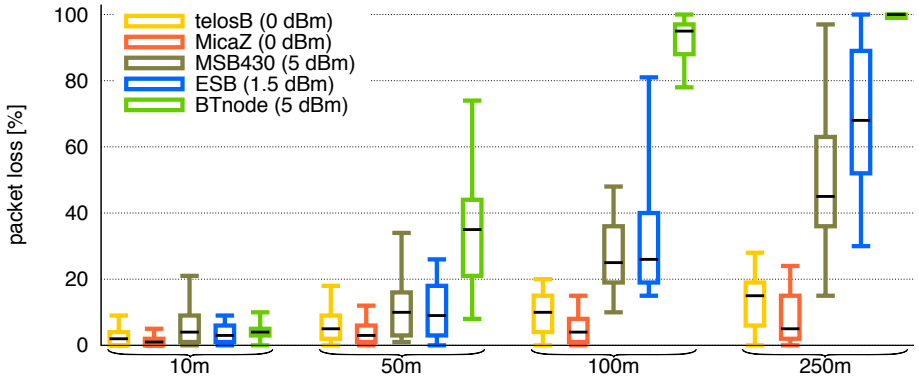


Figure 3.13: Outdoor packet loss with external and inter-flow interferences.

3.3.4 IEEE 802.11 Interferences

We evaluated the interferences caused by IEEE 802.11 transmissions to the CC2420. The CC2420 radio module is operating in the same 2.4 GHz ISM radio band like IEEE 802.11 networks. We used the WISEBED WSN real world testbed at the IAM building, introduced in Section 2.4.2, to estimate the influence of the installed IEEE 802.11 networks to transmissions made by telosB nodes. We used a simple network topology of six telosB nodes in a row featuring a high SNR. To avoid internal interferences, we sent a packet only every two seconds from node 1 to node 6. This is sufficient time to pass all five hops. We used Contiki 2.4 wit nullRDC and no additional reliability functions. If a packet gets lost, it is not retransmitted. We sent 200 packets on each channel before switching to the next channel. We repeated the experiments twenty times during an entire workday and twenty times after midnight. Figure 3.14 shows the results of this measurement. IEEE 802.11 access points that are close to the tested nodes, probably cause the peaks at channel 12 and 17. One access point with lower impact seems to be operating in the range of channel 22. There is a slightly higher packet loss during daytime in the IEEE 802.11 frequency range from channel 11 to 24. The channels 25 and 26 outside of the IEEE 802.11 frequency spectrum show almost no packet loss during night. At work time, there was a time period of very high packet loss. A student using the telosB nodes for his Bachelor thesis generated traffic in a local testbed on channel 26. This traffic interfered with our measurements. The advantage of channel 26 is

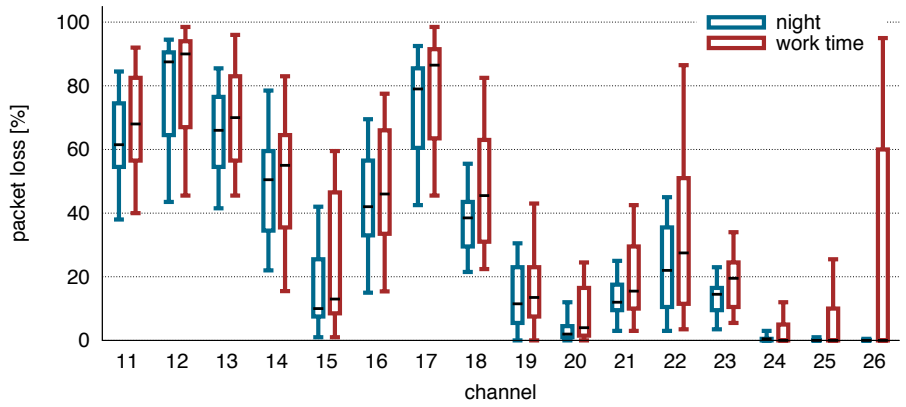


Figure 3.14: Packet loss over 5 hops on different channels and different daytimes.

that it avoids interferences by IEEE 802.11 networks. For our future evaluations we use channel 26 and ensure that no other testbed is enabled at the same time period.

3.4 Network connectivity

The available sensor node platforms differ in computing power, sensing capabilities, energy consumption and used radio module. To resolve complex environmental sensing requirements, different sensor nodes with specific characteristics have been selected to fit to the particular operation purpose. The CC2420 radio module provides an IEEE 802.15.4 conform physical interface to enable direct communication between different sensor node platforms. The sensor node platforms telosB and MicaZ are both using a CC2420 radio module. This makes it possible to communicate directly between two different sensor nodes using different microcontroller platforms. Direct communication between the two sensor node platforms has been working perfectly fine. Also the autonomous acknowledgment function of the CC2420 is working perfectly between telosB and MicaZ. Every radio module that is conform to IEEE 802.15.4 should be able to directly communicate to telosB and MicaZ nodes.

3.5 Hardware Pre-Evaluation - Conclusion

The results show that the CC2420 radio module is the most energy efficient and robust radio module. The DSSS mechanism of the CC2420 is significantly more robust against interferences than the bit/byte-oriented radio modules. It is the only radio module that supports a standardized physical layer and a link layer to enable direct communication between heterogeneous sensor nodes. The CC2420 is widely used in different sensor nodes such as telosB, micaZ, sensinode or imote2. The

link layer functions offered by the CC2420 radio module reduce the load of the microcontroller and ensure a precisely predictable transmission timing. Therefore, we select the CC2420 radio module to design our energy efficient and reliable link layer protocols. In the next chapter we introduce our contributed WSN protocols.

Chapter 4

WSN Protocols

This chapter introduces our contributed WSN protocols and the resulting network stack. Two of our contributed protocols are located on the link layer, one on the application layer. The link layer protocol offers the main functions to provide energy efficiency and hop-to-hop reliability. The third protocol is an application layer overlay protocol. It provides end-to-end reliability for UDP data flows.

Our link layer protocol stack is divided into two sub-layers to reduce the complexity as shown in Figure 4.1. The lower layer handles the access to the physical channel and the radio duty cycle mechanism to preserve energy. The upper layer ensures hop-to-hop reliability. The lower layer is called *radio duty cycle* (RDC) layer, according to the naming used by the Contiki [23] operating system. On the other side Contiki names the upper layer the *MAC* layer, which can lead to confusions. To avoid misunderstandings, we named this layer *hop-to-hop reliability* (H2H) layer.

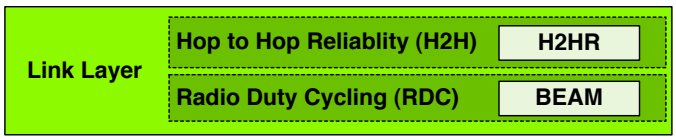


Figure 4.1: The two sub-layers of the link layer protocol stack.

On the RDC layer we implemented the *Burst-aware Energy-efficient Adaptive MAC* protocol (BEAM) [3, 5]. BEAM is suitable to work with all IEEE 802.15.4 compliant radio modules such as the CC2420. It uses adaptive radio duty cycle mechanisms to optimize the wake-up frequency corresponding to current network load. BEAM uses positive acknowledgments and link condition information to support the reliability functions of the H2H layer. We developed a traffic prediction mechanism to enable BEAM to determine the optimal duty cycle.

On the H2H layer, we implemented the *hop-to-hop reliability protocol* (H2HR) [3, 97]. H2HR ensures reliable hop-to-hop forwarding by local retransmissions. Most retransmissions are required during periods with high internal interferences

and cause additional packet loss. Therefore, handling reliability at high traffic loads is a major challenge for adaptive hop-to-hop reliable protocols. Main functions for H2HR are detecting of traffic congestion and calculating an optimal transmission delay for a new or a retransmitted packet. H2HR uses different information sources from one-hop neighbor nodes to determine the current conditions of the physical channel. An optimal retransmission delay reduces packet collisions and retransmission during a high traffic load period. This preserves energy, minimizes internal interferences and decreases the traffic load and resulting interferences to a noncritical level.

On the application layer we developed the *UDP end-to-end reliability* (UDP-E2E) protocol to add an end-to-end reliability mechanism to UDP unicast flows. The design of UDP-E2E is based on the sequence number mechanism of RMST [86]. UDP-E2E uses negative acknowledgments to request retransmissions of lost packets. It requires significantly less signaling packets than TCP to handle low end-to-end packet loss rates.

To support the design process we started with implementing different protocol versions in the OMNeT++ network simulation framework [2]. The OMNeT++ simulator enables extensive and repeatable comparisons and analyses of the different protocol variations in a short time period. This enables improving the efficiency, reliability and robustness of the protocols. Basic prerequisites for meaningful results were a realistic model of radio channel interferences and a detailed radio module implementation. Afterwards, we migrated the protocols to the network stack of Contiki. This enables testing the protocols on real sensor nodes. First, we used small scenarios to evaluate and improve the basic functions of our protocols. Second, we used the WISEBED WSN real world testbed with different larger network topologies to verify and compare the performance to other protocols.

The remainder of this chapter is structured as follows. Subsection 4.1 describes the protocol design of BEAM. Subsection 4.2 introduces the hop-to-hop reliability protocol H2HR. Subsection 4.3.2 describes the end-to-end reliability application layer overlay protocol UDP-E2E. Subsection 4.4 shows the embedding of our protocols into the Contiki network stack.

4.1 Burst Aware Energy Efficient Adaptive MAC Protocol

The main purpose of BEAM is to minimize the energy consumption to enhance the lifetime of a WSN. BEAM preserves energy by duty cycling the radio module. The radio duty cycle mechanism periodically turns the radio module on to check the radio channel for incoming packets. Between these channel checks, the radio module is turned off. The wake-up periods of the individual nodes in a WSN are unsynchronized. This prevents synchronization messages and enables dynamic duty cycles. Dynamic duty cycles enable the adaptation of the radio module sleep-

ing periods to the actual traffic load. BEAM additionally implements an ARQ mechanism to support hop-to-hop reliability. BEAM is developed to work best with the energy efficient packet-oriented and IEEE 802.15.4 compliant radio module CC2420 [91] from Texas Instruments. It employs all advanced features of this radio module. The design of the BEAM frame format is IEEE 802.15.4 compliant. The direct support of the IEEE 802.15.4 frame format enables the CC2420 radio module to execute common link layer tasks such as sending acknowledgments or validating the CRC without interacting with the microcontroller as well as the direct communication between heterogeneous sensor nodes.

The remainder of this section is organized as follows. First, Subsection 4.1.1 describes the requirements to implement a radio duty cycle protocol on the CC2420 radio module. Afterwards, Subsection 4.1.2 explains the basic functionality of BEAM. Then several optimizations for BEAM are discussed in Subsection 4.1.3. Subsection 4.1.4 specifies the offered reliability support for the H2H layer. Finally, Subsection 4.1.5 describes the requirements to add FEC support to a RDC protocol such as BEAM.

4.1.1 Impact of CC2420 Characteristics on BEAM Design

The hardware evaluation in Chapter 3 showed that the packet-oriented CC2420 radio module works best for our goals. On one hand, packet-oriented radio modules reduce the flexibility in designing a link layer protocol, but, on the other hand, they are able to efficiently execute common link layer tasks, such as channel checking or acknowledgment handling. Shifting link layer tasks from the microcontroller to the radio module can significantly reduce the execution time of link layer tasks. Additionally, it preserves energy and reduces the code size on the microcontroller. Related work in Subsection 2.6.1 introduced two common approaches for an asynchronous radio duty cycle protocol on a packet-oriented radio module. To estimate the capability of the two approaches, we implemented the basic functions of a LPP and a LPL based protocol in the OMNeT++ network simulator. The evaluations did not show a superior protocol for handling high traffic periods. But under low traffic, LPL based protocols require significantly less energy than LPP based protocols. Therefore, BEAM is designed as an LPL protocol using beacon strobes.

The following subsections explain how to design an IEEE 802.15.4 compliant LPL protocol using CC2420 radio functions.

Energy Efficient LPL Data Transmission Detection with the CC420

An energy efficient channel check mechanism is a basic requirement to achieve an energy efficient link layer protocol. A channel check is required to periodically scan for incoming traffic. This subsection describes LPL based data transmissions and how to minimize energy costs of periodic channel checks for incoming traffic with a CC2420 radio module. Data transmissions with LPL based protocols on a packet-oriented radio module works basically as depicted in Figure 4.2. A node

with a pending packet sends a burst of beacon strobes to announce an upcoming data transmission. At some point in time, the radio duty cycled receiver node wakes up and listens to the channel for a particular time period (1). Within this particular time period, any possibly existing beacon strobe burst can be detected. The shorter the required time period to detect a beacon strobe the less energy is consumed by the radio module if no frame has been detected. If a single beacon strobe has been received that is addressed to the receiver, then an acknowledgment has to be sent back to the sender (2). The acknowledgment enables the sender node to recognize the wake-up of the receiver node. Now, the sender transmits the pending packet to the listening receiver (3).

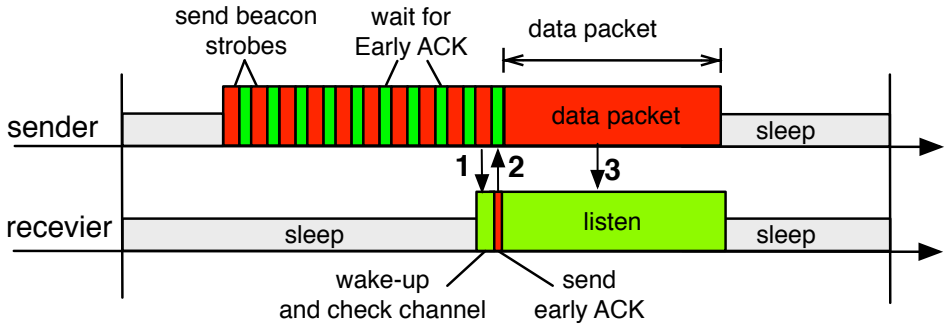


Figure 4.2: Transmission with a LPL based protocol on a packet-oriented radio module.

The length of the required listen period to reliably detect a beacon strobe, determines the energy consumption during very low traffic periods. The required minimum length of a channel check to reliably detect a beacon strobe depends on two factors. The first factor is the frame detection technique used by the **receiver** to recognize an ongoing transmission. The second factor is the time period required by a **sender** to detect the acknowledgment after having sent the last byte of a beacon strobe. This time determines the length of the **silence period** between the single beacon strobe transmissions.

There are four different **frame detection techniques** to recognize a possible beacon strobe by a **receiver**.

1. *First In First Out P*-interrupt: The CC2420 FIFOP-pin triggers an interrupt on the microcontroller after receiving the last byte of a frame. This technique is used by the evaluated XMAC implementation of Contiki described in Chapter 5. The nodes listen to the channel until the CC2420 triggers the interrupt. Using this technique, a frame is detected after its complete reception by the radio module.

2. Checking *Start of Frame Delimiter*: An ongoing frame reception can be detected by checking the SFD pin. The value of the SFD pin is set to 1 after a valid synchronization header has been received (see Section 2.1.7). Frames

with missing synchronization header cannot be detected using this detection technique. This might be the case when the radio module was not yet in listening mode or by too many interferences while the synchronization header was sent.

3. Using *Clear Cannel Assessment*: The CCA function, described in Subsection 2.1.7, makes it possible to detect ongoing transmissions, even if the synchronization header is missing. The value of the CCA pin can be arbitrarily checked by the microcontroller. The time interval between single CCA checks has to be shorter than the time required to transmit a beacon strobe. Otherwise, a beacon strobe could be missed.

4. Consistently reading the RX buffer: Instead of waiting for the interrupt, the RX buffer can also be periodically copied to the microcontroller while the radio module is in listen mode. This works well for short packets with known length, such as XMAC beacon strobes or acknowledgments.

A sender of beacon strobes has to minimize the **silence period** between the individual beacon strobes. The silence period is the time period between two subsequent beacon strobes, when the sender of the beacon strobe is listening to the channel for the expected beacon strobe acknowledgment. The shorter this silence period is, the shorter is the time period that has to be checked by the receiver to ensure that there is no ongoing beacon strobe transmission. LPL requires a silence period to enable the receiver acknowledging the successful reception of the beacon frame. The acknowledgment is usually triggered by the microcontroller, using so-called **software acknowledgments**. Therefore, the microcontroller has to copy the frame from the radio module first. Then, the frame has to be interpreted by the microcontroller, before it can generate the acknowledgment frame and copy it to the TX buffer of the radio module. Finally, the microcontroller has to trigger the radio module to send the acknowledgment.

The packet oriented radio module CC2420 offers an *AUTOACK* function for sending **hardware acknowledgments**. They are handled without any interaction with the microcontroller. Supporting IEEE 802.15.4 compliant link-layer headers enables the radio module to construct link-layer header data, to calculate the CRC and to generate an acknowledgment. These operations are executed while the frame is being received. The elimination of interactions with the microcontroller results in constant and reliable acknowledgment response times. Acknowledgments expected by the sender can be read with a small static delay from the RX buffer after transmission of the last data frame. The response time of acknowledgments handled by the microcontroller depends on the payload size and workload of the microcontroller. Figure 4.3 shows the transmission of beacon strobes if expecting a hardware or software acknowledgments. The *send* parts depict the beacon strobe transmission announcing a pending packet. The incoming delay of the software acknowledgment basically depends on the length of the previously sent data frame. The waiting time is long enough to turn off radio module. The radio module has to

be turned on and switched into listening mode around 300 μs before the software acknowledgment is expected. The 300 μs consist of 250 μs for starting the radio module and 50 μs to compensate the time jitter caused by microcontroller operations of the receiver. Then, the microcontroller reads the RX buffer to check if the received data is an acknowledgment.

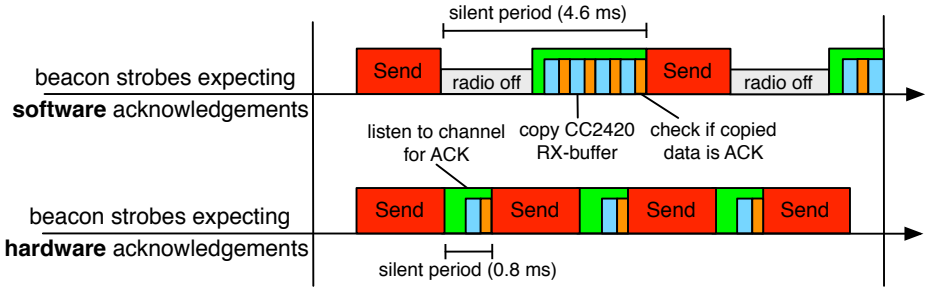


Figure 4.3: LPL MAC beacon strobes for hardware and software acknowledgments.

With the quick and static response time of the hardware acknowledgments, the radio module is switched to listening mode after sending the beacon frame. After a short delay, the acknowledgments can be copied from the RX buffer. As soon as the received frame corresponds to the expected acknowledgment, the data frame is sent. Otherwise, the next beacon strobe is sent. Section 5.2 of the evaluation part describes a detailed performance analysis of the different channel check and acknowledgment handling techniques.

4.1.2 Basic Functionality of BEAM

The design of BEAM is based on the LPL approach introduced in Subsection 2.6.1. We developed two LPL based versions of BEAM. One version is based on the design of XMAC. Like XMAC, it uses **short beacon strobes** to announce a pending data packet transmission. The data part is transmitted after receiving an acknowledgement for a sent beacon strobe. In contrast to XMAC an additional acknowledgement for the data part is sent. The additional acknowledgement is used by our hop-to-hop reliability mechanism, introduced in Section 4.2, to detect packet loss. The other LPL version of BEAM is different from XMAC. This BEAM version uses **beacon strobes including the payload**. The payload is piggybacked to the beacon strobe to avoid the additional data transmission after the beacon strobes.

All transmitted packets are IEEE 802.15.4 compliant to maximize reusability of BEAM and enable the full support of the CC2420 radio module. BEAM uses 16-bit IEEE 802.15.4 sender and destination addresses. To ensure a successful transmission of the data part, every correctly received packet is acknowledged. The next two subsections describe the two LPL based versions of BEAM in more detail.

BEAM using Short Beacon Strobes

Figure 4.4 depicts the BEAM version using short beacon strobes to announce a pending frame. This version is based on XMAC. Four successful transmissions are required to forward a single packet. The numbering of the following list refers to the numbers depicted in Figure 4.4.

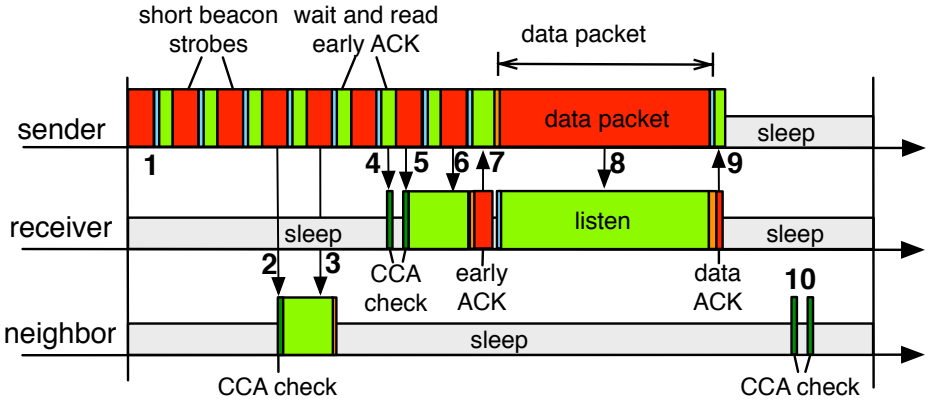


Figure 4.4: BEAM using short beacon strobes.

1. A sender with a pending packet repeatedly transmits short beacon strobes. With the CC2420 radio module, a short beacon strobe is realized as an IEEE 802.15.4 conform frame with an empty *MAC frame payload* addressed to the receiver.

2. A neighbor node wakes up to check the channel. The CCA check detects an ongoing transmission. Therefore, the neighbor node does not turn off the radio module. It stays in listening mode to be able to receive the next frame.

3. The neighbor receives the next short beacon strobe. Since the packet has not been addressed to the node, the radio module is turned off for the remaining duty cycle period.

4. The addressed receiver node wakes up to check the channel. During the first CCA check of the receiver node, the sender is listening for an incoming acknowledgment. Therefore, the receiver cannot detect a transmission with the first CCA check. To save energy, the receiver now shortly turns off the radio module, before the second CCA check is performed.

5. At the second CCA check, the receiver detects an ongoing transmission, but the frame start has been missed. The receiver now waits for receiving the next beacon strobe. The receiver now waits for receiving the next beacon strobe that raises the FIFOP interrupt.

6. The receiver node receives the next short beacon strobe.

7. After receiving the entire frame, the receiver instantly returns an early acknowledgment. With the CC2420 radio module this can be realized by the automatic acknowledgment (*AUTOACK*) without any interaction with the microcontroller. With enabled *AUTOACK* function, an acknowledgment is sent if the receiver address matches, the acknowledgment flag in the link-layer header is set and the CRC was correct. Simultaneously, the FIFOP interrupt triggers the microcontroller to copy the beacon strobe to a local packet buffer. After the beacon frame has been processed, the receiver expects a data packet. To save energy, the radio module can shortly be turned off, as some time is required by the sender to process the acknowledgment before the transmission of the data part can be started.

8. The sender receives the early acknowledgment, which announces that the receiver is ready. Now, the sender starts the data transmission without any channel check. This works as the successful exchange of beacon strobe and early acknowledgment indicates a free channel.

9. After receiving the entire data packet, it is processed and acknowledged by the receiver node. With the CC2420 radio module, this time the acknowledgment flag has not been set to in the IEEE 802.15.4 link-layer header. The software acknowledgment is created directly by BEAM, copied to the CC2420 radio TX buffer and then sent without any channel check. A data acknowledgment created by BEAM offers the opportunity to add additional state information to the frame. The detailed meaning of the state information is explained in Section 4.1.3. To save energy, the sender turns the radio module shortly off between sending the data packet and receiving the data acknowledgment.

10. The neighbor node performs two CCA checks to ensure that there is no ongoing traffic. Now, both CCA checks have been negative. Therefore, the node goes back to sleep until the next wake-up period.

BEAM with Beacon Strobes Including Payload

BEAM with beacon strobes including payload, piggy-back the data part to the beacon strobe. A sender with a pending packet periodically transmits beacon strobes including payload until an acknowledgment has been received. Only two successful transmissions are required to forward and acknowledge a data packet. This reduces the complexity of the protocol. If a receiver gets a corrupted data payload, it can wait for the next beacon frame including the same data payload. A drawback of this approach is the higher energy consumption at non-involved receivers. In case of no detected traffic, both versions require exactly the same amount of energy. Figure 4.5 depicts how BEAM is using beacon strobes including the payload. The numbering of the following list refers to the numbers depicted in Figure 4.5.

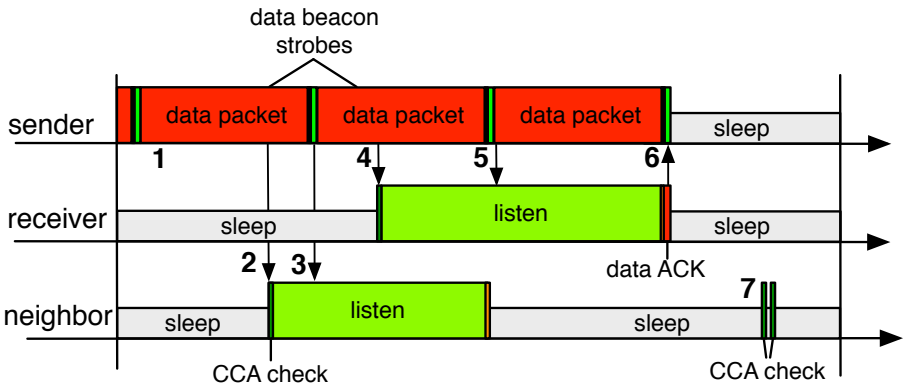


Figure 4.5: BEAM with beacon strobes including the payload.

1. A sender with a pending packet repeatedly transmits beacon strobes including payload. These beacon strobes are IEEE 802.15.4 conform frames. There is no difference compared to the data packet sent by BEAM with short beacon strobes.

2. A neighbor node wakes up and detects an ongoing transmission. Therefore, the node waits for the next synchronization header.

3. Since the target address of the next beacon strobe belongs to another node, no acknowledgment has to be sent. After receiving the data packet, the radio module is turned off for the rest of the wake-up interval. Especially when sending long packets, the neighbor has to listen to the channel for a significantly longer time than with short beacon strobes.

4. The CCA check of a receiver node detects an ongoing transmission. The synchronization header has been missed. The radio module stays in listening mode to receive the next beacon strobe.

5. The radio module of the receiver detects and receives the next beacon frame.

6. The beacon strobe has been transmitted completely and is acknowledged by the receiver node. With the CC2420 radio module, the acknowledgment can be realized by the *AUTOACK* function without any interaction with the microcontroller. After processing the beacon frame, the receiver node turns off the radio module. The transmitter goes back to sleep after the data acknowledgment has been received.

7. In case of no ongoing traffic, a node goes back to sleep immediately after two CCA checks. There is no difference between the both BEAM versions if no frame has been detected.

The performance characteristics of both BEAM versions are compared in Subsection 5.2.3. The next subsection introduces our approaches to optimize the performance of both BEAM versions.

4.1.3 BEAM Optimizations

The first part of this subsection introduces different approaches to optimize the performance of BEAM. This includes one of our main contributions, namely the **buffer index**, which is used to determine the optimal length of the next sleeping period. The second part shows how BEAM integrates the time and energy efficient link layer functions offered by the CC2420 radio module. The use of the *AUTOACK* and the CCA function enables an energy efficient operation during time periods with low traffic load.

Adaptive Duty Cycles

BEAM supports asynchronous adaptive duty cycles. The adaptivity is used to change the duration of the sleeping period to the needs matching the current traffic load. During idle periods with low traffic load, a predefined **default duty cycle period** is applied. The default duty cycle period is the longest sleeping period performed by BEAM. A drawback of long sleeping periods is the long delivery time required to forward a single packet throughout the WSN. The duration of the default duty cycle period can be configured according to the application requirements of the WSN. If an environmental monitoring application requires a short response time, then the default duty cycle period has to be rather short. Otherwise, the default duty cycle period can be extended according to the required energy consumption.

Subsection 5.2.4 evaluates different default duty cycle period for BEAM. XMAC and ContikiMAC are using a default duty cycle period of 125 ms.

BEAM uses four different predefined adaptive duty cycle periods. The corresponding duty cycle period is selected according to the expected traffic load. Subsection 4.1.3 shows how the expected traffic load is calculated. An appropriately selected duty cycle period only consumes as few energy as required to provide sufficient bandwidth for the current traffic load. Increasing the bandwidth during periods with high interferences also saves energy by improving the reliability performance due to less retransmissions and lower packet loss. Too low bandwidth may cause packet buffer overflow. Packets dropped by a packet buffer overflow require an energy costly end-to-end retransmission. Moreover, shorter duty cycle periods reduce the time duration between to wake-up periods. This reduces the number of beacon strobes required to be detected by the receiver node. A lower number of required beacon strobes decreases inter-flow and intra-flow interferences and resulting packet loss of simultaneously forwarded packets.

Figure 4.6 explains the advantage of an appropriately selected duty cycle period with the help of an example. The nodes labeled with C (C-nodes) are able to detect

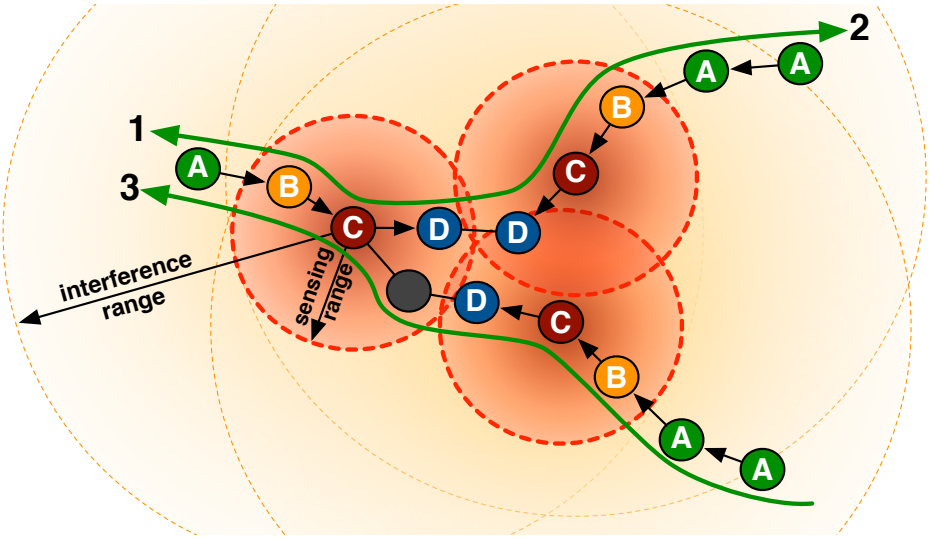


Figure 4.6: Congestion in a WSN.

transmissions of other nodes placed inside the **sensing range** area. The sensitivity of the radio module and the used transmission power determine the sensing range. The smaller of the dashed circles around the C-nodes depicts the sensing range of the individual nodes. Additionally, this circle depicts the range in which a C-node is able to forward a packet. The larger dashed circle depicts the **interference range** of a C-node. Nodes within an interference range experience bit errors in the packet reception. Nodes labeled with A are able to forward their packets to the next hop. Less than 5% of these transmissions fail and, therefore, require hop-to-hop retransmission. The nodes labeled with B are able to forward the received packets to C-nodes. Some of these transmissions have to be retransmitted due to bit errors. C-nodes have serious problems to forward their packets towards nodes labeled with D. C-nodes receive significantly more packets than they are able to forward. Unfortunately, they are not able to detect each other's transmissions. This results in a hidden node problem, causing high internal interferences, respectively a too low SNR at the nodes labeled with D. If congestion cannot be dissolved, then C-nodes have to drop packets from their local packet buffers.

We identified two network conditions provoking congestion. First, bidirectional traffic on a single network path, as represented by line 1 and line 2 in Figure 4.6, may cause intra-flow interferences. Second, packets concurrently forwarded on different nearby located network paths, represented by line 1 and line 3, may cause inter-flow interferences and hidden node problems. Both network conditions tend to cause congestion and packet loss at higher traffic loads.

Protocols such as BEAM, which require link layer acknowledgments, may increase the congestion problem caused by intra-flow and inter-flow interferences. In some cases, a node is able to receive most of the data packets, but the acknowledg-

ments fail. The missing acknowledgments trigger additional non-required retransmissions. This further increases interferences and prevent the sender node from processing the next frame.

By reducing the duty cycle period, bandwidth is increased and the delivery time is reduced. A faster delivery time can reduce the number of simultaneously forwarded packets in the network. If only one packet is forwarded, no internal interferences appear. Internal interferences significantly increase bit error rates. The resulting packet loss further triggers additional retransmissions to recover lost packets. Triggered retransmissions generate additional traffic and interferences. Rapidly increasing interferences and resulting retransmissions may cause congestion. Incoming packets during congestion have to be buffered by the node. The packet count that can be stored by a node is limited. Therefore, packets usually have to be dropped if the congestion persists for longer period of time.

Handling simultaneously forwarded packets is a **major challenge for reliable radio duty cycle protocols**. Our tests with the OMNeT++ network simulator confirm that congestion and packet loss are basically caused by several packets forwarded during the same time period in the same interference range. The resulting congestion and blocked traffic are hard to dissolve and may cause significant packet loss. To avoid congestion and to simultaneously preserve energy, an appropriate duty cycle period has to be applied to the individual nodes. To apply an appropriate duty cycle period, the current traffic load has to be measured.

The next subsection discusses how to determine the current traffic load to apply an appropriate duty cycle period.

Buffer Index

Adaptive duty cycle protocols adapt the duty cycle period according to the current traffic load. Therefore, adaptive duty cycle protocols require a traffic load measuring mechanism to determine the current traffic load. Protocols such as XMAC and MaxMAC use **traffic monitoring**, which counts the recently forwarded packets to determine the current traffic flow. These mechanisms work quite well in simple network topologies with low traffic rates where almost no internal interferences can occur. Network topologies including multiple connections and simultaneously forwarded packets cannot be handled by these simplistic mechanisms. Resulting internal interferences may provoke congestion, which then considerably delay the forwarded traffic flow. Counting recently forwarded packets shows the same results for low traffic load caused by less generated packets as well as for congestion caused by a too high traffic load. Therefore, traffic monitoring mechanism will increase the duty cycle period in both cases. Longer applied duty cycle periods in case of congestion will increase resulting packet loss even more. Congestion requires short duty cycle periods to enhance offered network bandwidth for high traffic load. Therefore, traffic monitoring is not suitable to avoid congestion caused by internal interferences. With traffic monitoring, packets forwarded during the same time period are dropped to reduce the internal interferences.

BEAM uses forward-looking **traffic prediction** to determine an appropriate duty cycle period. This mechanism is based on a two-bit **buffer index** representing the number of pending packets. The buffer index is added to every beacon strobe to inform a receiving node about pending packets for the addressed node. The buffer index is also copied to the software acknowledgment used by the BEAM version with short beacon strobes. The hardware acknowledgments used by the other BEAM version cannot be modified. The buffer index in the software acknowledgment informs a sender about total pending packets in the buffer of the receiver node. To not waste the limited IEEE 802.15.4 MAC frame payload, the buffer index is written inside two of the five reserved bits of the IEEE 802.15.4 *Frame Control Field*.

Table 4.1 shows the mapping between the value of the buffer index and the amount of pending packets. The column *generic* shows a mapping, which is independent of the used packet buffer capacity. The column *implemented* shows the mapping used in our implementation with a packet buffer capacity of six packets. A packet buffer capacity of six packets is also the default capacity used by Contiki.

Transmitted buffer index	Pending Packets (*generic*)	Pending Packets (*implemented for a buffer capacity of 6 packets*)
0	0	0
1	1	1
2	2 - 50% of the buffer capacity	2-3
3	>50% of the buffer capacity	4-6

Table 4.1: Buffer index selected according to the amount of pending packets.

A node stores the received buffer index with the current timestamp and corresponding sender address in its **neighbor table**. This table holds all relevant neighbor node information received by beacon strobes, data frames or acknowledgments. For every neighbor node an individual buffer index entry is maintained in the neighbor table. The use of the buffer index depends on the addressing of the received beacon strobes. If the beacon strobe was addressed to this node then the buffer index is stored as *pending* buffer index to the neighbor table. Every time the radio module is turned off, the pending buffer indices are accumulated to calculate the expected traffic and resulting duty cycle period. In case of an implicit acknowledgment or if a packet has been received by overhearing, the buffer index is stored in the neighbor table as *fuzzy* buffer index. The *fuzzy* buffer index indicates the expected traffic load between neighbor nodes, while the *pending* buffer index is announcing incoming traffic for this node. Therefore, the *fuzzy* buffer index value is not used by BEAM to calculate the expected incoming traffic load. It is only offered to the reliability layer to estimate the expected overall local traffic load and resulting inter-flow interferences.

The use of the pending buffer index enables a sender to announce multiple pending packets with a single beacon strobe to a receiver. The accumulated received pending buffer indices determine the length of the next duty cycle period. Every time after the radio has been turned off, the next wake-up time point is calculated. This mechanism enables a node to directly switch from the longest to the shortest duty cycle period by receiving a single beacon strobe announcing abruptly increasing traffic load. Table 4.2 shows two possible mappings between the accumulated pending buffer indices and the applied duty cycle period. The values in column *wake-ups per sec. (implemented)* represents the mapping used by our BEAM implementation for comparison with XMAC and ContikiMAC. This mapping uses, like XMAC and ContikiMAC, a default duty cycle period of 125 ms. resulting in eight wake-up periods per second. Column *wake-ups per sec. (low energy profile)* shows an alternative mapping for a more energy preserving configuration. A drawback of the energy preserving mapping is the longer delivery times at low data rates. A detailed evaluation for a maximum and a minimum value for the wake-ups per second is shown in the evaluation part in Subsection 5.2.4.

Traffic load	Accumulated pending buffer indices	Wake-ups per sec. (*implemented*)	Wake-ups per sec. (*low energy profile*)
Low	0	8	1
Moderate	1	16	8
High	2 - 4	64	64
Maximum	>4	256	256

Table 4.2: Two mappings of the pending buffer indices to the selected wake-up frequency.

Optimizing Beacon Strobe Transmissions

With a LPL protocol, the exact timing of the next wake-up of a receiver is usually unknown to a sender. Therefore, BEAM starts transmitting the beacon strobes as soon as a new packet has been received from the upper layer. The longer the sleeping period of the receiver is, the more beacon strobe transmissions are required on average. Optimization of the beacon strobe transmission, presented in this subsection, tries to calculate the next wake-up of a receiver. This enables BEAM to delay the start of the beacon strobe transmissions until the receiver wakes up. Shorter beacon strobe transmission periods reduce the interferences and preserve energy.

A precondition for reliable wake-up calculations are consistent time lengths between the individual wake-up periods, i.e. each of the four predefined duty cycle periods shown in Table 4.2 has to be exactly as long as defined in the table. Frame or noise detection during the wake-up phase must not affect the total length between two wake-up periods (one duty cycle period). This means if currently eight wake-up periods per second are applied, then every 125ms a new wake-up period

has to be performed. Therefore, the time taken for the wake-up period is subtracted from the total duty cycle period to get a matching sleeping period length. Sometimes, the duration of a wake-up period is longer than the predefined duty cycle period. In this case a multiple of the current duty cycle period is used to get a suitable sleeping period duration. This results in **consistent channel check periods**, where a node is able to receive a frame.

Next to the consistent channel check periods, a two-bit **duty cycle index** is added to the IEEE 802.15.4 *Frame Control* field. The duty cycle index informs the neighbor node, which of the four available duty cycle periods is currently used by the node. It is added to every beacon strobe and software acknowledgment. A received duty cycle index is stored in the neighbor table. The duty cycle index, in combination with the timestamp and neighbor node address, provides an estimation of the next wake-up time for a receiver. The transmission delay is not applied if the next wake-up time is estimated within the next 8 ms. If no acknowledgment has been received, beacon strobes are sent for an entire duty cycle period of the receiver. If still no acknowledgment has been received, then the beacon strobes are sent for the duration of a default duty cycle period.

Figure 4.7 shows a delayed transmission for BEAM with short beacon strobes. The upper layer respectively the reliability layer H2HR forwarded a packet to BEAM at (1). BEAM checks if there is a valid entry for the receiver in the neighbor table. According to the last timestamp and duty cycle index the next wake-up is expected at (3). BEAM delays the beacon strobe burst until this point in time (2). BEAM using beacon strobes including the payload works exactly the same. The transmission delay mechanism introduced in this subsection is analyzed in Subsection 5.2.2.

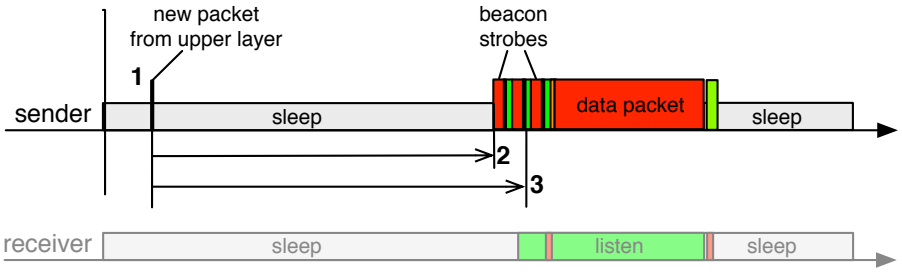


Figure 4.7: Transmission delay for beacon strobes.

Reducing Energy Consumption during Low Traffic

During periods with low or no traffic load, hundreds of wake-up periods may be performed between two forwarded packets. During these idle periods, all energy consumed by the radio module is used to detect a potential beacon strobe transmission. To preserve energy, the required length of the listen interval to check the

channel, has to be as short as possible.

The CC2420 radio module offers different channel checking techniques. Figure 4.8 shows the energy required by three different channel checking techniques, which could be used by BEAM. The shown measurements are generated by the RIGOL multimeter with 50'000 measurement points per second. They are intended to get a basic impression about the performance of the different mechanisms. A detailed evaluation of the channel check techniques is performed in Subsection 5.1.4. Most energy is required by using the FIFOP pin interrupting the microcontroller to detect a transmission. This technique is for example used by XMAC. Using the *AUTOACK* and CCA function offered by the CC2420 radio module enable clearly shorter channel checks. The curve for the continuous CCA check shows the shortest execution time. During the used time period, three individual CCA checks are performed in a row. This is sufficient to detect any ongoing BEAM beacon strobe transmission acknowledged by the *AUTOACK* function. The curve with the two short peaks shows the energy profile if the radio module is turned off between the single CCA channel checks. Here the radio module is turned on twice for the minimum time to achieve a valid value of the CCA pin. The applied time period between the two individual CCA checks is shorter than the transmission time required to send the shortest possible beacon strobe. Otherwise, a beacon strobe could be missed. As a result BEAM will use the *AUTOACK* function and two short CCA checks to minimize the energy consumption during idle periods.

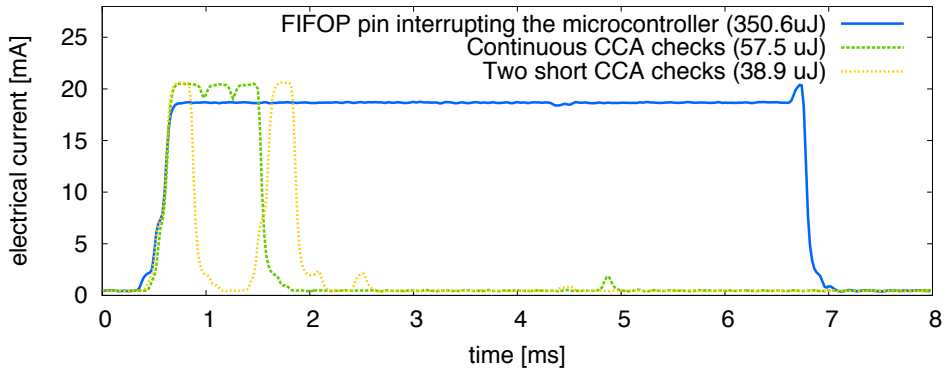


Figure 4.8: Energy requirement of different with beacon strobes including the payload.

CRC Calculation

The *AUTOACK* function of the CC2420 radio module requires an IEEE 802.15.4 conform CRC to validate incoming frames. To relieve the microcontroller from calculating and verifying a compliant CRC, the *AUTOCRC* function of the CC2420 can be used. The *AUTOCRC* function calculates the CRC on the radio module and transmits it after the last data byte of the TX buffer. On the receiver side, the CRC

is calculated and checked for validity. Calculating the CRC by the radio module reduces the code size, preserves energy and reduces the execution time.

4.1.4 BEAM Reliability Support

Introducing BEAM reliability support requires a short introduction into the cooperation between BEAM and the overlaying hop-to-hop reliability protocol. The reliability protocol is responsible for handling the packet queue and additionally delaying the transmissions if required. Transmission delays are used to prevent and handle congestion during time periods with too high traffic load or internal interferences.

Forwarding a packet with BEAM and the overlaying hop-to-hop reliability protocol works as follows. First, the reliability layer determines which packet at which time has to be forwarded next. This packet is delegated to BEAM including a sequence number for reference purposes. Then, BEAM tries to transmit the packet to the next hop. After processing the packet, BEAM sends a **transmission report** including the sequence number back to the reliability protocol. The transmission report provides additional information necessary to calculate an appropriate transmission delay if required. BEAM defines four different transmission reports:

1. **Transmission successful:** The acknowledgment for the data frame has been received. The data packet has been successfully transmitted. Additionally, BEAM offers the measured RSSI and LQI values offered by the CC2420 radio module to the reliability protocol. This information can be used for routing decisions.

2. **Missing Acknowledgment:** The beacon frame strobe has been sent. But no acknowledgment or other packets have been received during the beacon frame strobe. This may indicate that there is currently a hidden node problem or that the node has disappeared.

3. **Interference detected:** The beacon frame strobe has been interrupted due to interferences. Some other nodes are transmitting data.

4. **Channel busy:** The channel has been busy. There was no possibility to start the beacon strobes.

BEAM also provides access to its **neighbor table**. This offers the reliability protocol additional information about the current network condition at the neighbor nodes. Table 4.3 shows the values provided in the neighbor table.

The introduced BEAM transmission report and BEAM neighbor table provide detailed information about the current network condition in the local area. Our hop-to-hop reliability protocol uses the information to detect and dissolve congestion.

Value	Description/Purpose
Timestamp	Time of the last successful reception
Address	Address of the neighbor node
Duty cycle index	Currently used duty cycle period
Pending buffer index	Expected packets from this neighbor
Fuzzy buffer index	Pending packets of this neighbor

Table 4.3: BEAM neighbor table

4.1.5 BEAM FEC Support

FEC codes are a common approach to enhance reliability of wireless transmissions. The intention of a FEC code is basically to improve the probability of a successful transmission by adding redundant information. They do not include a retransmission mechanism and, therefore, do not save energy. Packet loss that is caused by corrupted a physical frame start cannot be detected by FEC mechanisms. Therefore, retransmissions have to be performed by an additional ARQ mechanism. Moreover, energy efficiency has to be provided by a mechanism such as radio duty cycling. Although FEC codes are able to reduce the number of required retransmissions, they do not directly preserve energy by default. Instead of retransmitting a lost frame, the radio module is switched into listening mode. Unfortunately, this does not save energy, since the CC2420 radio module consumes less energy in transmission than in listening mode. The radio module has to be turned off to save energy. Nevertheless, FEC codes are known to be able to enhance the energy efficiency and reliability performance of a network stack.

We performed a detailed evaluation of the general potential of two different FEC codes. One FEC code features a high recovery capability with high energy and time requirements. The other FEC code has low energy and time requirements, but also a lower recovery potential. We measured the energy consumption and reliability performance under different network conditions. Additionally, we analyzed the impact of the physical channel to FEC codes by using a packet-oriented and a byte-oriented radio module. The evaluation and resulting potential of both FEC codes are presented in Section 5.4. To verify the results about the general potential of the FEC codes, we included both FEC codes into BEAM. The rest of this subsection describes the requirements and resulting design decisions to add the FEC codes to a radio duty cycle protocol using an IEEE 802.15.4 compliant radio module.

The additional energy and time requirements of FEC codes imply a serious challenge for energy efficient radio duty cycle protocols. Especially the decoding of an erroneous packet depicts a problem. FEC decoding cannot be handled by the radio module, which makes it impossible to use the *AUTOACK* function. A FEC encoded frame has to be encoded by the microcontroller. This delays frame acknowledgment, leading to longer silence periods between the individual beacon

strobes. The decoding delay additionally increases the acknowledgment delay and the resulting acknowledgment delay increases the energy required to perform the periodical channel check.

Furthermore, a valid IEEE 802.15.4 frame header is required to make use of the *AUTOACK* function. Therefore, the link layer header cannot be encrypted by a FEC code, when using a CC2420 radio module. Only the link layer payload can be encoded. These restrictions result in the following protocol design:

- **Use short beacon strobes:** Short beacon strobes do not contain any payload that can be encrypted. This enables a receiver to make use of the *AUTOACK* function to acknowledge a beacon strobe. The resulting silence period between the individual beacon strobes is as short as possible, which enables an energy efficient wake-up periods. In case of erroneous beacon strobe, the next beacon can be used to receive the payload.
- **Encryption of the payload:** The encryption of the payload is done by the reliability protocol handling the local packet buffer and retransmissions. By buffering the encrypted packet for retransmissions, the packet has to be encoded only once.
- **Software acknowledgement for data frames.** The receiver only has to recover a packet if the CRC check of the radio module indicates a bit error. The maximum acknowledgement waiting time of the sender depends on the transmitted payload length. This is due to the fact that the time required by the receiver to read the frame from the RX buffer and to recover potential bit error depends on the received frame length.
- **Static and adaptive versions:** We implemented two versions of the Hamming(12,8) and Reed-Solomon(255, 225) FEC codes introduced in Section 2.5.4. The **static** FEC code versions encode every transmitted packet regardless of the bit error rate of a link. The **adaptive** FEC code versions announce in the data acknowledgment if the packet was received with bit errors. The sender encodes a packet if either the data acknowledgment for the last data frame was missing or if the receiver announced bit errors in the received packet. After receiving a data acknowledgment announcing error free transmission, FEC encoding is disabled for the next data transmission.

Section 5.4 presents a detailed evaluation of FEC codes concerning energy efficiency and reliability performance in WSNs. This includes the evaluation of BEAM supporting FEC introduced in this section.

4.1.6 BEAM Summary

BEAM is an energy efficient link layer protocol to maximize the lifetime of a WSN. It employs adaptive and unsynchronized radio duty cycle periods to minimize energy usage of the attached radio module. BEAM uses a traffic prediction mechanism for adapting the duty cycle periods to the current traffic load. BEAM employs

all advanced features of the energy efficient IEEE 802.15.4 compliant radio module CC2420 to optimize the energy efficiency. Supporting the IEEE 802.15.4 standard additionally enables a direct communication between different heterogeneous sensor node platforms. The performance of BEAM is evaluated in Chapter 5.

4.2 Hop-to-Hop Reliability Protocol

The purpose of the hop-to-hop reliability protocol (H2HR) [3, 97] is to ensure reliable hop-to-hop forwarding. The design of H2HR has its source on the evaluation results of different end-to-end reliability protocol mechanisms. We analyzed different existing reliability mechanism for WSNs with the OMNeT++ network simulator [4]. We implemented the different mechanisms introduced by the authors of [22] and [12]. These mechanisms are located on a layer between the IP and the TCP layer to handle caching and forwarding of TCP/IP packets. One of these optimization mechanisms makes use of acknowledgment information offered by the link layer. In case of failed forwarding to the next hop, a cached copy of the lost TCP frame is immediately injected into the network layer to be forwarded again. This mechanism has been the most efficient mechanism evaluated by far. Unfortunately, the caching mechanism requires that all frames of a single TCP connection are sent over the same route through the WSN. Furthermore, they require an adaptation of IP forwarding rules to send every incoming TCP/IP frame to the transport layer instead of routing them directly. This would violate the layer separation in our network stack, where transport protocols are only involved at the endpoints of a TCP connection.

Therefore, we shifted the additional reliability functions in our network stack to the link layer and created H2HR. H2HR insistently tries to retransmit every unsuccessfully transmitted packet to avoid end-to-end retransmissions. This enables reliable hop-to-hop forwarding for all unicast based transport protocols. We expect lower energy consumption by excessive local retransmissions than for end-to-end retransmissions. A main challenge for H2HR is to avoid packet loss caused by congestion. H2HR is able to detect critical network conditions as well as adapting traffic flow by controlling the particular point in time of every transmission. This enables H2HR to avoid or dissolve congestion.

Subsection 4.2.1 describes the packet buffer queue for storing packets in H2HR. Subsection 4.2.2 then introduces the congestion detection and control mechanism to calculate an optimal transmission time. Subsection 4.2.3 explains the H2HR backpressure mechanism. Subsection 4.2.4 describes the different processes of H2H when forwarding a data frame. Finally, Subsection 4.2.5 introduces the packet aggregation mechanism used by H2HR and BEAM to minimize the amount of forwarded packets.

4.2.1 Packet Queue

A basic requirement for a hop-to-hop reliability layer is to be able to buffer packets in a packet queue. All packets received from an upper layer have to be stored in the packet queue for possible retransmissions. Incoming packets from the lower layer can be forwarded to the upper layer without the need of being processed. To enhance the effectiveness of the packet queue, we add some status information to the slots of the packet queue.

- **Ready to send**. This packet is ready to be sent. There was no transmission of this packet yet.
- **Transmission in process**. The packet in this slot is currently processed. Either BEAM is trying to send the packet right now or the H2HR packet handler is delaying the transmission.
- **Successfully transmitted**. The packet has been successfully forwarded to the next node. The queue slot can be overwritten. This is also used as default state after the system has powered on.
- **Transmission failed.** It was impossible to forward the packet to the next hop. The retransmission limit for this packet has been reached.

H2HR does neither delete a packet after successful forwarding nor after reaching the retransmission limit. Slots are just marked with the corresponding state and timestamp. New packets received from the upper layer are copied to a slot labeled with *successfully transmitted* if available. Otherwise, they are copied to slots marked with *transmission failed*. If none of these slots is available, the packet is dropped. Most of the packet loss, which occurred in our experiments, was caused by overflow of the local packet buffer during congestion. BEAM already tries to minimize congestion by offering sufficient bandwidth during critical periods, but sometimes too many packets are forwarded within the WSN. The resulting congestion disrupts traffic flow and results in packet buffer overflow. The next subsection describes the H2HR **congestion detection and control** mechanism to detect and prevent congestion by reducing the traffic flow and internal interferences.

4.2.2 Congestion Detection and Control Mechanism

The goal of the H2HR congestion detection and control mechanism is to adapt the traffic flow before it is disrupted by congestion. Most congestion is either caused by too much interference in a challenging network topology or by too many packets generated and injected into the network. The adaptive radio duty cycle protocol BEAM already adapts the bandwidth concerning the expected traffic flow to avoid traffic bottlenecks. H2HR slows down the traffic flow if the offered bandwidth by BEAM is not sufficient. Reducing the traffic flow also reduces the inter-flow interferences in network topologies with multiple traffic flows. This enhances the

probability of blocked nodes to forward their buffered packets. In case of too many generated packets, packets have to be dropped as soon as possible in order not to overload the WSN. The most energy efficient procedure is to drop them already on the generating node, i.e. before they are transmitted for the first time. Controlling the delays of the individual packet transmissions causes a hop-to-hop backpressure mechanism.

When detecting upcoming congestion, H2HR reduces the data rate by calculating an optimal transmission delay. H2HR uses **three** different information sources to enable an accurate congestion detection and determination of an appropriate transmission delay:

1. **Transmission report of BEAM:** The transmission report provides information about the **channel load**. The message *channel busy* as well as *interference detected* shows detected nearby transmissions. *Missing acknowledgments* indicate a hidden node problem.

2. **Neighbor table of BEAM:** The *neighbor table* provides information about the **receiver load** such as cached packets and current duty cycle period (see Table 4.3). This enables the detection of upcoming congestion. The more traffic load a receiver node currently has to handle, the longer is the used transmission delay to that node.

3. **Retransmission count of a packet:** For each retransmission attempt of an individual packet the transmission delay is increased.

An optimal transmission timing of the individual packets provides an optimal traffic flow. In case of low traffic load, forwarding of a packet can be executed rapidly. Moreover, a lost packet can be instantly retransmitted. With increasing traffic load, especially the retransmissions of lost packets have to be timed carefully. Otherwise, too much interference may overload the radio channel. The next subsections describes the use of the information of the three different available information sources in more detail.

1. Source: Transmission Report of BEAM

The transmission report enables the estimation of the current **channel load**. The higher the channel load is, the higher is the probability of internal interferences resulting in bit errors and retransmissions. Unfortunately, only interferences caused by transmissions from nearby nodes can be directly detected by the radio module. The radio module cannot directly detect interferences caused by distant nodes. H2HR uses the following transmission reports of BEAM to detect interferences by distant nodes:

- *Transmission successful* shows that no interferences are detected. In this case, no delay is required for sending additionally cached packets.

- *Missing acknowledgment* indicates existing interferences outside the sensing range. A hidden node problem probably prevents the node from a successful transmission. Adding longer delays can solve the hidden node problem by providing the blocked nodes sufficient time to successively send their packets. To solve packet collisions caused by a hidden node problem, the traffic load has to be significantly reduced.

- *Interference detected* is caused by detecting another transmission than the expected acknowledgment. This can be caused for example by an acknowledgment destined for another node, a simultaneous transmission outside the sensing range or external interferences.

- *Channel busy* is reported if every of the 10 performed channel checks showed a busy channel. It was not possible to send one single beacon strobe. This points to a nearby transmitter sending its own beacon strobes.

A single transmission report only indicates a problem with the channel load. For example, the transmission reports *Missing acknowledgment* and *Interference detected* are basically caused by external interferences during very low traffic periods. Therefore, the frequency and type of the transmission report has to be taken into account for calculating the appropriate transmission delay. Considering the frequency and type of the transmission report, we define a so-called **channel load factor**. The range of the current channel load factor, shown in Table 4.4, is defined between 1 and 20. A calculated channel load factor higher than 20 is always reduced to 20. A range with a limit higher than 20 would result in a very long delay, longer than the maximum delay we apply. Every *transmission successful* reduces this counter by a value of 2. For every *missing acknowledgment* report, we add a value of 2, for every *interference detected* and *channel busy* report we add 1. For example, a current channel load factor of 3 for specific node is increased to 5 by a *missing acknowledgment* report. Besides the channel load, the receiver load and already executed retransmissions have to be taken into account to determine an appropriate transmission delay. The following subsections show, how they are considered to calculate the actual transmission delay.

Transmission report	Channel load factor increase/decrease
Transmission successful	-2
Missing acknowledgment	$+2$
Interference detected	$+1$
Channel busy	$+1$

Table 4.4: Channel load factor.

2. Source: Neighbor Table of BEAM

The information available from the BEAM neighbor table (see Table 4.3) enables conclusions about the current **receiver load** of the one-hop neighbor nodes. H2HR uses this neighbor table information of the receiver node to calculate an appropriate transmission time. The two buffer indices indicate pending packets, while the duty cycle index shows the current duty cycle period. If the buffer indices are not available or zero, then no additional delay is used. Otherwise, H2HR delays the next transmission for an appropriate time period as shown in Table 4.5. The values shown in the table are determined by evaluations with the OMNeT++ network simulator and the WISEBED WSN real world testbed (see Section 5.3). If the duty cycle index is not available, then the default duty cycle period is used. Sometimes, congestion can prevent the traffic required for updating the neighbor table. This may lead to out-of-date neighbor tables, i.e. the update lags behind reality.

Buffer index of the target node	Transmission delay
0	no delay
1	1 - 3 duty cycle periods
2	2 - 6 duty cycle periods
3	4 - 12 duty cycle periods

Table 4.5: Transmission delay by buffer index.

3. Source: Retransmission Count of a Packet

For every retransmission attempt of an individual packet, an additional delay is added. Table 4.6 shows the used relation between retransmission attempts and delay. The values were determined by evaluations with the OMNeT++ network simulator and the WISEBED WSN real world testbed (see Section 5.3). If the retransmission limit has been reached, the neighbor node has some reception problems. In case of a high traffic load, the corresponding node is usually overloaded. Otherwise, the neighbor could be moved away or run out of energy.

Retransmission attempts	Transmission delay
1	1 - 3 duty cycle periods
3, 4	2 - 6 duty cycle periods
4, 5	4 - 12 duty cycle periods
6-10	8 - 24 duty cycle periods

Table 4.6: Transmission delay by retransmission count.

Total Transmission Delay

Information source	Information	Unit	Symbol
Transmission report	Channel load	Factor	f_{CL}
Neighbor table	Receiver load	Number	dp_{RL}
Retransmission count	Retransmission count	Number	dp_{RC}
Neighbor table	Duty cycle index	Time	$Node_{DL}$

Table 4.7: Information sources.

Table 4.7 shows the different information sources used by the congestion detection and control mechanism. All information sources are taken into account to determine an appropriate duration for the next transmission delay (TX_{delay}). The applied transmission delay time is calculated by:

$$TX_{delay} = Node_{DL} \cdot (f_{CL} \cdot (dp_{RC} + dp_{RC}))$$

The maximum applied delay is limited to 1000 ms. An evaluation of the different information sources is shown in Section 5.3.2.

4.2.3 H2HR backpressure mechanism

Some WSN protocols are using hop-to-hop based backpressure techniques to avoid congestion (see Section 2.6.2). If too many sensor nodes continuously generate and forward packets towards the sink, then the resulting traffic load can be too high to be forwarded in the WSN. The resulting interferences cause congestion and packet loss. The H2HR retransmission delay mechanism works as a hop-to-hop based backpressure mechanism. In the best case, the retransmission delay mechanism is able to delay the transmission attempts on the entire path up to the nodes generating the packets. This reduces the amount of packets injected into the WSN.

4.2.4 Forwarding a Data Frame

This section explains how H2HR handles packet forwarding with the help of an example. H2HR tries to ensure the successful forwarding of every frame to the next node. With H2HR, frames are only dropped due to packet buffer overflow or if the receiver node seems to be disappeared. This mechanism minimizes the amount of required end-to-end retransmissions. End-to-end retransmissions cause a significant higher energy consumption and generate more intra-flow interferences than extensive local retransmissions. Figure 4.9 shows how a packet is forwarded by H2HR. The H2HR packet handler executes the required actions and controls the packet queue. The packet handler processes one packet at a time. New packets received from the upper layer are copied to the packet queue (1). We extended the packet queue implementation of Contiki with the status information introduced

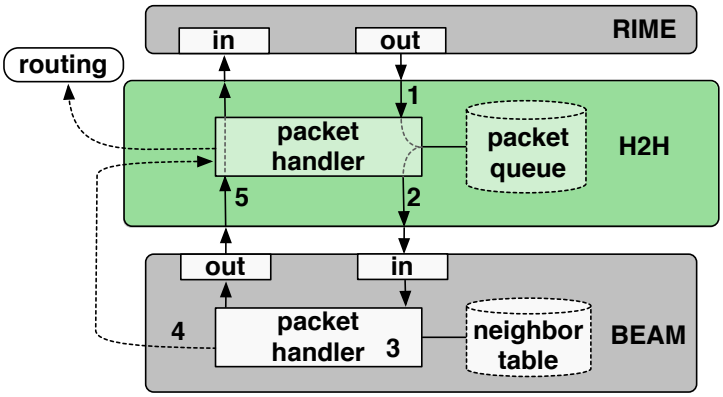


Figure 4.9: H2HR protocol.

in Section 4.2.1. The packet handler calculates the transmissions delays and delegates the packet at the corresponding time to BEAM (2). BEAM immediately tries to send the packet (3). After having finished a transmission attempt, the corresponding transmission report is sent back to the H2HR packet handler (4). Packets received from other nodes are directly forwarded from BEAM to upper layer (5).

4.2.5 Packet Aggregation

A way to reduce internal interferences is to reduce the number of simultaneously forwarded packets within the network by aggregation. Figure 4.10 shows the protocol header for handling aggregated packets. H2HR adds to every aggregated data frame the corresponding length and sequence number values. The sequence number is used by the reliability layer to detect packet duplication. Aggregated packets are announced to the receiver by a set **multiple frames** bit. When using the CC2420 radio, the multiple frames can be set as one of the five reserved bits of the IEEE 802.15.4 frame. Including the two bits of the buffer index and the two

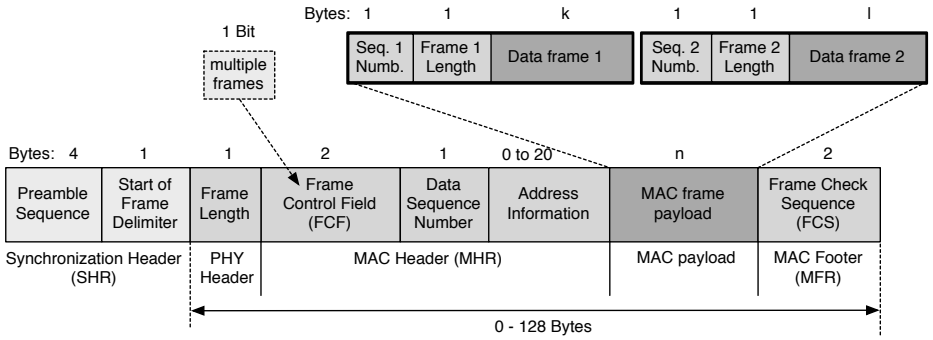


Figure 4.10: Packet Aggregation Format.

bits of the duty cycle index, all five reserved bits are used. If future revisions of IEEE 802.15.4 prevent us from using five reserved bits, then BEAM can introduce an additional one-byte header inside the IEEE 802.15.4 MAC payload.

When using the CC2420 radio, the packet aggregation mechanism can only be applied to small packets. This is due to the limited MAC payload of the IEEE 802.15.4 frame. When using μIP and UDP, only three packets fit into an aggregated data frame. Subsection 5.2.6 analyzes the impact of the packet aggregation mechanism.

4.3 UDP End-to-End Reliability Protocol

To provide a reliable WSN network stack, an end-to-end reliability mechanism is required. End-to-end reliability mechanisms are able to detect dropped packets on intermediate nodes and to trigger retransmissions from the source node. The TCP support of Contiki enables using the retransmission mechanisms of TCP. Unfortunately, the protocol overhead of TCP has some drawbacks in WSNs. One problem with TCP is the overhead caused by the end-to-end acknowledgments for successfully received data frames. They generate intra-flow interferences and, therefore, require additional energy. The authors of [12] introduced different techniques to optimize the TCP end-to-end acknowledgment mechanism to enhance bandwidth utilization and reduce power consumption in wireless sensor networks, see Section 2.6.2.

For WSNs, we use UDP, which shows only a low protocol overhead. Unfortunately, UDP does not offer an end-to-end reliability mechanism to recover packet loss. Therefore, we added an end-to-end reliability mechanism to UDP called UDP-E2E. It is located and executed on top of UDP, i.e. as an application layer protocol. This enables establishing reliable UDP flows between a sensor node and any server in the Internet. UDP-E2E provides datagram sockets to establish host-to-host communications. An application can bind a socket with a combination of the IP address and the service port of the unicast flow endpoint. The reliability mechanism is handled by UDP-E2E and hidden to the application. The remainder of this section describes the functionality of UDP-E2E.

4.3.1 UDP-E2E Sequence Numbers

The design of UDP-E2E is based on the sequence number mechanism of the transport protocol RMST [86] introduced in Section 2.6.2. UDP-E2E does neither support in-network caching nor repair functions of RMST. RMST has to manage forwarding of a packet on every intermediate node to support in-network caching and repair functions. Using a transport layer protocol, such as RMST, on intermediate nodes would violate the layer separation in our network stack, where the network layer is responsible to handle packet forwarding of a packet. Usually, transport protocols are only involved at the endpoints of a connection. In our stack,

local caching and repair functions of RMST are shifted to the hop-to-hop reliability layer. Moreover, bit error detection is only performed hop-to-hop by IEEE 802.15.4 and not end-to-end by UDP-E2E. UDP-E2E only implements the packet loss detection based on sequence numbers and the end-to-end retransmission mechanism of RMST.

The sequence numbers are used to enable the receiver to detect packet loss. In order to request a retransmission of a lost packet, a negative acknowledgment has to be sent, which includes the sequence number of the missing packet. In case of multiple lost packets, all missing sequence numbers are included in one negative acknowledgment. In contrast to TCP, there is no connection establishment and termination in our transport protocol UDP-E2E.

Figure 4.11 shows how UDP-E2E works. Using UDP-E2E works in the same way such as TCP or UDP, i.e. by using sockets. An application sends the payload to a socket, which is handled by UDP-E2E. The sequence number handler first fragments the received application payload into smaller UPD-E2E frames if necessary (1). The maximum payload for a UDP-E2E frame is 86 bytes. Otherwise, it would not fit into an IEEE 802.15.4 frame in our network stack. The sequence number handler adds a 8-bit sequence number to each UDP-E2E frame. If the application payload has been fragmented into multiple UPD-E2E frames, then the total number of fragmented frames is added to every UDP-E2E frame header. This enables UDP-E2E on the receiver side to reassemble the individually received UDP-E2E frame again to the original application payload. If the application payload fits into one UPD-E2E frame, then the same sequence number is added a second time. The individual frames are sent to a UDP socket (2). A watchdog at the receiver periodically checks for missing or outstanding sequence numbers (3). In case of detected packet loss, the watchdog sends a selective negative acknowledgment back to the sender to trigger retransmission of missing packets (4). Otherwise, the data is reassembled if required and forwarded to the application (5).

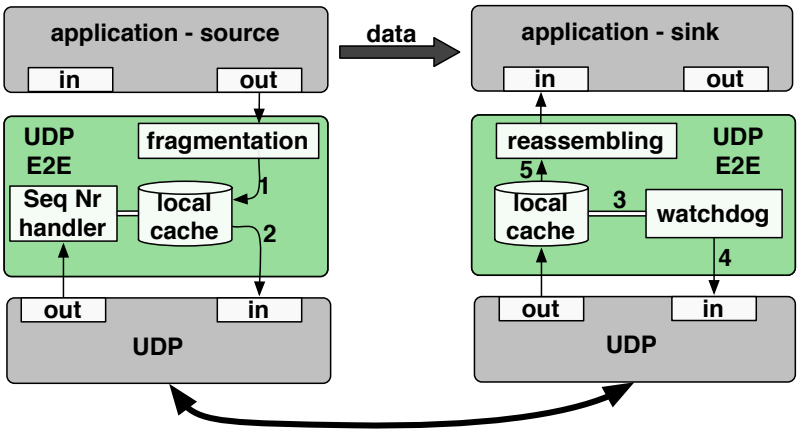


Figure 4.11: UDP-E2E protocol.

4.3.2 UDP-E2E Frames

This subsection gives an overview about the four different frame types defined by UDP-E2E. The first byte of the UDP-E2E header characterizes the frame type. To minimize the header size we use the following mechanism. If the first header byte of the received frame is either 0xFD, 0xFE or 0xFF, then the received frame is not a data frame. Otherwise, the received frame is a data frame and the first byte represents its sequence number. Table 4.8 shows the different frame types and corresponding first byte.

Frame type	Length	First frame byte
Data Frame [Default]	3 - 88	0x00 - 0xFC
Data Frame with Acknowledgment	4 - 88	0xFD
Negative Acknowledgment Frame	2 - 88	0xFE
Positive Acknowledgment Frame	2	0xFF

Table 4.8: Transmission delay by retransmission count.

Default Data Frames

The default data frame with its two header bytes is depicted in Figure 4.12. It is used to encapsulate and transport the application payload over UDP. The first header byte represents the sequence number of the current frame. The second one indicates into how many individual UDP-E2E frames the application payload was fragmented. This enables a receiver to rebuild fragmented application payload and detect missing fragments.

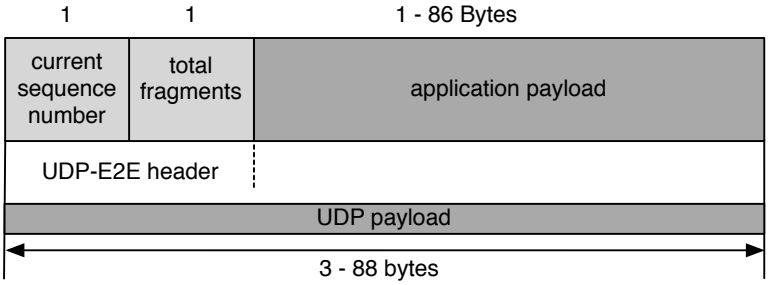


Figure 4.12: UDP-E2E data frame.

Data Frame Triggering Explicit End-to-End Acknowledgment

Negative acknowledgment mechanisms require at least one successfully delivered packet to detected the loss of other packets. Depending on the frequency of received packets it can take a longer time period to detect a packet loss. For exam-

ple, in an event detection scenario each node only sends one short event-notification frame. If this single packet gets lost, the receiver detects the loss not before another succeeding packet has been received. Thus, the sink misses one event. Therefore, we implemented a data frame triggering an explicit end-to-end acknowledgment. The frame format of such a data frame is depicted in Figure 4.13. The *message type* 0xFD indicates that this data frame requires an explicit end-to-end acknowledgment. The *current sequence* number is the sequence number given by the UDP-E2E fragmentation mechanism. The header field *total fragments* announces the total number of UDP-E2E frames into which the application data frame was fragmented.

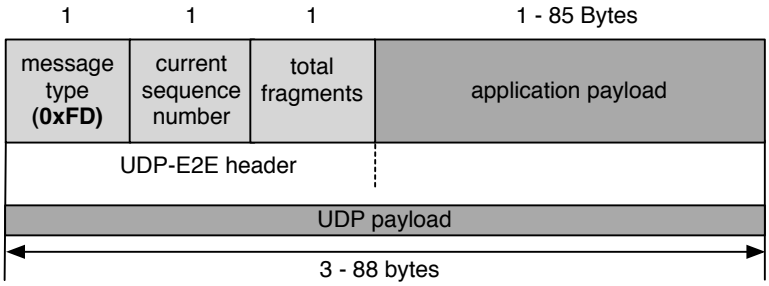


Figure 4.13: UDP-E2E data frame triggering explicit end-to-end acknowledgment.

Negative Acknowledgment Frame

Figure 4.14 shows a negative acknowledgment frame. The first byte is 0xFE and announces a negative acknowledgment. Each following byte then includes a sequence number of a missing packet.

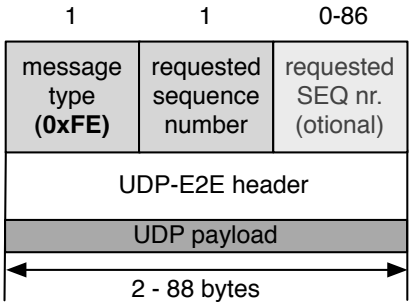


Figure 4.14: UDP-E2E negative acknowledgment frame.

Positive Acknowledgment Frame

Figure 4.15 shows a positive acknowledgment frame. The first byte is 0xFF and announces a positive acknowledgment. The receiver sends a positive acknowledgment if it was requested by the sender.

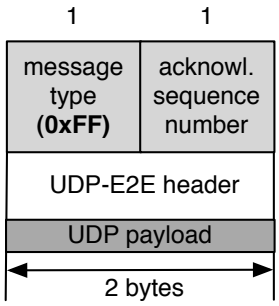


Figure 4.15: UDP-E2E explicit acknowledgment frame.

4.4 Network Stack Overview

Figure 4.16 shows the Contiki network stack including our contributed protocols to support energy efficiency and reliable data flow. The different layers and protocols are described in the following. Our all protocol implementations are fully interoperable with existing Contiki protocols. Therefore, BEAM and H2HR can easily be replaced by any protocol implementations delivered by Contiki without touching the other layers. This enables meaningful comparisons with other protocols.

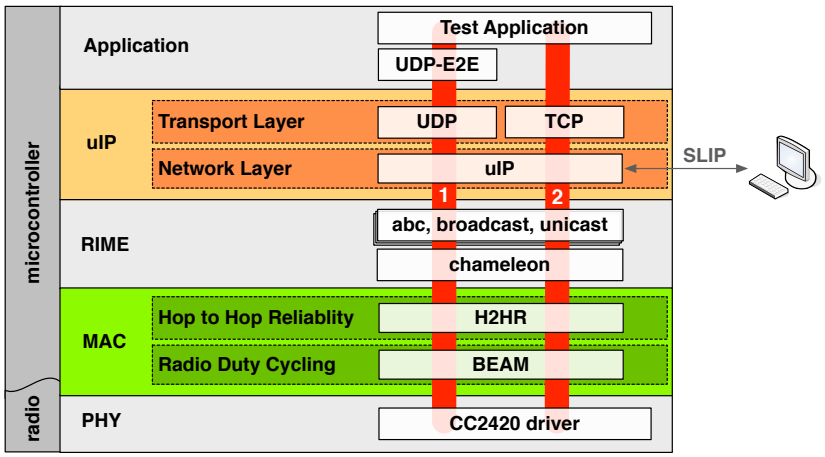


Figure 4.16: WSN network stack layers.

- **Application**: The test application sends and receives packets for the evaluation scenarios. In addition, it logs transmission times and the overall data loss. UDP-E2E is an application-layer overlay protocol that provides reliable end-to-end communication over UDP. It can use negative and optional positive acknowledgments.
- **Transport Layer**: UDP and TCP is part of μIP. Both protocols are RFC compliant to support heterogeneous network connectivity. End-to-end reliability is supported by TCP.
- **Network Layer**: The RFC compliant IP implementation is supporting heterogeneous network connectivity. The network layer additionally supports SLIP to interconnect the network layer with another computer over the USB interface.
- **RIME Layer**: It consists of several sublayers to support reusability of the link layer with different network protocols. Our RIME layer uses the chameleon, abc, broadcast and unicast sublayers.
- **Hop-to-hop reliability**: H2HR handles hop-to-hop reliability. Congestion control is performed by adapting the forwarded data rate. H2HR does not require any frame header.
- **Radio Duty Cycle mechanism**: Adaptive radio duty cycle mechanisms are employed to save energy and to adapt the sleeping period length according to the current traffic load. All used frame formats are IEEE 802.15.4 compliant.
- **Physical Layer**: The energy efficient CC2420 radio module is used on the physical layer. The CC2420 radio module is robust against interferences and is IEEE 802.15.4 compliant to support network connectivity. In addition, it supports and exploits the offered hardware functions of the CC2420 radio module.

All our protocols use either RFC or IEEE standard conform protocol headers. This enables communication between heterogeneous nodes inside the WSN and to nodes located in the Internet. The standardized physical layer enables direct communication between different sensor nodes. The support of IP on the network layer provides routing and forwarding of packets between sensor nodes and the Internet, without using any proxy mechanisms. The RFC compliant transport protocols UDP and TCP and the application layer protocol UDP-E2E enable reliable end-to-end data flows between WSN nodes and servers in the Internet.

Chapter 5

Evaluation

This chapter describes the evaluation of our contributed protocols presented in Chapter 4. In a first step we evaluate the performance of the different protocol options introduced in the previous chapter. Based on this evaluation, we are able to select the best protocol parameterization for the final version of our protocols and the resulting network stack. In a following step, we compare the final protocol versions of BEAM and H2HR to existing protocols.

Section 5.1 starts with the description of the different experiment scenarios and setups. This includes the description of the used network topologies and energy measurement methodologies to evaluate the energy usage and reliability performance under different conditions. Section 5.2 describes the evaluation of the different protocol versions of BEAM introduced in Section 4.1.3. The results are used to select the best performing optimization techniques for the final BEAM implementation. Then, Section 5.3 analyzes the H2HR hop-to-hop reliability mechanisms introduced in Section 4.2. The results indicate the best hop-to-hop reliability mechanisms for H2HR. The impact of FEC codes on the energy efficiency and reliability performance is evaluated in Section 5.4. The results show whether FEC codes have a benefit to our network stack. Section 5.5 compares our final protocol stack with protocols implemented in Contiki. These protocols are designed for packet oriented radio modules. In Section 5.6 we compare BEAM/H2HR to protocols that require bit/byte oriented radio modules. This is done by comparing real world results of BEAM/H2HR to the theoretical performance of the introduced hypothetical link layer protocol on the different radio modules. Finally, Section 5.7 describes the experienced differences of results achieved by our tests in a simulator and by our real world measurements in a testbed. Additionally, it outlines the impact of the used traffic load pattern and network topology on the received evaluation results.

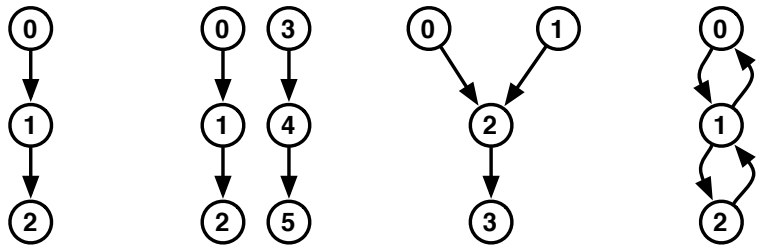
5.1 Evaluation Setup

A main contribution of this thesis is the design and implementation of an energy efficient and reliable link layer protocol that can handle any kind of traffic patterns

in different network topologies. Many existing energy efficient link layer protocols have been evaluated at low traffic load values or only in simple network topologies. These evaluation setups are not suitable to evaluate the impact of intra-flow and inter-flow interferences. Therefore, we define more challenging traffic patterns and network topologies generating significant internal interferences. Subsection 5.1.1 describes our four small-scale network topologies to cover the basic traffic patterns appearing within a WSN. The small-scale network topologies are used to evaluate the impact of intra-flow and inter-flow interferences on different protocol optimizations. Then, Subsection 5.1.2 introduces our three large-scale scenarios with individual traffic patterns. The purpose of the three scenarios is to cover most of the traffic patterns and resulting interference patterns generated by real world WSN applications. These topologies are used to validate protocol optimizations in larger scenarios and to compare our network stack to other protocol implementations delivered by Contiki. Subsection 5.1.3 describes which performance characteristics are evaluated and how they are defined. Especially, the determination of the energy consumption is challenging and requires an accurate preparation. Finally, Subsection 5.1.5 depicts the analyzed Contiki network stacks including all contributed and evaluated protocols.

5.1.1 Small-scale Scenarios and Testbed Setup

First, we define four different small-scale network topologies to cover the basic traffic and interference patterns appearing within a WSN. These topologies are used for basic tests, i.e. to analyze and compare the impact of the different protocol optimization techniques. A schematic overview of the four defined small network scenarios is given in Figure 5.1. The line scenario (5.1a) is the simplest one. This offers the possibility to analyze the forwarding performance of a node. The parallel scenario 5.1b enables the evaluation of inter-flow interferences. The merging scenario 5.1c represents a common network condition, e.g. the traffic flow from several source nodes towards the sink. One node must handle and forward traffic of two different nodes. The cross scenario 5.1d evaluates the impact of crossing traffic as well as intra-flow interferences. We implemented these four scenarios in



(a) Line scenario. **(b)** Parallel scenario. **(c)** Merging scenario. **(d)** Cross scenario.

Figure 5.1: Small-scale network topologies for basic evaluations.

the OMNeT++ network simulator, in the WISEBED WSN real world testbed and in a local small-scale testbed consisting of six telosB nodes.

The nodes of the local small-scale testbed are connected over USB to a single desktop computer as depicted in Figure 5.2. The local small-scale testbed enables a detailed evaluation of high traffic load. Every sensor node in this testbed receives radio transmissions from every other node with high SNR. This reduces collisions to a minimum, as every node is able to detect all possible transmissions before starting an own transmission. Collisions in such scenarios happen when two nodes start to send at the same time. The high SNR of all possible connections prevents a hidden node problem. External interferences show only low impact on the bit error ratio due to high SNR.

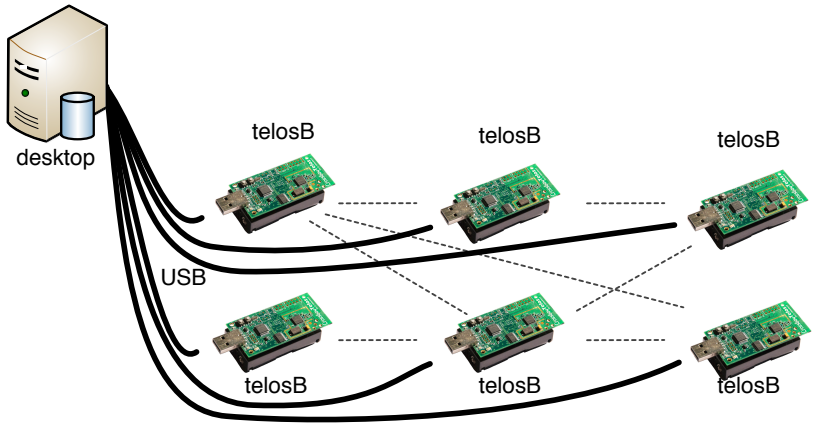
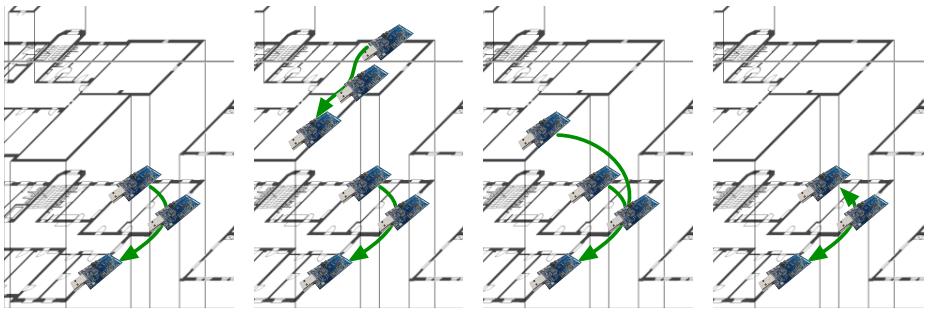


Figure 5.2: Local small-scale testbed with six telosB nodes.

The four small-scale scenarios used in the WISEBED testbed are shown in Figure 5.3. The line and cross scenarios consist of the same three nodes. These three nodes are also used to setup the parallel and the merging scenario.



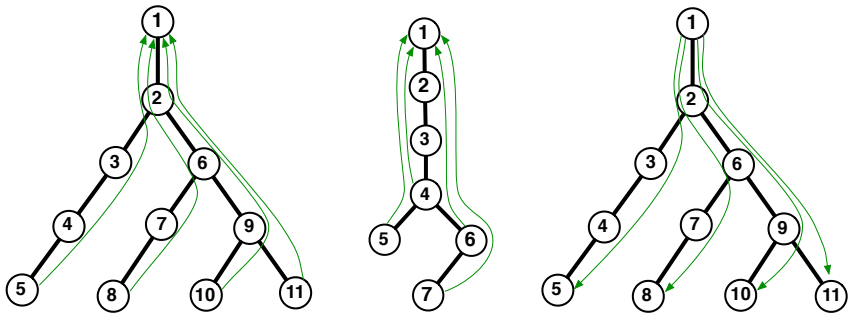
(a) Line scenario. **(b)** Parallel scenario. **(c)** Merging scenario. **(d)** Cross scenario.

Figure 5.3: Small-scale testbed setup in the WISEBED testbed.

The WISEBED testbed and the local small-scale testbed show some differences. First, the longer distances between the nodes in the WISEBED testbed result in a lower SNR while receiving packets. This results in a higher packet loss in the WISEBED testbed. Second, the WISEBED testbed uses TARWIS for the automatization of the experiment execution. This gives us the opportunity to execute a large amount of measurements without any manual interaction. Experiments with the local testbed have to be started and controlled manually. Third, the offered TARWIS event recording system provides the evaluation results in the WiseML format [20], which enables comfortable analyzation of the measurements. Unfortunately, the recording performance of the TARWIS event recording system is quite limited. Too closely spaced reporting events may result in faulty WiseML evaluation reports. We experienced a limit of 20 to 30 records per second in the final WiseML evaluation report. This is caused by the mechanism used by TARWIS to write events reported from the sensor nodes to the database on the TARWIS portal server. We added sequence numbers to the Contiki logging statements to identify entries that have been dropped by the TARWIS event recording system. In addition, the logging frequency of the individual nodes has to be low to not overload the TARWIS event recording system for throughput measurements. The local testbed with telosB nodes directly connected to a desktop computer, supports at a significantly higher recording rate. This enables measuring the throughput of the basic scenarios and sending instructions to two nodes at the same time. The testbeds introduced in this subsection are basically used to compare the performance characteristics of the different BEAM and H2HR protocol versions.

5.1.2 Large-scale Scenarios and Testbed Setup

We implemented the three large-scale scenarios in the OMNeT++ network simulator and in the WISEBED testbed. With the three large-scale scenarios, we evaluate the reliability performance of the different H2HR mechanisms and compare our final protocols to already existing ones. Figure 5.4 shows the different traffic patterns and network topologies used in the large-scale scenarios. In the streaming



(a) Stream scenario. **(b)** Event scenario. **(c)** Burst scenario.

Figure 5.4: Large-scale network topologies with corresponding traffic pattern.

scenario, four leaf nodes are sending continuously small packets to the sink. It represents a WSN scenario collecting sensor data from the environment. In the event scenario, four neighbor nodes detect an event at the same time. These four nodes simultaneously try to send a notification message to the sink. In the burst scenario the root node sends 800 bytes to all four leaf nodes. For example, this may represent a code update.

Figure 5.5 shows the used telosB nodes for the streaming and burst scenarios in the real world WISEBED testbed at IAM. This is the largest mesh network that we were able to build with the WISEBED testbed, as the amount of nodes having three neighbor nodes with an adequate SNR is limited. An appropriate neighbor node obtains at most 10% packet loss caused by external interferences and wave propagation effects. Nodes located far from each other in this topology are out of each other's detecting range. However, they still interfere with each other, which enhances the probability of bit errors and can cause a hidden node problem. This results in clearly more challenging network conditions than in case of small-scale topologies.

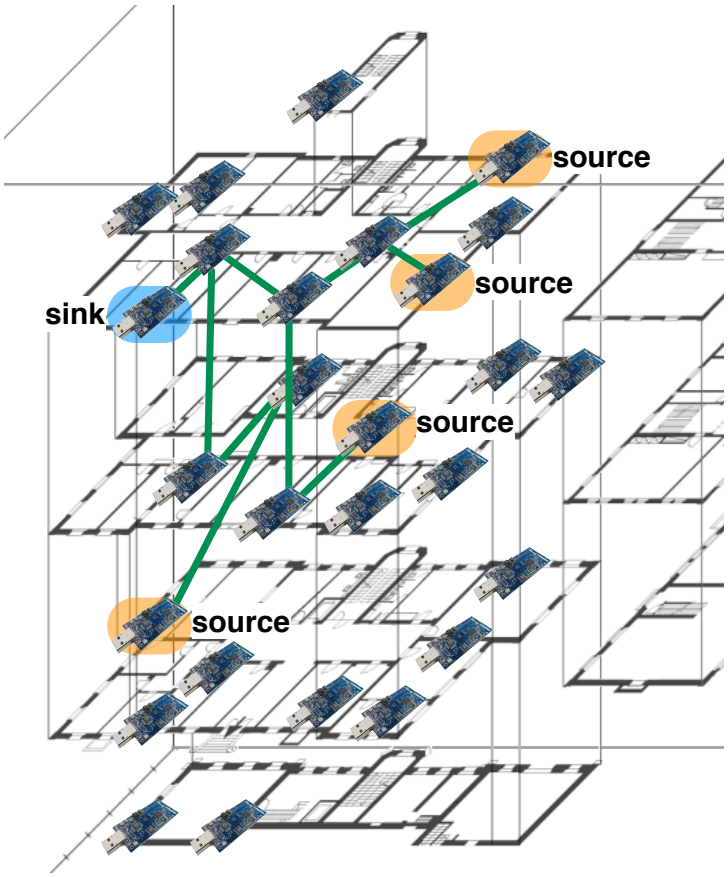


Figure 5.5: Large-scale testbed scenarios in the WISEBED testbed.

A basic requirement of the evaluated traffic load is that more than one single packet is simultaneously forwarded in the network. Evaluations, in which each generated packet can pass the network before the next one is generated, do not deliver significant results. They completely ignore the impact of internal interference, which represents the major challenge for any link layer protocol.

In all large-scale scenarios, we additionally record the energy profile of neighbor nodes, which are not participating in data forwarding. These nodes periodically wake-up to check the channel for packets that are addressed to them. With the WISEBED testbed, we record the energy consumption of all 27 nodes located in the building of the Institute of Computer Science and Applied Mathematics (IAM).

5.1.3 Evaluated Performance Characteristics

In this subsection, we define the evaluated performance characteristics. We use the following four characteristics to compare the performance of the different protocol designs:

1. **Energy consumption**: Measuring the energy consumption is a very challenging very task. We use two different techniques to determine energy consumption. On the one hand, we use the RIGOL digital multimeter to measure the electrical current on a single node. On the other hand, we use a software based energy profiler that records the radio state switches.

2. **Reliability**: To determine the reliability performance we use the three metrics: ETX, end-to-end packet loss ratio and end-to-end retransmission attempts, which were all introduced in Section 2.5.6. The reliability performance is closely connected to throughput (3.).

3. **Throughput**: The throughput shows how many packets can be successfully forwarded to target nodes for a given time interval.

4. **Packet delivery time**: The packet delivery time is defined as the time between the first transmission by the source node and the successful reception by the sink.

Measuring energy consumption is technically more complex than determining packet loss, throughput and packet delivery time. To measure energy usage of a single node, we used a digital multimeter. To simultaneously determine the energy consumption of the individual sensor nodes in our real world testbeds, we used a software based energy profiler. The next subsection explains the different setups to determine energy consumption. This includes a description of the modification and verification of the used software based energy profiler.

5.1.4 Energy Evaluation Techniques

We use a RIGOL DM3052 digital multimeter introduced in Subsection 2.4.3 and a software based energy profiler introduced in 2.3.1 to measure energy consumption.

Figure 5.6 shows the used evaluation setup to measure the electrical current of a single sensor node. The used power source is a VOLTCRAFT VLP-1303 PRO power supply described in Subsection 2.4.3. We solder copper wires to one of the telosB node. The copper wires are connected to the power supply at 3.00 V and the RIGOL multimeter. The RIGOL multimeter is connected over an USB interface to a notebook running the measurement and control software. The buffer size of the RIGOL multimeter is limited to 2'100'000 recording samples. This results in a maximum recording duration of 42 seconds when recording 50'000 samples per second. The energy profiles recorded by the measurement software can be used for the following purposes:

- **Energy consumption**: The energy profile shows the varying energy consumption over a certain period of time. It helps identifying the varying radio and microcontroller states and determining the corresponding energy consumption.

- **Energy profiler configuration and verification**: The energy profile calculated by the energy profiler can be compared to the energy profile recorded by the RIGOL multimeter.

- **Tracking radio and microcontroller state switches**: The high resolution of the RIGOL multimeter enables monitoring of the CC2420 radio state switches as well as tracking of the microcontroller activity. Microcontroller tracking can be used to exactly determine the computation time for encoding FEC codes. Tracking the radio state switches enables debugging and optimization of protocol timers.

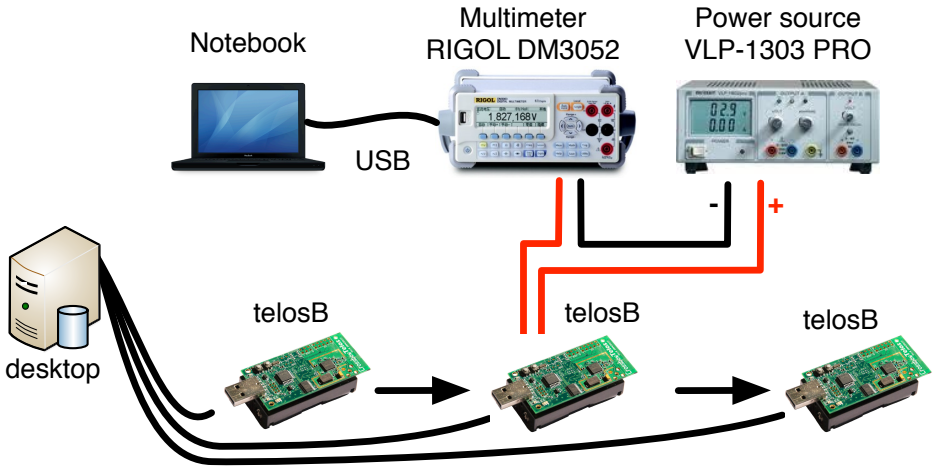


Figure 5.6: Experimental setup to measure the electrical current of telosB node.

The RIGOL multimeter can only monitor one sensor node for a limited time period. To monitor every node within a network topology, a software based energy profiler of Contiki can be used. The energy profiler estimates the current energy usage according to the currently executed tasks. It is integrated into the radio driver, where the radio switches are handled. It measures how long the radio module remains in the individual radio states. The duration is measured in steps with a resolution of 31 μs. The Contiki profiler for the CC2420 distinguishes between three radio states: sleeping, sending and listening. Listening and receiving are represented by the same state. We use the same approach to measure the energy with the OMNeT++ network simulator. We integrated the energy profiler into a newly implemented CC2420 radio module, which we developed according to the technical specifications of CC2420 radio module for OMNeT++. To calculate the energy consumption (E_{radio}), we assumed that every radio state has a constant energy usage. The energy is calculated by multiplying the recorded time periods (Δt) of the individual radio states with the according electrical current (I) and battery voltage (U_{bat}) :

$$E_{radio} = U_{bat} \cdot ((I_{sleep} \cdot \Delta t_{sleep}) + (I_{send} \cdot \Delta t_{send}) + (I_{listen} \cdot \Delta t_{listen}))$$

The electrical current measured for the different radio states corresponds to the values declared in the CC2420 manual. In a following step, we evaluate the accuracy the energy profiler by comparing the profiles recorded by the RIGOL multimeter with profiles generated by the Contiki energy profiler. This is done in two steps. First, we compare the energy profiles generated for typical link layer operations such as a channel check or sending/receiving a packet. Based on these results, we adapt the Contiki energy profiler to the CC2420 radio module and the used protocol algorithms. Then, we simultaneously measure the energy required to forwarding packets by the RIGOL multimeter and the adapted Contiki energy profiler. The results show the accuracy of the adapted Contiki energy profiler.

Energy Consumption in Idle Mode

To verify the Contiki energy profiler, we started analyzing the energy consumption of Contiki in idle mode. In idle mode the radio module is turned into the *Power Down* state and all sensors are turned off. The Contiki operating system minimizes the energy usage of a microcontroller by keeping it most of the time in sleep mode.

The energy profile of Contiki recorded by the RIGOL multimeter is shown in Figure 5.7. The small energy peaks that occur every 7.8 ms are caused by periodic wake-ups of the microcontroller. The mean energy required by a telosB node running Contiki at a voltage (U_{bat}) of 3.00 Volt is 0.4779 mA with a standard deviation of 0.0129 mA caused by small energy peaks. According to the CC2420 manual, the radio module is using 0.020 mA in *Power Down* mode. This value cannot be measured without removing the radio module from the telosB node. We subtract the energy used by the CC2420 radio in *Power Down* mode (I_{radio}) from the total

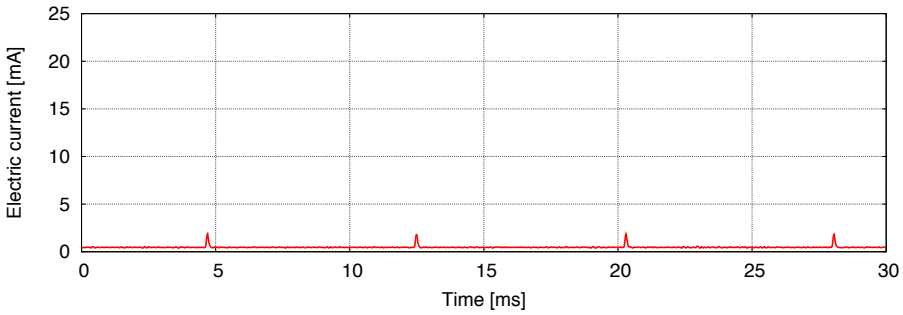


Figure 5.7: Contiki energy profile on a telosB node in idle mode.

energy used in idle mode (I_{idle}) to calculate the amount of energy used by telosB running Contiki in idle mode (E_{idle}):

$$E_{idle} = U_{bat} \cdot (I_{idle} - I_{radio}) \cdot t)$$
$$E_{idle} = 3.00V \cdot (0.478mA - 0.020mA) \cdot 1s = 1.374\mu J$$

Energy Consumption of Channel Checks

Next, we analyze the energy required for a channel check if no traffic can be detected. We recorded 4'000 channel checks with the RIGOL multimeter and the Contiki energy profiler. 2.2% of the performed channel checks were wrongly detected as ongoing transmissions. This is because the measured RSSI level during the channel check was above the defined threshold. We removed these wrongly detected channel checks to calculate the energy required for a channel check.

Figure 5.8 shows the energy profile for a channel check recorded by the RIGOL multimeter. The peaks represent the two individual channel checks required by ContikiMAC and BEAM to detect ongoing beacon strobe transmissions. The red area shows the energy measured by the RIGOL multimeter. The radio driver switches the radio module on at position (1) in Figure 5.8. Now, the radio module switches from *power down* mode to *receive* mode. At point (2) the radio module is able to listen to the channel. Now the channel must be checked for at least eight symbols (128 μS) to ensure a valid CCA value. At point (3), the microcontroller reads out the CCA-pin of the CC2420. If no ongoing transmission has been detected, the radio module is switched off. The ending shapes of the two peaks are a little bit different as the RDC protocol must process the results of the channel check and calculate the next wake-up. This requires some additional energy by the microcontroller. For calculating the energy required by the radio module and the microcontroller, we subtract E_{idle} from the RIGOL energy measurements. The first peak requires 19.28 μJ with a standard deviation of 0.0034 μJ. The second

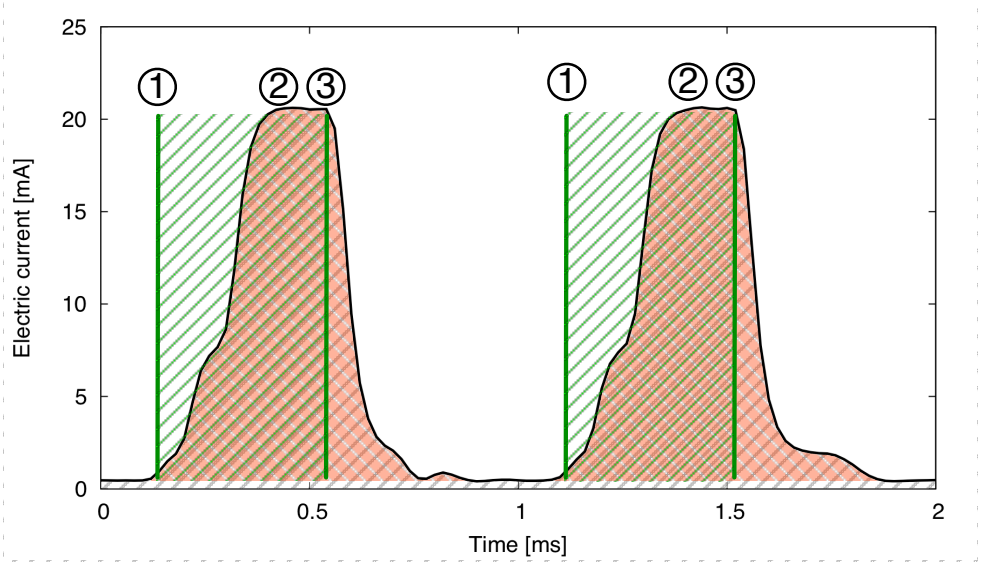


Figure 5.8: CCA channel check energy profile of a CC2420 radio.

peak requires 19.64 μJ with a standard deviation of 0.0039 μJ. This includes the additional energy required by the microcontroller to perform the channel check.

The green area shows the energy measured by the Contiki radio energy profiler. For both peaks, always a time period of 10 time steps (0.31 ms) is measured. The Contiki radio energy profiler assumes a constant current during this time period. This results in 18.3 μJ per peak using a current of 19.7 mA as defined in the manual [91]. If we add the energy from the microcontroller energy profiler, we get 19.5 μJ per peak.

In addition, we analyzed the energy required for a wrongly detected transmission by the channel check. In this case, the RDC layer keeps the radio module in listening mode to wait for the next frame start. If the frame start is still missing after a certain time out, the radio module is switched off.

Energy Consumption for Sending and Receiving a Single Frame

Besides the channel check, we analyze the energy required for sending and receiving a single frame. Figure 5.9 shows the recorded energy profiles by the RIGOL multimeter for a sending node and the corresponding receiving node. The sending node performs two channel checks to ensure a free channel (see previous subsection). Then, it starts to periodically send beacon strobes including payload until it receives an acknowledgment from the receiver. The receiver wakes up while the sender is transmitting the third beacon strobe. The frame start of the actual frame is missed. The receiver waits for the next beacon strobe and sends an acknowledgment after checking the CRC.

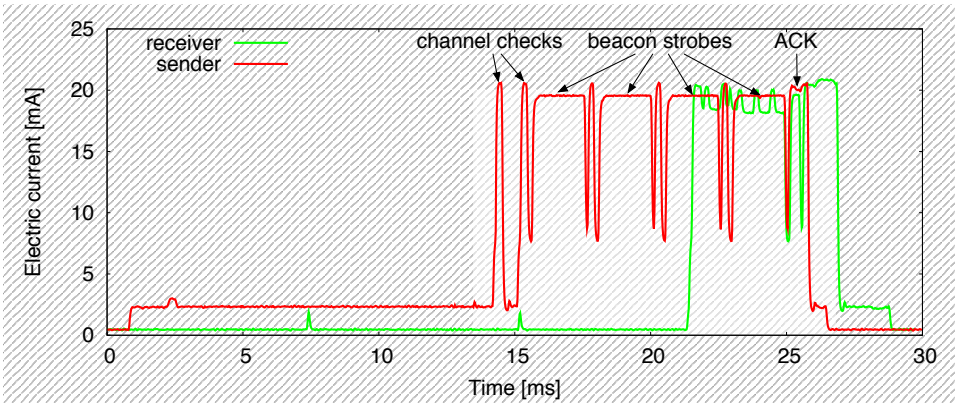


Figure 5.9: Energy profile of a packet forwarding with beacon strobes.

We divided sending and receiving into individual smaller tasks. Sending is divided into channel checks before sending, strobe sending, channel checks for acknowledgment and acknowledgment receiving. Receiving is divided into channel checks before receiving, data receiving, acknowledgment sending and listening after acknowledgment sending. We analyze the energy consumption for the different tasks with the RIGOL multimeter and the Contiki energy profiler. We adapted the Contiki energy profiler in the following way. For tasks with dynamic durations, we record their duration. For tasks that have always the same duration and energy consumption, we simply count the number of their occurrences.

Verification of the Software Based Energy Profiler

To verify the adapted Contiki energy profiler, we use the measurement setup from Figure 5.6. We forwarded one packet per second over two hops. The forwarding node in the middle is connected to the RIGOL multimeter. We made 50 measurements for BEAM, XMAC and ConitkiMAC with 40 and 100 bytes payload. In each measurement, 30 packets are forwarded. The energy used to forward one individual packet was measured simultaneously with two different methods. We measured once using the RIGOL multimeter and once using the adapted Contiki energy profiler. The energy for an individual packet measured by the RIGOL multimeter is used as reference value. The deviation of the results achieved by the adapted Contiki energy profiler to the measurements of the RIGOL multimeter is shown in Figure 5.10. The energy recorded by the software based energy profiler shows a mean deviation below 4% compared to the energy measured by the RIGOL multimeter. There is no significant difference in the accuracy between frames with 40 and 100 bytes payload.

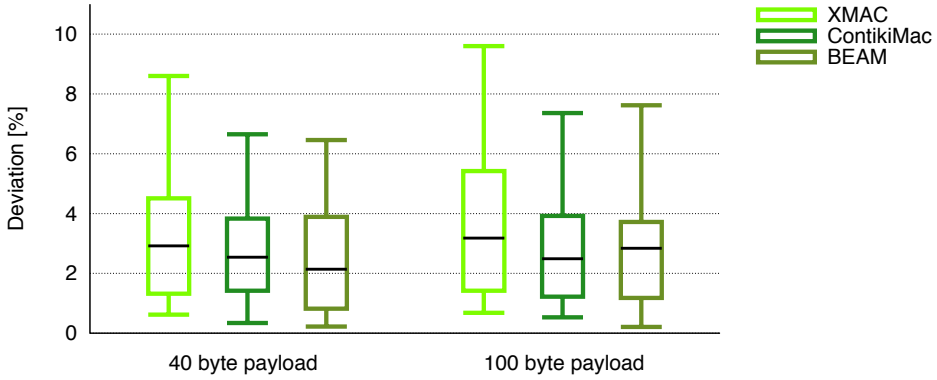


Figure 5.10: Deviation of the used software energy profiler to the RIGOL multimeter.

Energy Evaluation Conclusions

The RIGOL multimeter is able to generate a very accurate energy profile of a single node with 50'000 samples per second. Unfortunately, the RIGOL multimeter buffer size is limited. This results in a maximum recording duration of 42 seconds with 50'000 samples per second. Here, the adapted Contiki energy profiler provides an interesting solution, as it is able to record energy consumption with an adequate accuracy for all nodes within the WSN. This software based energy profiler can log the energy of every node within a real world testbed over the entire experiment execution time. We used the adapted Contiki energy profiler in all of our evaluations.

5.1.5 Evaluated Contiki Compliant Network Stacks

Our contributed link layer protocols BEAM and H2HR have been integrated into the Contiki network stack. BEAM is fully compliant to the *Radio Duty Cycling* layer protocol defined by the Contiki network stack. The same applies to H2HR concerning the *Hop to Hop Reliability* layer. On top of the link layer, we use Contiki's RIME layer to interconnect the link and network layer. On the network layer, μIP is used. UDP and TCP are used on the transport layer. TCP is additionally used to support the required end-to-end reliability. To improve UDP, we use the application layer overlay protocol UDP-E2E to support end-to-end reliability.

Figure 5.11 shows the reliable network stack used with UDP. The *Evaluation Application* located on the application layer is able to receive instructions from the TARWIS experiment management system. We use these instructions to create and schedule the injection of new packets into the network. The schedule and subject of the instruction commands can be predefined by the WISEBED testbed. The *Evaluation Application* adds an incrementing 16-bit number to the payload of every generated packet. The generated sequence numbers and the sequence number

of each received packet are reported to the TARWIS event recording system. The timestamp mechanism of the TARWIS event recording system enables measuring one way packet delivery time of each packet.

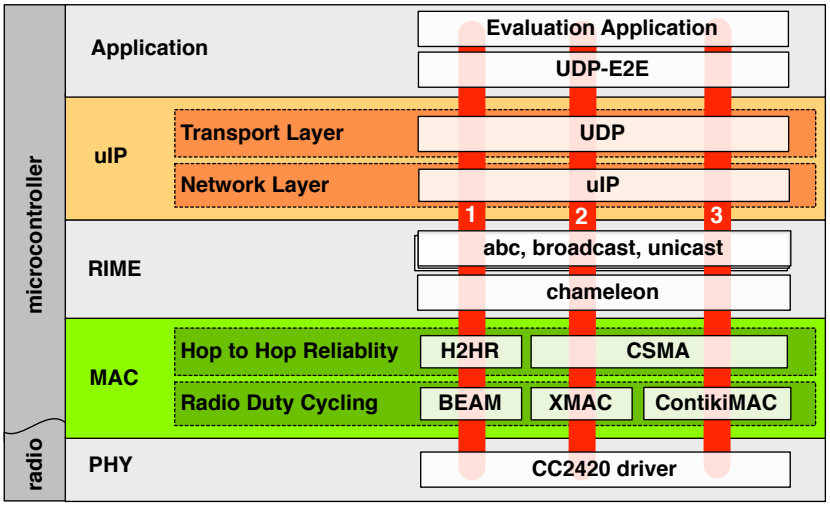


Figure 5.11: Reliable Contiki compliant network stacks with UDP.

Figure 5.12 shows the reliable network stacks used with TCP. It offers the possibility to compare the end-to-end reliability support of TCP and UDP. TCP sends positive acknowledgments for successfully received data frames. While UDP-E2E is sending negative acknowledgments for missing data frames.

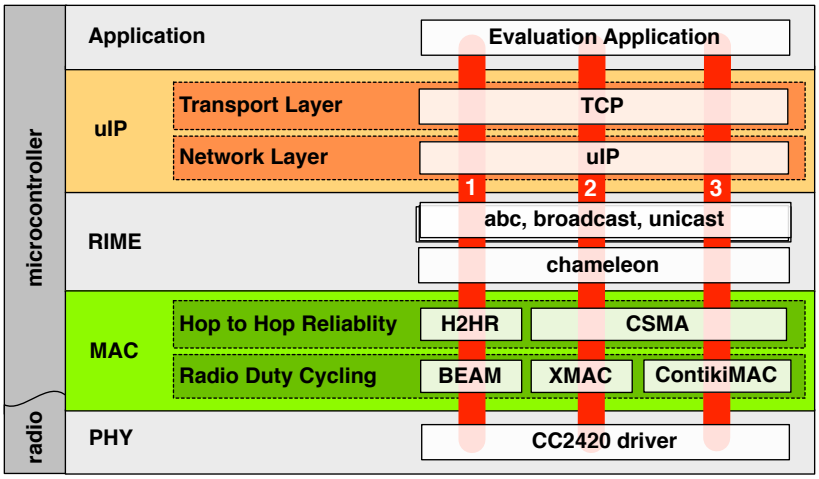


Figure 5.12: Reliable Contiki compliant network stacks with TCP.

5.1.6 Summary of Evaluation Setup

In this section, we defined the different network topologies and performance characteristics used to analyze and compare WSN network protocols. Four small-scale network topologies are defined to analyze and compare different protocol optimization techniques of BEAM and H2HR. These small-scale network topologies were implemented in the OMNeT++ network simulator, in the WISEBED testbed and within a local small-scale testbed consisting of six telosB nodes. Additionally, we defined three large-scale topologies with according traffic patterns to achieve realistic interference patterns similar to the ones generated by real world WSN applications. They are used to optimize the reliability mechanisms of H2HR and to compare the finalized network stacks to other protocol implementations delivered by Contiki. The large-scale network topologies were implemented in the OMNeT++ network simulator and the WISEBED testbed. We defined four performance characteristics, namely energy consumption, packet loss, throughput and packet delivery time, to evaluate and compare the protocol implementations. The next section uses the small-scale network topologies to find the best protocol optimization techniques for BEAM.

5.2 Evaluation of BEAM Protocol Optimization Techniques

This section presents the evaluation of the different protocol versions and optimization techniques of BEAM. The results are used to identity the best performing optimization techniques and protocol versions for the final version of BEAM. The following list summarizes the individual optimization techniques and performed evaluations considered in this section.

- **Acknowledgment mechanism:** We compare hardware and software acknowledgments by analyzing their energy profiles and the required execution times. The energy and time profiles are established using measuring by the RIGOL multimeter.
- **Beacon strobe transmission delay:** First, we test the impact of the beacon strobe transmission optimization on the energy consumption measured by the RIGOL multimeter. Then, we use BEAM in the four small-scale scenarios in the WISEBED testbed to measure the impact of the transmission delay optimization on the energy consumption and reliability performance.
- **Beacon strobe design:** We compare BEAM with short beacon strobes and its variation with beacon strobes including payload in the four small-scale scenarios in the WISEBED testbed. We evaluate the impact of the traffic load and the payload size on the energy consumption and reliability. Additionally, we analyze the energy consumption of the noninvolved idle neighbor nodes.
- **Duty cycle evaluations:** We evaluate the energy consumption and throughput of different duty cycle durations. During low traffic load periods, the

applied duty cycle duration should use as low energy as possible while still providing short packet delivery times. In case of high traffic load, the duty cycle duration must offer maximum throughput. We search for ideal duty cycle durations for both situations. We perform the evaluation using the local small-scale testbed with six telosB nodes to enable the evaluation of throughput.

- **Traffic prediction versus traffic monitoring:** We compare our duty cycle adaptation mechanism based on traffic prediction with traffic monitoring, which is used by other adaptive protocols. We evaluate the impact on energy consumption, packet loss and throughput. This evaluation was performed by using the four small-scale scenarios in the WISEBED testbed.
- **Packet aggregation:** We evaluate the impact of packet aggregation on energy consumption and reliability by the four small-scale scenarios in the WISEBED testbed.

5.2.1 Acknowledgment Mechanism

This subsection compares the energy consumption of hardware and software acknowledgments. In a first step, we compare the energy profiles of a sender sending beacons strobes with both acknowledgment mechanisms. The frequency of the beacon strobe transmission determines how long a receiver has to listen to the channel to detect a beacon strobe. In a second step, we evaluate the resulting energy profiles for a receiver performing a channel check to detect a beacon strobe. An energy profile depicts the power consumption over a certain time period.

Figures 5.13 and 5.14 show two recorded energy profiles for a **sender** transmitting BEAM beacon strobes with the size of 56 bytes including the physical preamble. In the depicted examples, the corresponding receiver never wakes up to send the acknowledgment. Both energy profiles are recorded by the RIGOL multimeter with 50'000 samples per second. The energy profile in Figure 5.13 represents BEAM with enabled hardware acknowledgment support. The energy profile in Figure 5.14 represents BEAM with software acknowledgments.

Both versions use the same mechanism to send a beacon strobe. At point (1), they turn the radio module into transmission mode to start the transmission of the next beacon strobe. The beacon strobe with a size of 56 bytes is transmitted by the CC2420 radio module at (2). The subsequent mechanism and time required to check the channel for an incoming acknowledgment of both versions is different.

The sender supporting BEAM with **hardware acknowledgments** switches directly to listening mode after sending the last byte. BEAM checks the channel to detect if there is an ongoing transmission (3). If no transmission has been detected, no acknowledgment was sent by the receiver. BEAM switches the CC2420 again to transmission mode to send the next beacon strobe (4).

The sender using BEAM with **software acknowledgments** turns off the radio module after sending the last byte of the beacon strobe (5). A receiver requires from

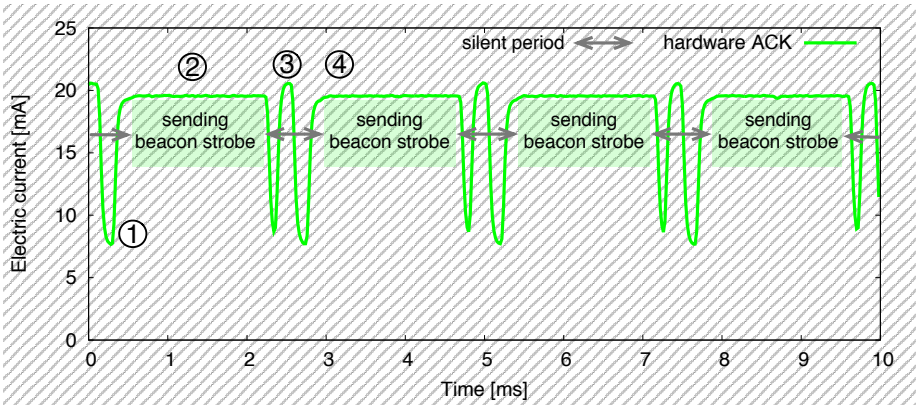


Figure 5.13: Energy profiles of beacon strobes with hardware acknowledgment.

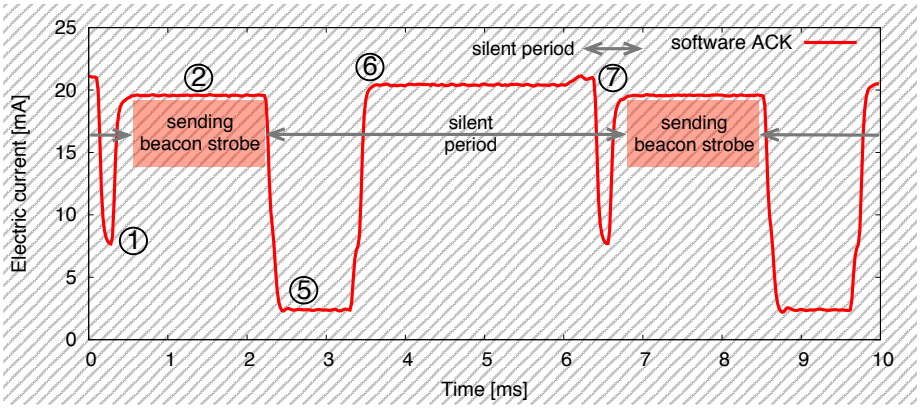


Figure 5.14: Energy profiles of beacon strobes with software acknowledgment.

1.7 ms to 4.2 ms to copy the received beacon strobe to the microcontroller, to create and copy the acknowledgment to the radio module and start the transmission. 1.3 ms after turning off the radio module, the sender turns on the radio module, switches it into listening mode and waits for the acknowledgment from the receiver (6). Experiments show that turning on the radio module after 1.4 ms causes a loss of 10% of the acknowledgments. The listen period must cover the entire time period during an acknowledgment could be received. If no acknowledgment has been detected, the next beacon strobe is sent (7).

The **silence period** shown in Figures 5.13 and 5.14 to check the channel for an incoming acknowledgment determines how long a **receiver node** has to listen to the channel to detect a beacon strobe transmission. The channel checks, which are periodically performed by all nodes in the WSN, are the major energy consuming tasks during low traffic periods. Therefore, the periodical channel checks have to be as energy efficient as possible to achieve an energy efficient WSN. A channel

check can be either done continuously or periodically.

Figure 5.15 shows the two energy profiles of a receiver node to detect an incoming beacon strobe transmission with a continuous channel observation. For continuous listening, the radio module is turned on and immediately switched to listening mode. Then, one CCA check is performed to detect an ongoing transmission. If a CCA check detects an ongoing transmission, then BEAM keeps the radio module in listening mode to receive the next beacon strobe. If no transmission has been detected, BEAM keeps the radio module in listening mode to search for a frame start. The listen period corresponds to the remaining silence period between two beacon strobes. If still no frame start (SFD) has been detected after the remaining listen period, no beacon strobes are transmitted. BEAM turns off the radio module and calculates the start of the next wake-up period.

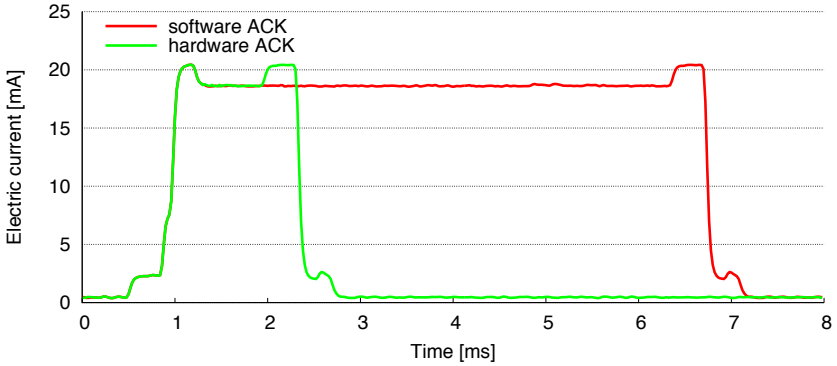


Figure 5.15: Channel check by a receiver node with continuous listening.

The channel observation can alternatively be done with periodic CCA checks. Figure 5.16 shows periodic CCA checks to detect ongoing beacon strobe transmissions with and without the CC2420 acknowledgment support. The periodic CCA

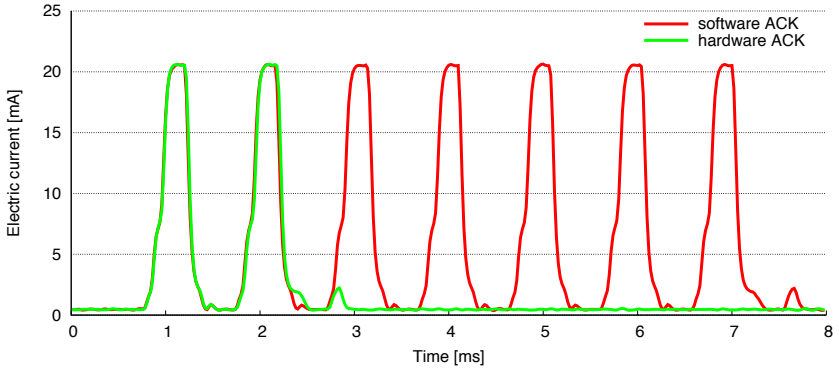


Figure 5.16: Channel check by a receiver node with periodic CCA.

checks are executed once per millisecond. Using hardware acknowledgment support, two CCA operations in a row are sufficient to detect beacon strobes. Using only software acknowledgments, seven CCA checks are required. The time interval between the individual CCA checks should not be longer. Otherwise, the time required to transmit a short beacon strobe is shorter than the time between the individual CCA checks. If a CCA check detects an ongoing transmission, then BEAM keeps the radio module in listening mode to receive the next beacon strobe.

Table 5.1 shows the required energy for the different channel check techniques shown in Figure 5.15 and 5.16. The results show that using hardware acknowledgments is clearly more energy efficient than using software acknowledgments. Moreover, it makes sense to use periodic CCA checking. An advantage using software acknowledgment is the possibility to modify the content of the acknowledgment. For example, to add node state informations to the acknowledgment. Based on the results of this subsection, we use hardware acknowledgments and periodic CCA checking for the final BEAM version.

Acknowledgments type	periodic CCA checking	Energy [μJ]	Standard deviation [μJ]
Software acknowledgment	No	322.59	0.4413
Software acknowledgment	Yes	116.13	0.0040
Hardware acknowledgment	No	82.79	0.0122
Hardware acknowledgment	Yes	**38.92**	0.0038

Table 5.1: Required energy for different kind of channel checks.

5.2.2 Beacon Strobe Transmission Delay Optimizations

This subsection describes the evaluation of the BEAM transmission delay optimization. Originally, BEAM immediately starts with sending beacon strobes after having received a single frame from the upper layer H2HR. The transmission delay optimization tries to estimate the next wake-up of the receiver. If an estimation was possible, BEAM delays the beacon strobe transmission near the expected wake-up time point of the receiver. Table 5.2 shows the used protocol properties to analyze the impact of the transmission delay optimization. To analyze the transmission delay optimization, we used the RIGOL multimeter setup from Figure 5.6. The node that is connected to the RIGOL multimeter creates a packet every second and forwards it to a second node.

Figure 5.17 shows a typical recorded profile with (red) and without (green) transmission delay optimization. At (1), the sensor application creates a packet and forwards it to the UDP socket. BEAM receives a 40 bytes frame from the upper layer, after the application payload has passed the network stack. BEAM without the transmission delay optimization immediately performs a CCA check

RDC protocol	BEAM
Strobe type	Including payload
BEAM-payload	40 bytes
Duty cycle duration	125 ms
Transport protocol	UDP
End-to-end reliability	No

Table 5.2: Measurement setup for transmission delay optimization evaluations

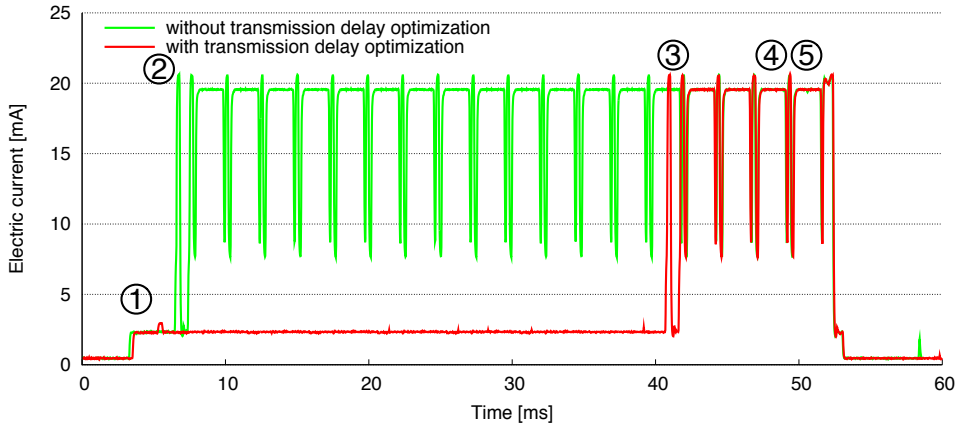


Figure 5.17: Energy profile of strobe transmissions with and without transmission delay.

and starts sending beacon strobes (2). This happens around 40 ms before the addressed receiver is expected to perform the next channel check at around (4).

BEAM using the transmission delay optimization postpones the beacon transmission until 8 ms before the estimated wake-up (3). The receiver node wakes up at (4) and detects an ongoing transmission. It receives the next beacon strobe and sends an acknowledgment back (5). Then both senders turn off the radio module. In this particular measurement, the node with the transmission delay optimization delays the transmission for 34 ms. This reduces the energy required by the sender in the shown measurement from 2436 μJ to 806 μJ.

Next, we use the WISEBED small-scale network topologies defined in Subsection 5.1.1 to analyze the impact of the transmission delay optimization on energy consumption and reliability. In each experiment, we send 10'000 packets at different data rates to evaluate the impact of the traffic load. First, we analyze the achieved reduction of the required amount of transmitted beacon strobes to forward a packet to the next hop. Figure 5.18 shows the measured delays applied by the transmission delay optimization.

- **Line scenario**: No serious interference or congestion problems appeared. BEAM is able to delay most of the beacon strobe transmissions until the ex-

pected wakeup of the receiver. Every packet reaches the sink node, before the next packet is generated. The transmission delay optimization significantly reduces the amount of transmitted beacon strobes in this scenario.

- **Parallel scenario**: There are only few collisions at higher traffic load as the individual nodes have individual wake-up periods. If two nodes have to forward a packet at the same time, the receivers usually have different wake-up time points. Therefore, the beacon strobe periods of the different transmissions do not overlap. Most of the transmission delay optimizations can be applied successfully.

- **Merging and cross scenarios**: These scenarios are more challenging with higher traffic load. The middle node receives traffic from two senders. The performance corresponds to the line scenario as long as only one single packet must be forwarded at the same time. But if both senders try to send a packet during the same period to the middle node, the beacon strobes will collide. One packet then must be retransmitted later.

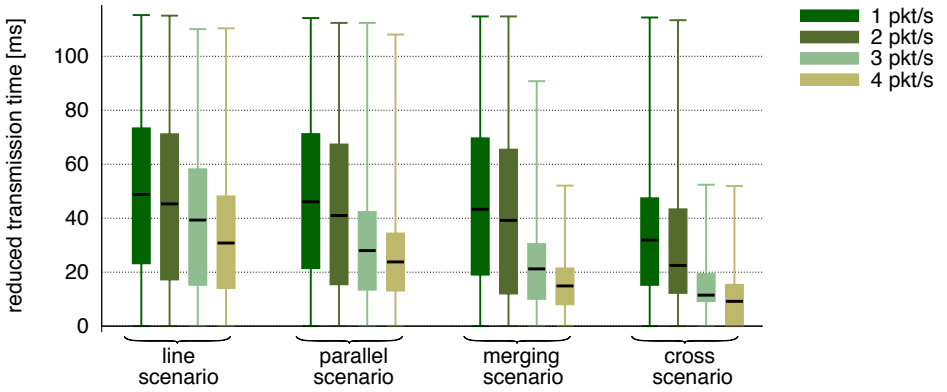


Figure 5.18: Reduction of the beacon strobe transmission time period.

The transmission delay optimization shows the best performance at low traffic load and basic network topologies defined in Subsection 5.1.1. The transmission delay optimization then significantly reduces the required beacon strobes.

The impact of the transmission delay optimization on energy costs is evaluated in the next experiment. We compare the energy consumption of BEAM with and without the transmission delay optimization in the same WISEBED testbed scenarios. The measured energy costs at two different traffic load values are shown in Figure 5.19. Moreover, we analyze the expected transmission count (ETX) introduced in Section 2.5.6. A retransmission is performed in case of a missing acknowledgment or if a collision has been detected during the beacon strobe transmission. Postponing a beacon strobe transmission due to a busy channel is not

counted as retransmission. Figure 5.20 shows the ETX count measured in the individual scenarios.

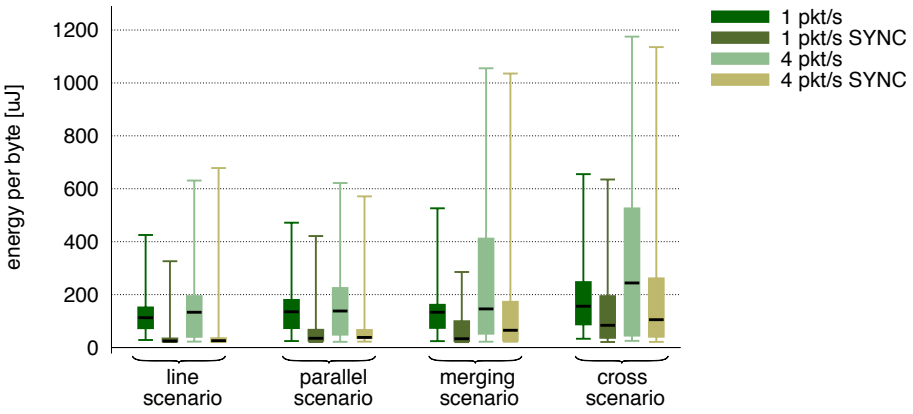


Figure 5.19: Energy per byte with and without transmission delay.

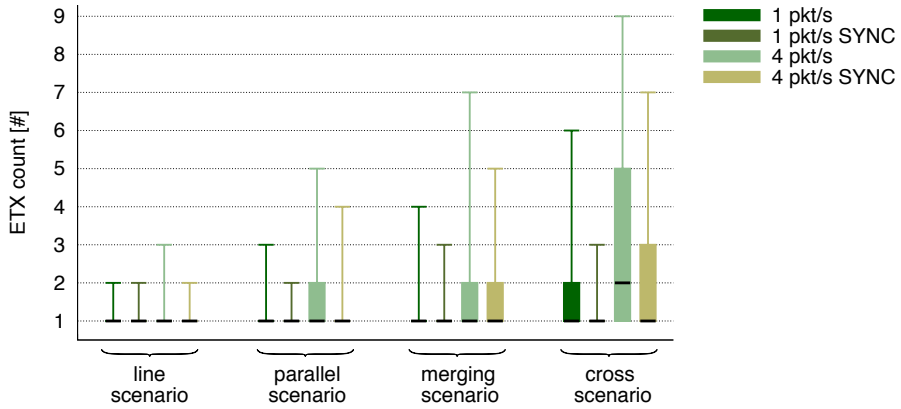


Figure 5.20: ETX count measured in the different scenarios.

The energy costs in the **line scenario** are significantly reduced by the transmission delay optimization. Over 95% of the packets can be forward at the next wake-up period of the receiver node. The transmission delay optimization works perfectly at higher traffic load rate. The energy costs for the **parallel scenario** are significantly reduced by the transmission delay optimization. Without transmission delay optimization, some retransmissions are required at higher traffic load as the longer beacon strobe periods result in a higher probability of collisions between the two paths in this scenario. The energy costs of the **merging and cross scenarios** are reduced considerably by the transmission delay optimization. These scenarios show several retransmissions at higher traffic load. They are basically caused by collisions when two nodes try do send a packet at the same time to the middle node.

At higher traffic load a sender sometimes must buffer several packets in the packet buffer. In this case, the BEAM traffic prediction mechanism reduces the duty cycle duration to increase the bandwidth. Shorter duty cycle durations reduce the average beacon strobes required by the BEAM version without transmission delay optimization.
We observed that in some cases that the transmission delay optimization cannot be applied or even fails:

1. If the neighbor table entry of the receiver has expired, no delay can be applied. This happens amongst others if too many transmissions fail.
2. The transmission delay is not applied if the expected wake-up time is within the next 8 ms.
3. If neither the acknowledgment nor a beacon strobe collision has been detected, the sender assumes that the transmission delay failed. In this case the neighbor table entry is marked as expired. For the next transmission, no transmission delay is used.

Due to the achieved results, the final BEAM version uses transmission delay optimization. It significantly reduces energy consumption and interferences.

5.2.3 Beacon Strobe Modes

This subsection evaluates the two BEAM beacon strobe versions introduced in Section 4.1.2. The BEAM version with beacon strobes including the payload requires only two instead of four successful transmissions to forward a data frame. On the other hand, the noninvolved neighbor nodes using the BEAM version with short beacon strobes require less energy to analyze overheard packets. Both versions use hardware acknowledgments and transmission delay optimization for beacon strobes. We compare their individual energy usage and reliability to determine the best beacon strobe strategy. Therefore, we evaluate the two versions in the WISEBED testbed small-scale network topologies defined in Subsection 5.1.1. We add six more nodes to each scenario to evaluate the impact of the beacon strobe mode on noninvolved neighbor nodes. Noninvolved neighbor nodes in these extended small-scale network topologies do not forward any traffic as they are never addressed. Sometimes, noninvolved neighbor nodes detect an ongoing beacon strobe transmission during their wake-up period. In this case, the node must receive the next beacon strobe to check the address of the beacon strobe. Table 5.3 shows the used protocol properties to analyze the impact of the beacon strobe design.

In each scenario extended with six noninvolved neighbor nodes, we sent 10'000 packets. The noninvolved nodes calculate the consumed energy to overhear the traffic in the neighborhood every second. First, we compare the energy consumption required to forward 40 bytes. Figure 5.21 shows the corresponding measured energy for both beacon strobe modes.

RDC protocol	BEAM
Strobe type	Including payload
BEAM-payload	40 & 100 bytes
Duty cycle duration	125 ms
Transport protocol	UDP
End-to-end reliability	No

Table 5.3: Measurement setup for beacon strobe evaluations

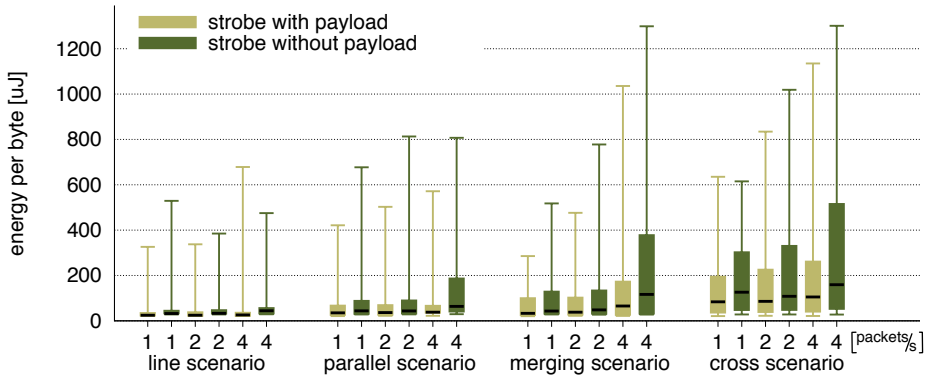


Figure 5.21: Energy per byte of involved nodes with 40 bytes payload.

The outliers are caused by retransmitted packets. In all tested scenarios, the variant using beacon strobes with payload requires less energy, especially at high traffic rates with more challenging interferences. This is basically due to three reasons:

1. With short beacon strobes, the receiver node has to receive at least two frames. This requires more energy than receiving a single beacon strobe including the payload.

2. With short beacon strobes, additional energy may be required to send the data frame part after receiving the early acknowledgment.

3. The short beacon strobe mode performed more frequently a retransmission as the receiver only has one chance to receive the data part after sending the early acknowledgment. If the payload is included in the beacon strobe, the data part is sent repeatedly. In case of corrupted data, the receiver can wait for the next beacon strobe.

Figure 5.22 shows the measured number of local hop-to-hop retransmissions. Postponing a beacon strobe transmission due to a busy channel is not counted as retransmission. The BEAM version using beacon strobes without payload requires more retransmission attempts, especially at higher data rates and in the challenging cross scenario.

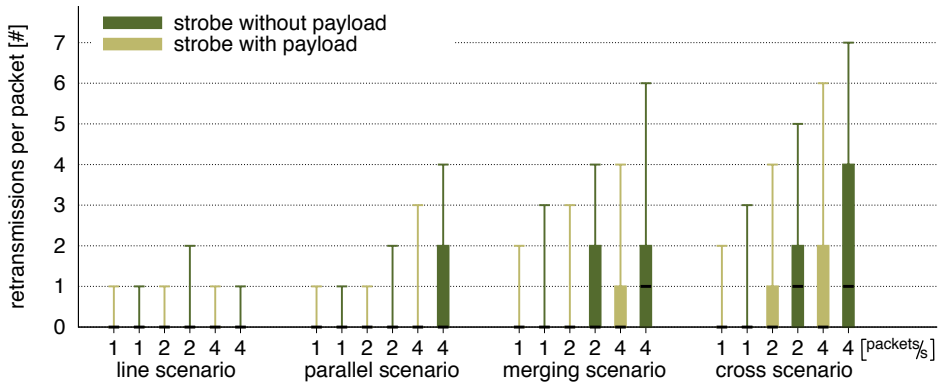


Figure 5.22: Required retransmissions per hop.

In addition, we tested the energy profile with the maximum BEAM payload of 118 bytes. Also with maximal payload, the version including the payload in the beacon strobe requires less energy. The required energy per byte decreases with increasing BEAM payload.

Next, we analyze the energy consumption of the noninvolved neighbor nodes. Here, we periodically measured the required energy consumption every second. Figure 5.23 shows the energy consumption of the noninvolved neighbor nodes with a payload of 40 bytes, Figure 5.24 shows the one with the maximum payload. All noninvolved nodes require more energy when using the version with beacon strobes including payload. Especially, high traffic load and large payloads require clearly more energy. The higher the traffic load is, the higher is probability of detecting an ongoing transmission. The beacon strobes including the payload are significantly

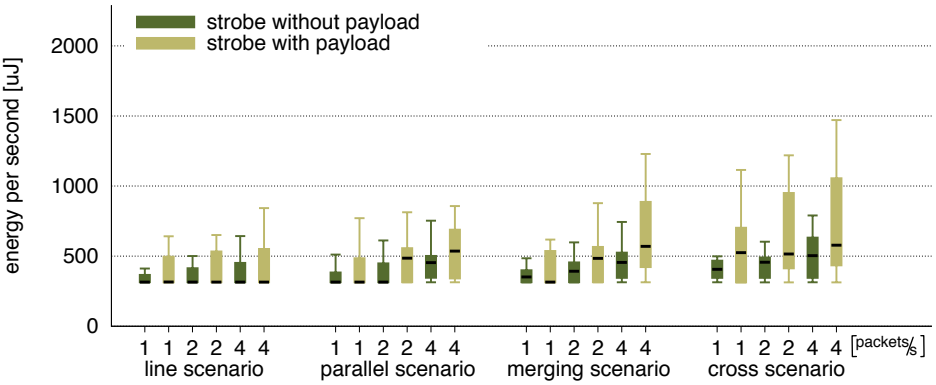


Figure 5.23: Energy per second of the noninvolved neighbor nodes with 40 bytes payload.

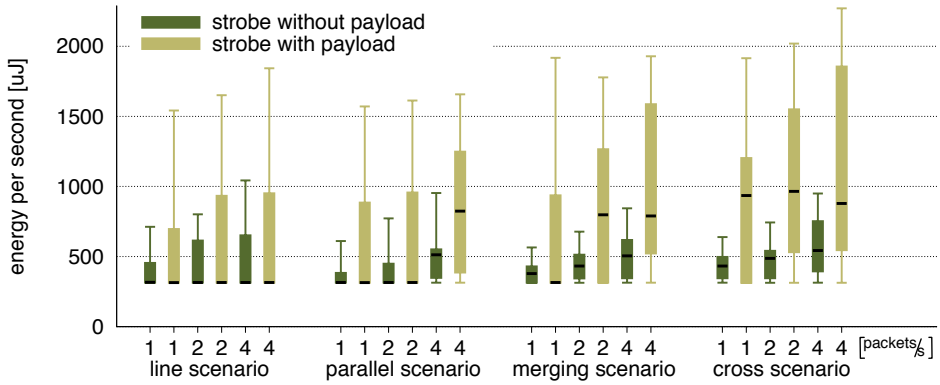


Figure 5.24: Energy per second of the noninvolved neighbor nodes with max. payload.

longer than the ones without payload. Therefore, the radio module requires a considerably longer listening interval to receive a beacon strobe including the payload. This results in a higher energy consumption.

Nevertheless, the overall energy required by beacon strobes including payload version is lower. When giving the same amount of energy to every node in a WSN, the WSNs using the beacon strobe version including payload will be longer able to deliver data to the sink. This is also the case if the used routing mechanism is able to replace nodes that ran out of power. For the final version of BEAM we use beacon strobes including payload for the following reasons:

1. **Energy consumption and network lifetime:** The overall energy consumed by involved and noninvolved is lower when using beacon strobes including payload. This increases network lifetime.

2. **Robustness against interferences:** Beacon strobes including payload are more robust against interferences. They require only two instead of four successful transmissions. Additionally, in case of bit error in the data frame, the receiver is able to just wait for the next beacon strobe.

3. **Code complexity:** The logic for beacon strobes including payload is less complex. It requires only two (strobe, ack) instead of four individual transmissions (strobe, ack, data, ack)to forward data to the next hop. This reduces code complexity and enhances protocol robustness.

5.2.4 Duty Cycle Evaluation

This subsection evaluates the longest and shortest useful duty cycle duration. The shortest reasonable duty cycle duration is required to achieve highest possible throughput at high traffic load. The longest duty cycle duration is used by all idle

nodes and to forward low traffic load. The less energy is required in idle mode, the longer is the network lifetime. Longer default duty cycle durations require less energy, but increase the required packet delivery time.

First, we search for the shortest reasonable duty cycle duration to achieve highest possible throughput. We use the *telosB small-scale testbed* line and parallel scenarios defined in Subsection 5.1.1 for these evaluations. The TARWIS event recording system is not able to handle this large amount of traffic. The application on the sender node creates 64 packets per second and sends them towards the network stack. A lot of them are dropped, due to insufficient bandwidth. Table 5.4 shows the used protocol configurations.

RDC protocol	BEAM with fix duty cycle durations
Strobe type	Including payload
BEAM-payload	40 bytes
Transmission delay	Yes
Transport protocol	UDP
End-to-end reliability	No
Used scenarios	Line, parallel

Table 5.4: Measurement setup for transmission delay optimization evaluations

The amount of wake-up periods executed per second is limited. BEAM performs two channel checks during a wake-up period to ensure a free channel (see Figure 5.8). Performing up to 256 wake-up periods per second, the radio module is able to handle a specific sleeping period after the two channel checks. At minimum duty cycle duration, one channel check after each other is executed. Technically the shortest time between two individual channel checks with the CC2420 radio module is 1.984 ms. This results in a maximum of 504 wake-up periods per second. Moreover, we measured the throughput without any sleeping period. In this case, the radio module stays in listening mode and waits for incoming packets. Figure 5.25 shows the successfully transmitted packets for different duty cycle durations. In both evaluated scenarios, throughput continuously increases for up to 256 wake-up periods per second. Then, throughput rate remains static in the line scenario and decreases in the parallel scenario. In the line scenario, the processing limit of the microcontroller is reached. It is not possible to forward more packets through the used μIP stack due to the processing load of the microcontroller. Decreasing of the throughput in the parallel scenario is caused by too much interference.

Using BEAM, the sender knows the current duty cycle duration of the receiver to calculate the retransmission delay. The corresponding retransmission delays for the *minimum duty cycle duration* and *no sleeping period* result in fast retransmissions. Fast retransmission can allocate so much radio channel time that no other node is able to forward any packets while the sender tries to forward all packets from the packet buffer. This often results in unbalanced traffic load between the two paths in the parallel scenario. In the final BEAM version we use a maximum

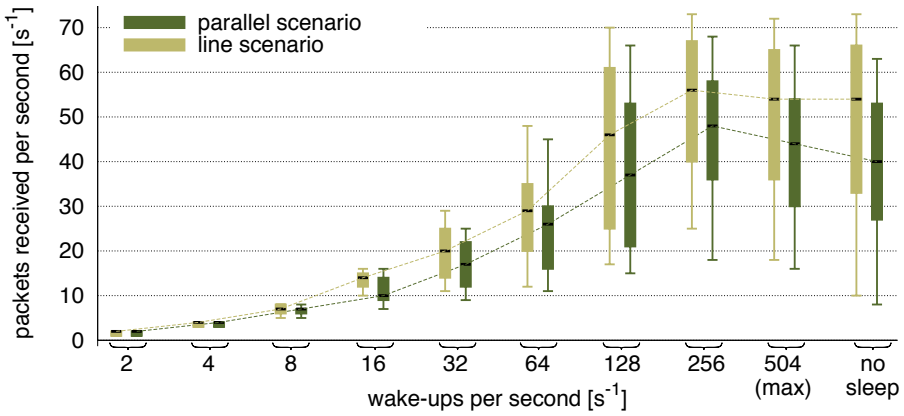


Figure 5.25: Throughput of different fixed duty cycle durations.

of 256 wake-up periods per second during high data rates.

To define a meaningful default duty cycle duration, we employed the WISEBED testbed. The consumed energy was recorded every second. We performed this measurement for different duty cycle durations for one hour during daytime (from 10 am to 11 am) and for one hour after might night. Figure 5.26 shows the measured energy consumption for different default duty cycle durations. More than 32 wake-up periods per second do not make any sense as the default duty cycle duration should be as long as possible to preserve energy. All measurements were made on channel 26 of the CC2420 at 2.480 GHz to prevent external interferences caused by IEEE 802.11 devices. As expected, the energy correlates with the amount of wake-up periods per second. The variation of the energy is caused by random external interference, which is interpreted as potential incoming frame. This causes a

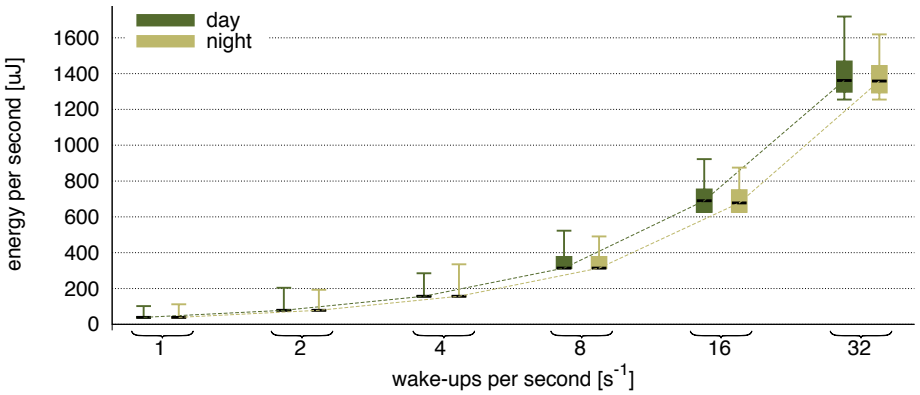


Figure 5.26: Energy consumption of different duty cycle durations.

longer listen cycle to wait for the next beacon strobe. The duty cycle duration and daytime does not show an impact on the percentage of executed wake-up periods detecting an ongoing transmission. In every scenario, around 2.5% of all channel checks detect an ongoing transmission. Especially, the measurements at 16 and 32 wake-up periods per second show that the external interferences are evenly distributed over time.

It is not possible to define a definitive meaningful default duty cycle duration for all scenarios in a WSN. Instead, the default duty cycle duration must be selected according to the requirements of the WSN application. Shorter default duty cycle durations are used for short packet delivery times, longer default duty cycle durations are used to enhance the lifetime of the WSN. For our further evaluations, we decide to use a default length of eight wake-up periods per second as ContikiMAC and XMAC are also additionally using eight wake-up periods per second in the Contiki default configuration.

5.2.5 Traffic Prediction

A major contribution of BEAM is its traffic prediction mechanism introduced in Section 4.1.3. It is used to adapt the current duty cycle duration to the expected traffic load. It replaces the traffic monitoring mechanism used by other existing adaptive MAC protocols to determine the current duty cycle duration. These protocols are only able to handle low traffic load and simple network topologies causing low interferences. Traffic monitoring is not able to handle inter-flow and inter-flow interferences generated by nodes that are sending a packet at the same time. We used the WISEBED testbed small-scale network topologies defined in Subsection 5.1.1 to compare BEAM using traffic prediction with BEAM using traffic monitoring. Table 5.5 shows the used protocol configurations. We used different traffic load values form 2 - 32 packets per second. Each experiment was executed twice, each time for around 100 minutes. The throughput achieved with traffic monitoring mechanism is shown in Figure 5.27. The results of the traffic prediction mechanisms are depicted in Figure 5.28. The lowest tested traffic load can be forwarded by using the default duty cycle duration of 125 ms. With increasing traffic load, shorter cycles are required to offer the required additional bandwidth. Moreover, the channel load increases with increasing traffic load. This increases the probabil-

RDC protocol	BEAM
Strobe type	Including payload
Default duty cycle duration	125 ms
BEAM-payload	40 bytes
Transmission delay	Yes
Transport protocol	UDP
End-to-end reliability	No

Table 5.5: Measurement setup for traffic prediction evaluations

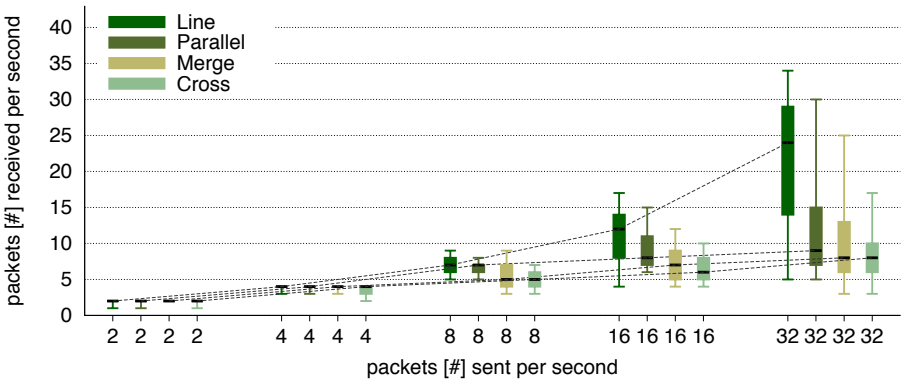


Figure 5.27: Throughput with traffic monitoring.

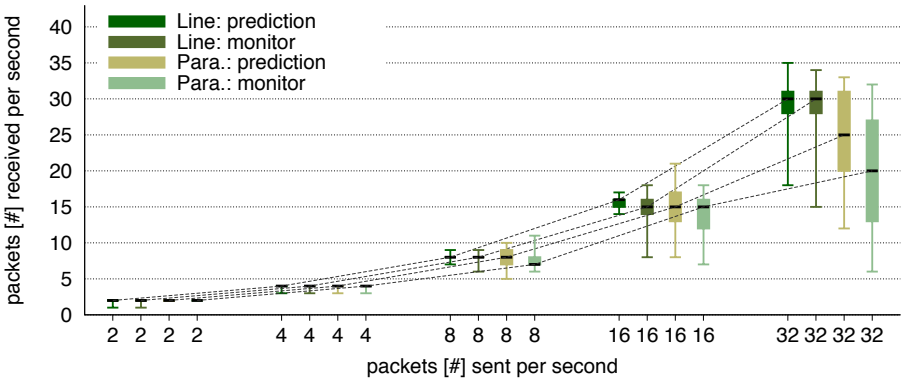


Figure 5.28: Throughput with traffic prediction.

ity that the channel is already busy if a sender tries to start sending beacon strobes, especially in the merging and cross scenario where two nodes try to send a packet to the same node. In addition, higher traffic load increases the probability of packet loss by internal interferences. These circumstances periodically cause temporary blocking of one or more links. During such blocking, some nodes are either not able to receive or to send a packet for a certain time period. The blocked sender nodes correctly detect congestion and extend the retransmission delay to reduce channel load. During this time, the sender must store additionally received packets in its packet buffer. Duty cycle handling of a blocked receiver node depends on the used adaptation mechanism.

By using traffic prediction, it is possible for the receiver to know that packets are pending. The receiver keeps short duty cycle durations. The first packet received after the blocking period then updates traffic prediction. The receiver can immediately update the duty cycle duration to an adequate length.

With traffic monitoring, the receiver detects decreasing traffic load during a blocking period. Therefore, the receiver adapts the duty cycle durations to lower traffic load. The unsteady traffic flow during congestion makes it difficult for traffic monitoring to adapt the duty cycle durations to a sufficient length.

- **Line scenario**: Traffic monitoring is forwarding up to eight packets per second without a significant end-to-end packet loss. Up to this traffic load, no adaptation of the duty cycle duration is required. At higher traffic load, traffic monitoring sometimes reduces duty cycle duration of a receiver node due to a blocked sender node. Traffic prediction is able to handle up to the fastest tested traffic load. At 32 packets per second, sometimes a sender is blocked too long. It must drop some packets then.

- **Parallel scenario**: In this scenario, BEAM started to adapt the duty cycle durations at a data rate of eight packets per second. Traffic monitoring is able to properly handle up to eight packets per second with adapted duty cycle durations. At higher throughput, we observed the following behavior with traffic monitoring. One of the two parallel flows is able to forward all generated packets, while the other flow must drop a large amount of generated packets. The flow with high packet loss uses the default duty cycle duration, while the other flow was able to adapt the duty cycle duration to the generated data rate. Sometimes, the flow with high packet loss is able to adapt the duty cycle durations for some seconds. Then, the resulting inter-flow interferences cause congestion, which decreases the data rate on one or both flows. The traffic monitoring mechanism recognizes the lower data rate and reduces the wake-up frequency to the default value for low data rates. Traffic monitoring is not able to detect the need to adapt the duty cycle durations at the flow with the high packet loss ratio. Traffic prediction is capable to keep short duty cycle durations on both paths, even if one is blocked for a certain time. Sometimes, a few packets have to be dropped during detected congestion.

- **Merging scenario**: Traffic monitoring is able to support up to 4 packets per second in this scenario. At higher data rates, one of the sender nodes detects congestion and reduces the data rate sometimes. The intermediate node shows most of the time a shortened duty cycle duration and is able to constantly receive packets from one of the senders. At 16 and 32 packets per second, it can happen that the sink node is interfered by transmission of the senders forwarding packets to the intermediate node and, therefore, reduces its duty cycle durations. The intermediate node then must drop a lot of packets, too. Traffic prediction is able to keep short duty cycle durations on both paths, even if one is blocked for a certain time. Nevertheless, it is not possible to successfully forward two flows with each 32 packets per second over a single hop. The resulting 64 packets per second exceed the performance of

the used network stack and the radio channel capacity reaches a critical limit at this data rate.

- **Cross scenario**: The cross scenario shows similar challenges such as the merging scenario. But the middle node must send and receive packets from two nodes. Traffic monitoring already shows some packet loss at four packets per second. Traffic prediction shows some packet loss at 16 and 32 packets per second.

Figure 5.29 shows the associated total packet loss. The lowest traffic rates of 2 and 4 packets per second are skipped to gain a better overview. All these skipped measurements show packet loss below 4%. The green area at the bottom of Figure 5.29 depicts the highest end-to-end packet loss ratio, which can be recovered by our end-to-end retransmissions. Higher packet loss can, usually, not be recovered. The hop-to-hop reliability mechanism already drops too many packets caused by too high data rate or interferences. Additional traffic caused by end-to-end reliability mechanisms would only increase the packet loss instead of reducing it.

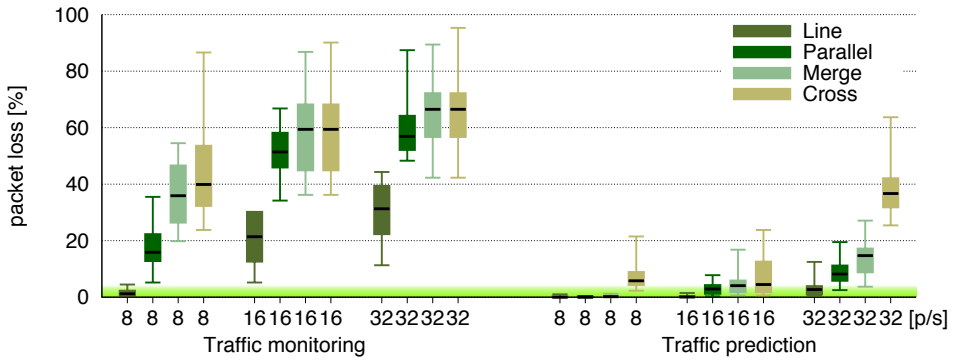


Figure 5.29: Packet loss at different traffic load values with traffic monitoring and prediction.

The results presented in this subsection clearly show that using traffic prediction performs significantly better than using traffic monitoring. Moreover, traffic prediction is able to detect the amount of pending packets during congestion. The final version of BEAM, therefore, uses traffic prediction to adapt the duty cycle duration to the current traffic flow.

5.2.6 Packet Aggregation

Packet aggregation requires the cooperation of BEAM and H2HR. It can minimize the amount of small packets in the network. Up to three UDP/μIP packets with a very short application payload can be aggregated into one BEAM frame. Packet

aggregation cannot be used with large payloads. A large packet only fits once into an IEEE 802.15.4 frame. The OMNeT++ simulator and the WISEBED testbed evaluation show that as soon as BEAM must decrease the duty cycle duration, traffic load is high enough to aggregate packets. At lower traffic load values, packet aggregation shows neither an impact on energy consumption nor on reliability. At high traffic load, packet aggregation can prevent or at least decrease congestion by reducing the number of concurrently forwarded packets. This reduces the overall energy consumption and the required number of transmissions.

Table 5.6 shows the used protocol properties to analyze the impact of packet aggregation optimization at high traffic load. We used the WISEBED testbed small-scale network topologies defined in Subsection 5.1.1. Each network topology is tested with two different traffic load values. In each setup, 10'000 packets are generated per UDP flow. Figure 5.30 shows the measured energy consumption. Packet aggregation reduces the required energy at high traffic load and in challenging network topologies by reducing the total amount packets to be forwarded. The reduction of forwarded packets lowers the total amount of required beacon strobe transmissions. Shortening of the beacon strobe transmissions additionally reduces interferences. Moreover, packet aggregation is able to reduce end-to-end retrans-

RDC protocol	BEAM
Strobe type	Including payload
BEAM-payload	37 bytes
Default duty duration	125 ms
Transmission delay	Yes
Transport protocol	UDP
End-to-end reliability	No

Table 5.6: Measurement setup for packet aggregation evaluations

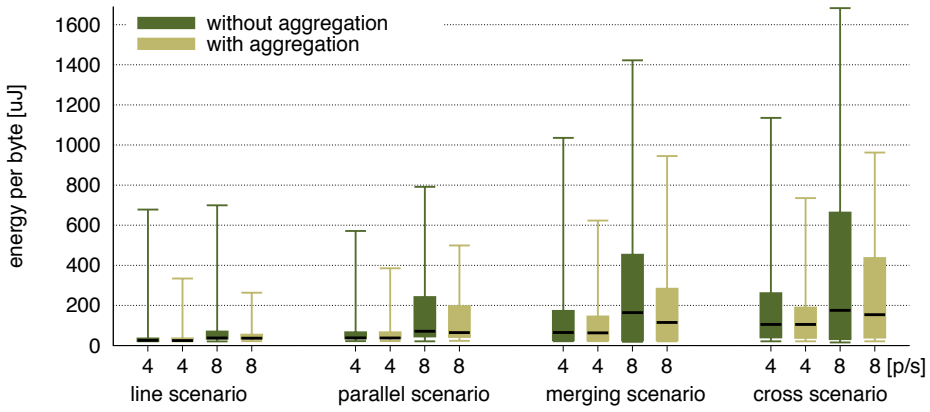


Figure 5.30: Energy consumption with and without packet aggregation.

missions and enhances throughput. Therefore, the final versions of BEAM and H2HR support packet aggregation to optimize performance during periods with high traffic load.

5.2.7 BEAM Protocol Optimization Techniques Summery

This subsection summarizes the selected BEAM protocol optimization techniques for the final version of BEAM. The decisions are based on the evaluation results, which have been presented above.

- **Acknowledgment mechanism:** We use the CC2420's hardware support for acknowledgments. Hardware acknowledgments significantly reduce energy consumption.
- **Beacon strobe transmission delay:** We use the introduced transmission delay optimization. It significantly lowers the energy consumption during low traffic load. With increasing traffic, it additionally reduces collisions and interferences.
- **Strobe design:** We use BEAM with beacon strobes including payload. It shows a slightly lower energy consumption. Nevertheless, it is more robust against interferences and code complexity is lower.
- **Duty cycle duration:** For comparison with ContikiMAC and XMAC, we use default duty cycle duration of eight wake-ups per second with these two protocols. BEAM can be configured for other application requirements by adapting the default duty cycle duration. If a long network lifetime is required, the duty cycle duration can be extended. If a low delay is required, the duty cycle duration can be reduced. To offer the highest possible bandwidth we use 256 cycles per second.
- **Traffic prediction:** We use traffic prediction to adapt duty cycle durations. Traffic prediction performs significantly better than traffic monitoring.
- **Packet aggregation:** We use packet aggregation. Small packets can be aggregated to reduce the total amount of transmitted packets. A lower packet count reduces interference caused by reducing the required beacon strobes. The lower packet count additionally reduces the energy consumption.

The following section makes use of the presented final BEAM version to evaluate the H2HR protocol.

5.3 Reliability Evaluations of H2HR and UDP-E2E

First, this section presents the evaluation of the H2HR hop-to-hop reliability mechanisms. H2HR is part of the link layer and located on top of BEAM. H2HR uses

BEAM transmission reports introduced in Section 4.2.2 to detect lost packets and retransmits them with an appropriate retransmission delay. The applied retransmission delays have a significant impact on the success of retransmission and congestion handling. Generally, a retransmission delay should be as short as possible to support steady and short packet delivery times. Longer random retransmission delays are required during congestion caused by high channel load. Too early executed retransmission at high channel load may fail and result in additional interferences. This results in a critical channel load and decreases the traffic flow even more.

We developed the **congestion detection mechanism** introduced in Section 4.2.2 to determine an appropriate retransmission delay. This mechanism uses the **three information sources** (number of retransmissions, transmission report and neighbor table introduced in Section 4.2.2) to estimate the current congestion state and to calculate an appropriate retransmission delay. The evaluation of an appropriate weighting for three the individual information sources was performed in two steps. First, we used the OMNeT++ simulator for extensive tests to compare all possible weights of the three introduced information sources.

In a second step, we evaluated the most promising weights within a real world testbed. The evaluation shows that, without delaying the retransmission during congestion, sometimes all transmissions fail in the affected area for a longer time period. Usually, this results in high packet loss caused by buffer overflows. Adding an appropriate retransmission delay enables continuous traffic flow. In addition, it enables BEAM to frequently update the neighbor table information to provide up-to-date network information. Some evaluated scenarios generated too many packets for the used WSN network topology. In this case, the retransmission delay mechanism resulted in a backpressure mechanism, which dropped packets at the generating node before they were sent. To enhance the backpressure effect, H2HR additionally applies an appropriate delay to the first transmission attempt of the next packet. In the following subsections, we evaluate how the three information sources can be used by H2HR to optimize the congestion detection mechanism for calculating an appropriate transmission delay. Afterwards, we evaluate the impact of the end-to-end reliability protocols UDP-E2E and TCP on the packet loss ratio. The evaluations show that the performance of end-to-end reliability protocols is limited. Under low traffic load, they are able to recover over 98% of the dropped packets. At higher traffic load, they may generate more traffic loss.

5.3.1 H2HR Simulation Evaluation

In a first step of finding the best usage of the three information sources, we implemented the streaming scenario shown in Figure 5.4a in the OMNeT++ simulator. We used the graphical runtime environment to determine a traffic pattern that creates serious congestion. Using this traffic pattern, we created an evaluation environment featuring 10'000 runs with different random seeds. This offers the opportunity to evaluate all different combinations and weights of the different in-

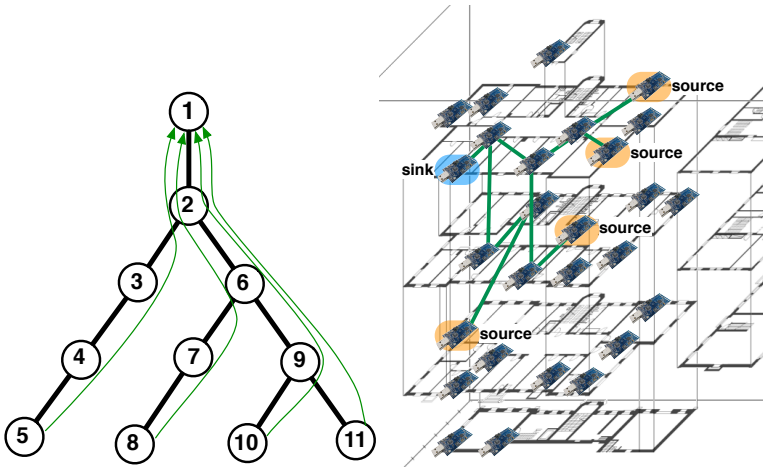
formation sources with exactly the same traffic timings and resulting interference patterns. The simulation results suggest that all of the three information sources have to be taken into account to detect congestion most reliably and rapidly. At low congestion no retransmission delay needs to be added. With increasing congestion, the retransmission delay must be quickly increased. The simulations additionally indicate that, contrary to one's expectations, an end-to-end reliability mechanism can decrease the packet-delivery ratio at high traffic load. Above a certain traffic load, the packet loss is higher with UDP-E2E than without, due to intra-flow interferences generated by negative acknowledgments. At lower traffic load, UDP-E2E increases the packet-delivery ratio to 99.8%.

Based on the simulation results, we evaluate the most promising weight combinations in the WISEBED testbed. The next subsection presents the results of the WISEBED testbed evaluation. The finally used weights are described in the H2HR protocol description in Section 4.2.

5.3.2 H2HR Real Word Evaluation

In the following WISEBED experiments, we determine the best usage of the three available information sources introduced in Section 4.2.2. Figure 5.31 shows the used large-scale streaming scenario defined in Subsection 5.1.2. The streaming scenario generates more internal inferences than the small-scale scenarios defined in Subsection 5.1.1. We used different combinations of the three information sources to evaluate the impact of the individual information source on reliability and energy consumption.

Table 5.7 shows the different used combinations of the information sources for the WISEBED testbed evaluations.



(a) Traffic flow. **(b)** WISEBED testbed nodes.

Figure 5.31: WISEBED testbed streaming scenario used for H2HR transmission delay evaluation.

One combination uses only the failed number of transmission attempts, which represents the information source commonly used by other hop-top-hop retransmission mechanisms. Two combinations additionally use either the BEAM neighbor table or the BEAM transmission report as second information source. One combination uses all three information sources. The shortcut defined in the first column of Table 5.7 called ID refers to corresponding combination. On top of μIP, we use UDP. UDP does not support any end-to-end retransmission functionality. Therefore, we use UDP-E2E on top of UDP to enable end-to-end reliability.

ID	Description	Number of retransmissions	Transmission report	Neighbor table
NR	Number of retransmissions (NR)	x		
NR/RL	Additional delay for receiver load (RL)	x		x
NR/CL	Additional delay for channel load (CL)	x	x	
All	Using all sources	x	x	x

Table 5.7: Information sources used for the WISEBED testbed evaluations.

Packet Loss

Each of the combinations shown in Table 5.7 has been tested with a **moderate traffic load** of 0.5 packets per second and a **high traffic load** of 2 packets per second on each of the four paths in the large-scale streaming scenario (see Figure 5.31). The channel load is rather low for this traffic load. BEAM is able to efficiently forward the packets at this traffic load. UDP-E2E can send a negative acknowledgment without overloading the radio channel. A data rate of 2 packets per second results in a traffic load that generates a critical level of inter-flow interference within the used network topology. Figure 5.32 shows the resulting end-to-end packet loss for both traffic load values. Both traffic load values are evaluated with retransmission limits of 3, 10 and 20 attempts (*RTX Att.*). For each measurement, we used 15'000 packets. There is no noticeable difference between a limit of 10 and 20 retransmissions, as less than 0.2% of the transmissions require more than 10 retransmissions. With a limit of three retransmissions, the packets are dropped too quickly during periods of congestion.

The H2HR hop-to-hop reliability mechanism is able to deliver over 99.9% of the generated packets at **moderate traffic load** and at least 10 retransmission attempts. Only isolated packets are missing. The additional use of the *transmission report* and *neighbor table* does not provide a noticeable improvement compared to a normal random back-off mechanism. However, it reduces the end-to-end packet loss at **high traffic load**. We achieved the best results by using all sources at the critical traffic load.

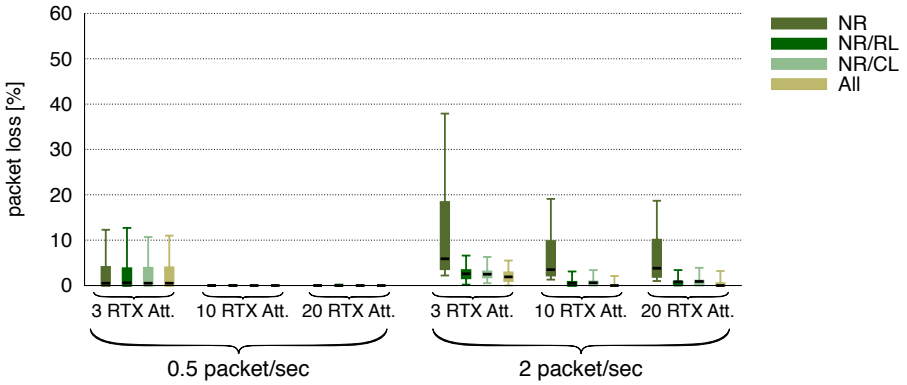


Figure 5.32: End-to-end packet loss ratio at different traffic load values.

Figure 5.33 depicts how many of the created data and acknowledgment packets are dropped between sender and sink. H2HR is able to forward over 99.5% of all packets towards the sink, at **moderate traffic load** and with at least 10 retransmission attempts. Packet loss is caused by a packet overflow during congestion. Most of the packets are dropped on the last hop, on which all four flows have to be handled concurrently. Here, the node receives more packets than it can forward to the next node. This results in an overflow of the packet buffer. The packet loss that occurs when having a limit of three retransmissions has a different reason. Here, most of the packets are dropped due to having exceeded the retransmission limit. The packet buffer does not overflow.

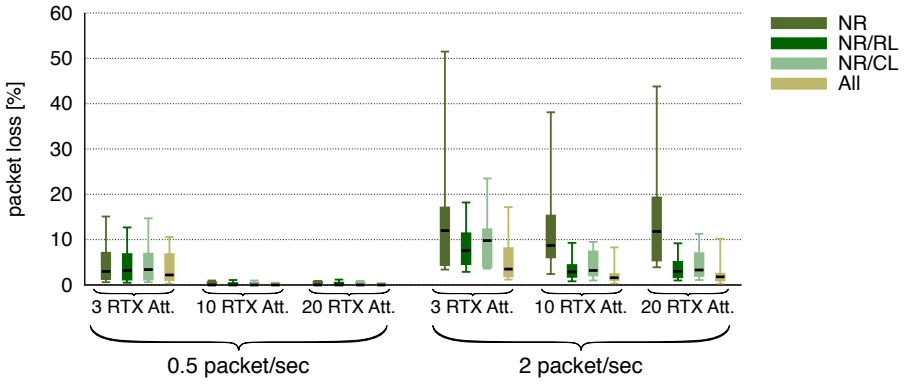


Figure 5.33: Packets dropped at different traffic load values.

At **high traffic load** significantly more and longer congestion periods have to be handled. The maximum number of applied retransmissions has significant influence to congestion handling. With a high limit of 10 or 20 retransmissions, more

than 60% of the dropped packets are dropped by the node that generated the packet. This reduces the total amount of packets in the network to a reasonable limit. With an overloading traffic load of four packets per second, over 90% of the dropped packets are dropped directly by the node that generated the packets. This is basically caused by the retransmission delay calculated by H2HR and packet loss caused by bit errors. Packets dropped by nodes generating them reduces the buffer overflows inside the network during congestion periods. With a low retransmission limit of three retransmissions, clearly more packets are injected into the network. Their packets are dropped during congestion periods by exceeding the retransmission limit. This reduces the total amount of concurrently forwarded packets during congestion periods and, therefore, it additionally reduces the congestion. The overall packet loss with a low retransmission limit is clearly higher than with sufficient retransmissions.

Every negative acknowledgment sent by UDP-E2E produces intra-flow interferences. This may cause additional packet loss if the traffic load is already reaching a critical level.

Energy Consumption

Figure 5.34 shows the energy consumption of the different measurement scenarios. The packet loss ratio shows two opposite effects with respect to energy. First, a minimized end-to-end packet loss ratio requires more energy for the excessive retransmission of lost packets. Second, every received packet reduces the energy percentage per successfully transmitted byte. With up to three end-to-end retransmissions per packet, many unsuccessful end-to-end retransmissions are skipped. It shows that reducing the packet loss ratio to the possible minimum requires much additional energy.

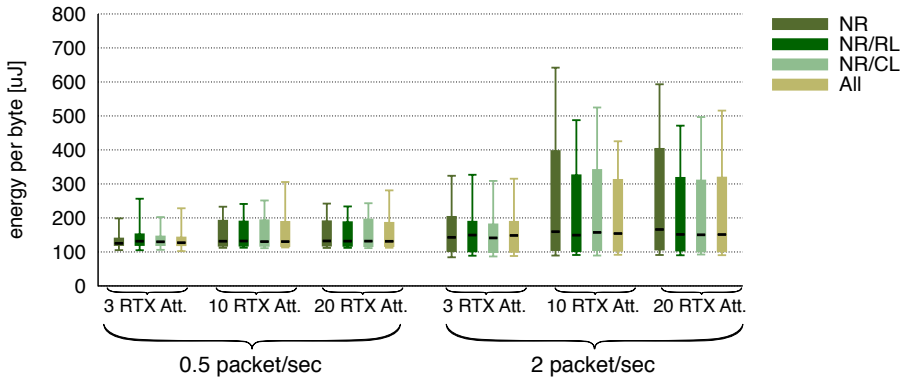


Figure 5.34: Energy per successfully transmitted byte.

The additional use of the *transmission report* and *neighbor table* information slightly reduces energy consumption per transmitted byte at high traffic load. The combi-

nation of *transmission report* and *neighbor table* information does not show an observable additional improvement in energy consumption.

UDP-E2E is responsible for a considerable part of the used energy. The negative acknowledgments require plenty of additional beacon strobes and shorter duty cycle durations to enable bi-directional traffic. Too many negative acknowledgments may cause even more packet loss instead of reducing it. The following subsection shows the impact of end-to-end retransmission mechanisms on packet loss.

5.3.3 End-to-end versus Hop-to-Hop Reliability

End-to-end reliability mechanisms ensure nearly 100% delivery ratio at low and moderate traffic load. In our measurements, end-to-end packet loss was below 0.2% for the applied traffic load values. Our network stack offers two different end-to-end reliability mechanisms introduced in Subsection 5.1.5. One offered reliability mechanism is UDP-E2E, the other is TCP. The two mechanisms significantly differ. UDP-E2E sends negative acknowledgments for missed packets, while TCP is using positive acknowledgments for successfully received packets. Sending the acknowledgment at an appropriate time is a challenging task for an end-to-end reliability protocol. TCP features a sophisticated mechanism for sending acknowledgments. UDP-E2E using excessive configuration sends negative acknowledgments whenever a lost packet has been detected. UDP-E2E using standard configuration also sends a negative acknowledgment if a lost packet has been detected, but waits with sending an additional negative acknowledgment either until the missed data packet has arrived or 1.5 times the average RTT.

Figure 5.35 compares end-to-end packet loss of TCP with UDP-E2E in standard and excessive configuration. Moreover, Figure 5.35 shows the end-to-end packet loss with UDP without any end-to-end reliability mechanism (*No-E2E*). All experiment setups are using H2HR with 10 retransmission attempts. *TCP* clearly

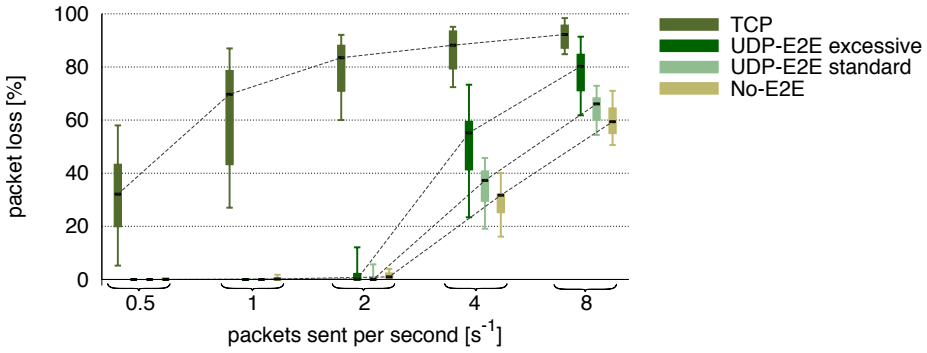


Figure 5.35: End-to-end packet loss.

shows the worst performance. The excessive transmission of acknowledgments in the opposite direction overloads the channel by too many intra-flow interferences. *UDP-E2E excessive* and *UDP-E2E standard* are able to successfully deliver 100% of the packets up to a data rate of one packet per second. At a generated data rate of 2 packets per second per node, the channel shows a critical channel load. Packet loss with *UDP-E2E excessive* is clearly higher than for *UDP-E2E standard*. *No-E2E* does not perform any end-to-end retransmission. On one hand, *No-E2E* shows the lowest packet loss for load above the critical value. On the other hand, *No-E2E* shows packet loss ratios of 0 - 1% at low and moderate traffic load. For the final protocol stack, we, therefore, use *UDP-E2E standard*. It generally achieves a delivery ratio of 100% at low data rates. But in a few measurements, some of the 15'000 generated packets are missing. This can be caused by an error in the TARWIS event recording system or by high external interferences. *No-E2E* cannot be used as it shows up to 1% packet loss at low and moderate traffic load.

5.3.4 Summary Reliability Evaluations of H2HR and UDP-E2E

We use the information of the *transmission report* and *neighbor table* to optimize hop-to-hop reliability in the final version of H2HR. It enables detecting occurring congestion as well as calculating an appropriate retransmission delay. Reliability functions generally increase the energy requirements, as additional retransmissions are required for dropped packets. The information sources used by H2HR decrease the additionally required energy by reducing the needed retransmissions during congestion periods and while increasing the delivery ratio. H2HR reduces the required end-to-end retransmissions, which additionally decreases energy consumption and congestion. Our evaluations show that a hop-to-hop reliability mechanism forcing local retransmissions clearly performs better in terms of reliability with lower energy requirements than when using end-to-end retransmissions. End-to-end reliability mechanisms are not suitable for high traffic load. At high traffic load, end-to-end reliability mechanisms can even increase packet loss instead of reducing it. However, end-to-end reliability mechanisms combined with hop-to-hop reliability mechanisms are able to offer an nearly 100% end-to-end delivery ratio at **low traffic load** values. Hop-to-hop reliability mechanisms without end-to-end reliability support only achieve an end-to-end delivery ratio of 99% at low traffic load values.

5.4 Impact of FEC Codes to Energy Efficient WSNs

In this section, we take a closer look to the impact of FEC codes on the energy efficiency and reliability of real world WSN stacks. We evaluate the impact of different radio modules, different SNR levels and interferences to the performance of the FEC codes. For example, the DSSS mechanism of the CC2420 is significantly more robust against interferences than the OOK modulation of the bit/byte-oriented

radio modules. Moreover, we integrate FEC codes to our protocols to evaluate the impact of FEC codes to the energy efficiency and reliability performance of a real world WSN stack. BEAM would support FEC codes if they are able to either reduce the energy consumption or improve reliability.

The following subsections evaluate step-by-step two FEC codes featuring different recovery options. In a first step, Subsection 5.4.1 determines the additional energy required for encoding, decoding and transmission of parity bytes. Subsection 5.4.2 evaluates packet loss of FEC codes within a simple real world testbed featuring different network conditions. This enables a basic estimation of the recovery potential of FEC codes. Subsection 5.4.3 analyses the impact of the FEC codes in an energy efficient and reliable real world WSN stack. Therefore, we integrated both FEC codes into BEAM to support energy efficiency and provide detection and retransmission of lost packets. Finally, Subsection 5.4.4 measures how the throughput is affected by FEC codes.

5.4.1 Encoding and Decoding Payload

This subsection measures the energy required for encoding and decoding data packets by different FEC implementations on a telosB node. Using FEC codes, every packet must be encoded before it can be sent. Decoding is only required if the CRC check indicates a bit error in the received packet. The authors of [9] and [41] evaluated the time required by different FEC codes to encode and decode data. They evaluated BCH code, Hamming code, repetition code [61] and Double Error Correcting, Triple Error Detecting (DETECT) code [32]. The presented evaluations are made by the use of the ECC library implemented in Scatterweb [31] on MSB430 nodes [8]. The authors additionally integrated their ECC library into Contiki [40]. For our measurements, we use Hamming(12,8) [33] and Reed-Solomon(255, 225) [70] with 66 bytes data payload. These two FEC codes are often used in WSN applications (e.g., [53] and [14]). Hamming(12,8) represents an FEC code with low energy requirements but low recovery potential. Reed-Solomon(255, 225) represents an FEC code with a high recovery potential. Unfortunately, the complex algorithms used by for Reed-Solomon(255, 225) result in significantly longer execution time and a higher energy usage than the Hamming(12,8) code.

To measure the required energy, we use again the RIGOL multimeter connected to a telosB node. Contiki puts the micro-controller as often as possible into sleep mode. During the entire FEC calculation period, the micro-controller is operating at full power. The difference between the energy levels represents the required energy for encoding and decoding. The measured energy consumption for one second of FEC code calculations is 16.72 mJ with a standard deviation of 0.1023 mJ. The energy consumption for one second during idle mode depends on the used operating system, which is responsible of handling the duty cycles of the microcontroller. A microcontroller running Contiki requires 1.43 mJ for one second during idle mode (Section 5.1.4). This results in an additional energy requirement of 15.29

mJ for one second during FEC calculations.

First, we measured encoding and decoding times required by the Hamming code implementation used by [9] and [41]. This implementation requires 102.5 ms to encode 66 bytes payload with Hamming(12,8). This is 79 times longer than the results published in [53]. The ECC library implementation used by [9] and [41] is inefficient and should not be used to retrieve meaningful results. The implementation used by [53] is not available. Therefore, we made a new implementation of a Hamming(12,8) code. Our own Hamming(12,8) code implementation requires only 30% more calculation time than the implementation from [53]. The difference in calculation time is most likely caused by a different implementation strategy, requiring less computation power on a telosB node. The performance difference of both Hamming(12,8) codes on a telosB node are shown in Figure 5.36. *Own implementation* refers to our implementation, while *reference implementation* refers to [53].

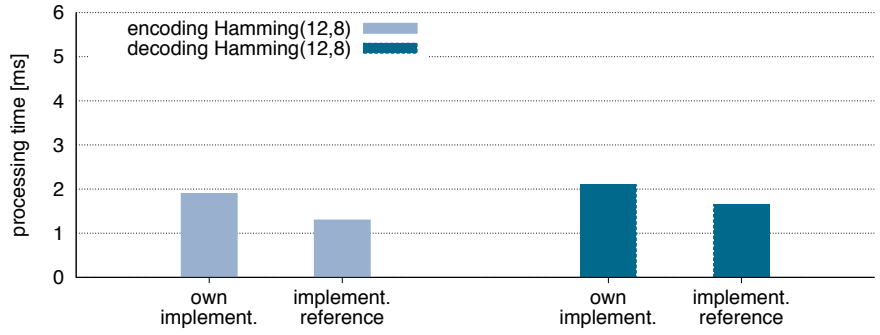


Figure 5.36: Encoding and decoding time of Hamming(12,8) for 66 bytes payload.

The available ECC library provided by [40] does not include a Reed-Solomon implementation. Therefore, we created our own Reed-Solomon(255, 225) implementation and compared its performance to the Reed-Solomon(255, 225) implementation of [53]. Figure 5.37 shows the encoding and decoding time for 66 bytes payload of both versions. Our own implementation is 55% faster in encoding and up to 6 times faster in decoding. The Reed-Solomon decoding time from the reference implementation depends on the occurred byte errors. The reference implementation requires longer time to decode a code word with 15 erroneous bytes than a code word without error. Our implementation requires always the same time for decoding regardless of the amount of byte errors in the code word.

Table 5.8 shows the additional energy required by the microcontroller to encode and decode data with our FEC code implementations on a telosB node with Contiki. The presented values do not include the additional energy costs required by the radio module to transmit the parity bits. With the telosB node, sending and receiving parity bits requires additionally 3.66 μJ/Byte (see Section 3.1). We use

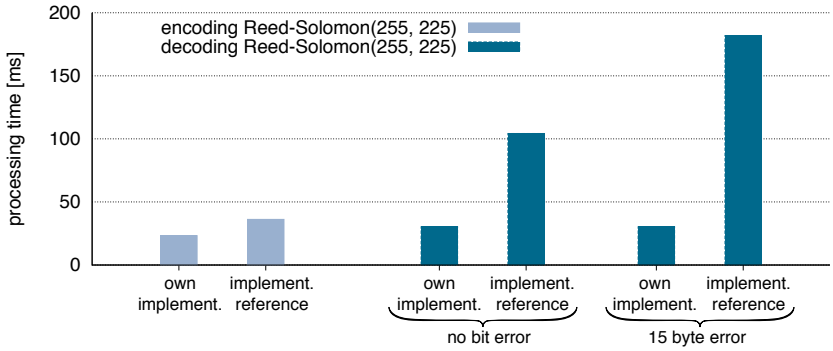


Figure 5.37: Encoding and decoding time of Reed-Solomon(255,225).

both FEC implementations in the following evaluations.

FEC code	Encoding [μJ/Byte]	Decoding [μJ/Byte]
Hamming(12,8)	0.147	0.162
Reed-Solomon(255, 225)	1.80	2.34

Table 5.8: FEC calculation costs per byte on a telosB running Contiki

Transmission Costs

Next, we measured the energy required to forward one single packet with a payload of 66 bytes for both FEC codes. Moreover, we measured the energy required by a hybrid packet recovery mechanism, combining ARQ with FEC to forward a packet with 66 bytes payload. We used the line scenario in the evaluation setup to measure the electrical current (see Figure 5.6). The individual telosB nodes are placed in a distance of 20 cm. This reduces the bit error ratio to below 1%. On the link layer, we use nullRDC and nullMAC without any duty cycle mechanism. In a first step of the measurement, we recorded the energy required by the sender to encode and transmit one single packet with 66 bytes payload. In a next step, we recorded the energy required by the receiver to receive a packet and decode the packet if required. The energy recordings are split into those two steps as only one sensor node can be connected to the RIGOL multimeter.

First, we measured the energy costs in case of an **error free** transmission. In these experiments the protocols using FEC codes encode the payload and forward the frame including the parity information to the receiver. The CRC check of the CC2420 on the receiver side indicates the error free reception of the packet. Therefore, the receiver can skip the FEC decoding and can just remove the parity information. With the ARQ protocol, the sender transmits a packet and the receiver sends back an acknowledgment.

Second, we measured the energy costs in case of a **single bit error**, which is recoverable by the FEC code. To simulate the erroneous transmissions, we intentionally added a one bit error to the encoded payload on the sender side. The receivers using FEC codes are able to recover this single bit error by decoding and processing the parity information. With the ARQ protocol, the sender has to retransmit the erroneous frame.

Figure 5.38 shows results of the energy measurements in these scenarios. The energy consumed by the radio module depends on the packet length. The parity information of the Hamming(12,8) (HAM) and Reed-Solomon(255,225) (RS) encoded payload requires additional energy. The hardware acknowledgment of the ARQ mechanism requires roughly the same additional energy as the 30 bytes parity data of the Reed-Solomon(255,225) code. ARQ is the most energy efficient mechanism in case of an error free transmission, even though an acknowledgment is sent. In case of a single bit error, FEC codes have to decode the parity information and ARQ must perform an additional transmission. In this case, the Hamming(12,8) code is more energy efficient than ARQ.

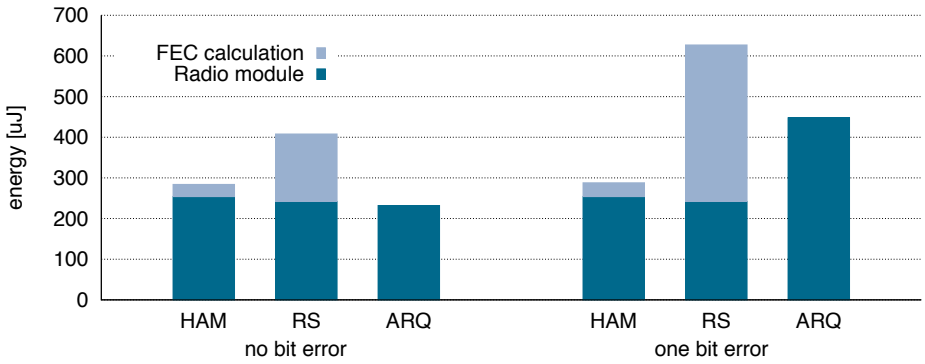


Figure 5.38: Energy to forward one packet with different reliability techniques.

If more than two retransmissions are required, then a successful recovering by Reed-Solomon(255,225) is more energy efficient than ARQ. These results do not allow a general conclusion about their energy efficiency within a real WSN. On one hand, only slightly damaged packets can be recovered by Hamming(12,8). Section 5.4.2 analyses how many of the received packets can be successfully recovered by the used FEC codes. On the other hand, FEC codes have to be integrated into an energy efficient network stack supporting local retransmissions. These influences and the impact of adaptive FEC algorithms are evaluated in Subsection 5.4.3. Before that, we evaluate the recovery potential of FEC codes at different network conditions in the next subsection.

5.4.2 Recovery Potential of FEC Codes

This subsection shows the recovery potential of our FEC code implementation under different network conditions. The following conditions affect the bit error ratio and the amount of detected frame starts:

- **Process gain:** Different radio modules feature different process gains. High process gain decreases the bit error ratio and increases the probability of detecting a frame start.
- **Received data transmission signal strength**. The received signal strength of the data transmission depends on environment characteristics such as distance and obstacles. The higher the received signal strength is the higher is the probability of a successful transmission.
- **Internal interferences:** Ongoing traffic increases internal interferences. A high traffic load increases the bit error ratio and decreases the probability of detecting a frame start.
- **SNR:** The combination of the received signal strength with the ongoing internal and external interferences determines the SNR. High SNR decreases the bit error ratio and increases the probability of detecting a frame start.

Testbed Setup

We use telosB and MSB430 nodes to evaluate the impact of the process gain. The radio module of the telosB shows a significantly higher process gain than the radio module of the MSB430 node. Both nodes types are connected to a notebook. The notebook acts as recording device to analyze the occurred bit errors. The FEC codes are implemented in Contiki. On the link layer, we use nullRDC and nullMAC. Figure 5.39 shows the testbed setup. For high SNR, nodes 1 and 2 are placed in a distance of 30 cm. For the low SNR values, we used nodes 1 and 3. They are placed in two rooms on the same floor level separated by a corridor. We placed node 4 on the upper floor level to generate **internal interferences**. The transmissions by node 4 are hard to detect by receiver node 1. The third node constantly transmits packets after a short back-off including a channel check. We divide the transmitted packets into four groups to analyze the recovery potential:

1. **Recovery success:** The occurred bit errors were recovered by the evaluated FEC codes. Packets of this group would be lost without FEC.
2. **Recovery fail:** The received packet contains too many bit errors to be recovered by the evaluated FEC code. An FEC code with a higher recovery capacity might be able to correct the bit errors.
3. **No bit errors:** Packets received without any bit error. FEC codes do not have an impact on this group.

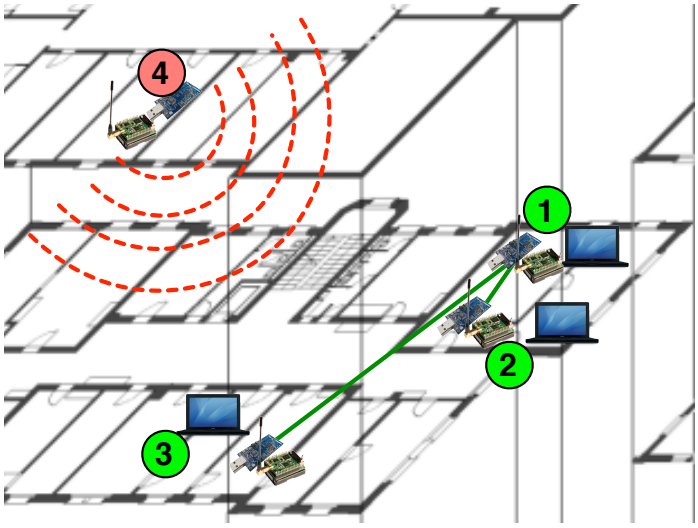


Figure 5.39: WISEBED testbed FEC evaluation setup.

4. **Missed frame start:** These packets could not be detected by the radio module due to bit errors in the frames start byte. FEC codes do not have an effect on this group.

Bit Errors Caused by External Interferences

Figure 5.40 shows the amount of received packets without internal interferences. External interferences are caused by surrounding electrical devices. Especially, devices operating in the 2.4 GHz ISM band may cause significant external interfer-

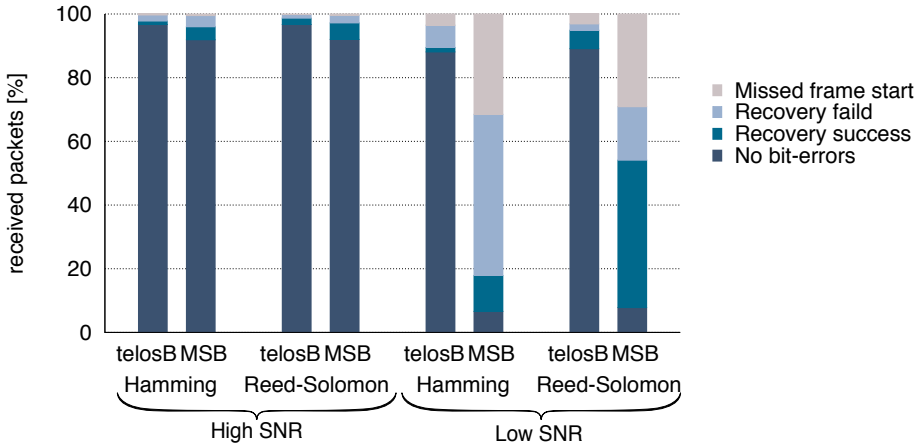


Figure 5.40: Different kinds of bit errors with external interferences.

ences in these experiments. For each result, 50'000 packets have been transmitted from node 2 to node 1 for high SNR and from 2 to node 1 for low SNR. In case of high SNR, over 99% of the packets have been received without any bit error. Less than 1% packets are lost. With low SNR, the higher process gain of the telosB node results in a significantly lower bit error ratio than for the MSB430 nodes. The telosB node was able to receive more packets without any bit errors than the MSB node was able to detect. Process gain enhances the probability of detecting a frame start. In all scenarios, Reed-Solomon(255,225) is able to repair clearly more erroneous packets than Hamming(12,8). Less than 6% of the packets transmitted by the telosB nodes are successfully recovered by an FEC code.

Bit Errors Caused by External and Internal Interferences

Figure 5.41 shows the measured amount of received packets including internal interferences. The original transmitter sends a total of 50'000 packets to the receiver. A packet is only transmitted if a free channel is detected. Otherwise, the transmission is postponed. The internal interferences show a noticeable effect to all measurement setups. The high process gain of the CC2420 radio module increases the amount of detected packets and decreases the bit error ratio. Over 75% of the received packets are received without any bit error. The Reed-Solomon(255,225) FEC code is able to repair more than half of the packets received with bit errors. Hamming(12,8) only repairs a small amount of the corrupted packets.

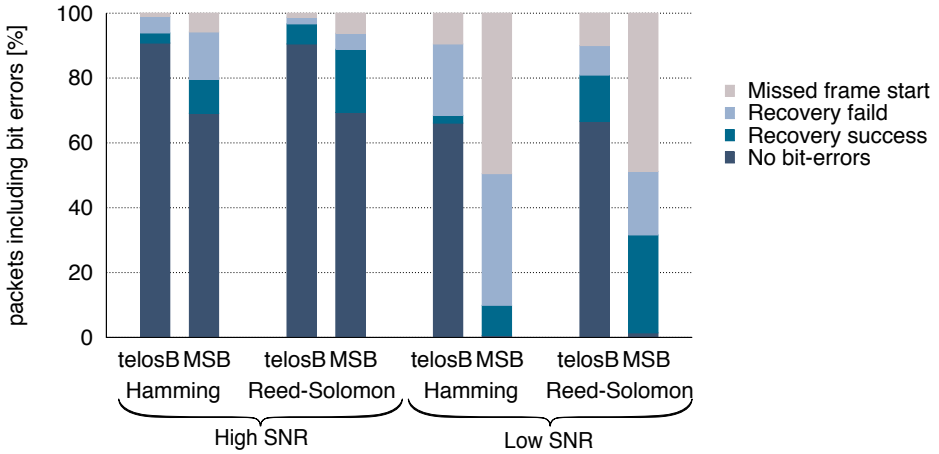


Figure 5.41: Different kinds of bit errors with external and internal interferences.

Conclusions of Recovery Potential of FEC Codes

The results presented in this subsection show that the process gain has a higher impact on the packet loss ratio than FEC codes. The radio module generates the

process gain during the carrier wave modulation. Therefore, it enhances the probability of a frame start detection. A high process gain is able to receive more error free packets than a radio module with low process gain is able to detect. Furthermore, no additional calculations are required by the microcontroller. Therefore, process gain requires **no additional calculation** time and **less energy** than FEC codes. The removal of the decoding delay significantly reduces the delay for sending an acknowledgment. In Subsection 5.2.1, we showed that a short acknowledgment response time is required to minimize the energy required for the periodic channel checks. An outcome of these experiments is the confirmation that packet oriented radio modules are the significantly better choice than bit/byte oriented radio modules concerning reliability and energy efficiency.

The results presented in this subsection additionally show that FEC codes could make sense for bit/byte oriented radio modules such as the CC1020 used in [41]. The evaluated byte oriented radio module CC1020 shows a clearly higher bit error ratio at low SNR values than the packet oriented radio module CC2420. This is caused by the simple OOK modulation technique used by bit/byte oriented radio modules. Furthermore, bit/byte oriented radio modules require several times more energy and time than packet oriented radio modules to forward the same amount of data (see Section 3.1). Therefore, a corrupted packet recovered by a FEC code on sensor node using a bit/byte oriented radio module preserves clearly more energy than a packet recovered on sensor node using a packet oriented radio module.

Nevertheless, all evaluated FEC codes were able to reduce the required numbers of retransmission attempts for all types of radio modules. In the following subsections, we evaluate the impact of FEC codes on packet oriented radio modules in more detail. Furthermore, we analyze if the energy preserved by less transmission attempts is sufficient to compensate the additional energy costs caused by the FEC calculations.

5.4.3 Evaluating FEC Codes in an Energy Efficient WSN Stack

This subsection evaluates the impact of FEC codes to a real world network stack supporting energy efficiency and reliability. Therefore, we added FEC support to BEAM and H2HR as introduced in Subsection 4.1.5. This enables evaluating our FEC code implementations including energy efficiency and an ARQ mechanism.

Testbed Setup

Table 5.9 defines and explains the acronyms for the evaluated FEC code implementations considered in this subsection. Using the static FEC implementations (*RS static*, *HAM static*), every transmitted packet is encoded. Using the adaptive FEC implementations (*RS adaptive*, *HAM adaptive*), introduced in Subsection 4.1.5, the FEC encoding is always applied after a transmission containing one or more bit errors. After a transmission without a single bit error the FEC encoding is disabled until the next bit error is detected. Moreover, we compare the different

FEC implementations to BEAM and H2HR without FEC support (*No FEC*).

Acronym	**FEC code**		**Application**	
	Hamming(12,8)	Reed-Solom(255,225)	Static	Adaptive
RS static		x	x	
HAM static	x		x	
RS adaptive		x		x
HAM adaptive	x			x
No FEC				

Table 5.9: Evaluated protocol versions with static, adaptive and without FEC support.

Table 5.10 shows the used WISEBED testbed to analyze the energy consumption and data delivery reliability performance of different FEC versions.

Testbed	WISEBED
Scenario	Stream (see Figure 5.5)
RDC protocol	BEAM with FEC support
Unencrypted data payload	66 bytes
Payload with Hamming(12,8)	99 bytes
Payload with Reed-Solomon(255,225)	96 bytes
Traffic load (low)	0.5 packets/sec
Traffic load (high)	2 packets/sec
Packets per experiment	10'000
Transport protocol	UDP
End-to-end reliability	UPD-E2E

Table 5.10: Measurement setup for BEAM with different FEC versions.

Performance Evaluation of Data Delivery Reliability

Figure 5.42 shows the ETX count measured for the different protocols. At 0.5 packets per second more than 50% of the transmissions were successfully at the first attempt. At this data rate, H2HR is able to achieve a delivery ratio of over 99.8% for all protocol versions. Therefore, less than 0.2% of the packets require end-to-end retransmission. The low traffic load creates only low internal interferences resulting in few packet collisions. Resulting congestion can be cleared before the packet buffer overflows. At low traffic load, the protocol *no FEC* shows a higher ETX count than the FEC enabled protocols. Therefore, the used FEC codes are able to reduce the required local transmission attempts. The FEC enabled protocols do not show a measurable difference among themselves.

The channel occupancy and the resulting internal interferences reach a critical level at a data rate of 2 packets per second. The FEC enabled protocols show now a higher ETX count as well as significantly more and longer periods of congestion

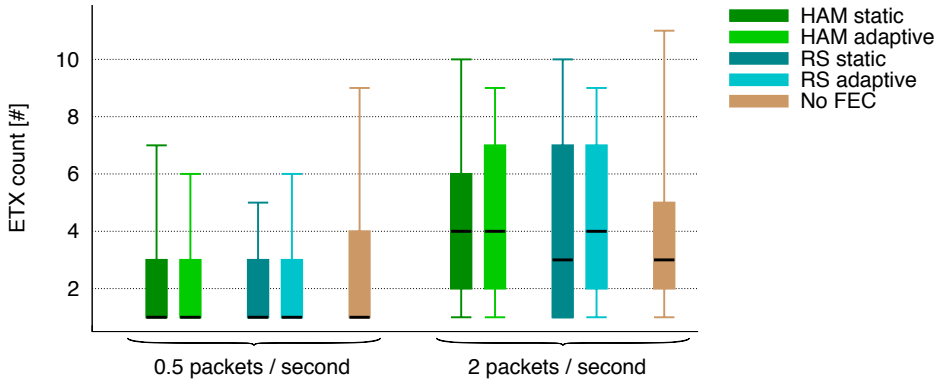


Figure 5.42: ETX count for protocols with static, adaptive and without FEC support.

than *No FEC*. This results in packet dropping caused by buffer overflow. The congestion additionally affects the transmission delay optimization of BEAM. The resulting longer beacon strobe periods increase internal interferences, which cause more additional bit errors in a single packet than the FEC code is able to recover. Therefore, the FEC codes even increase the packet loss ratio at high traffic load values.

Table 5.11 shows the amount of successfully recovered encoded packets. The rest of the received encoded packets either was free from bit errors or contained too many bit errors for a successfully recovering. The Reed-Solomon code was able to restore more packets than the used Hamming code. The adaptive versions recover more encoded packets than the corresponding static versions.

traffic load	**Hamming(12,8)**		**Reed-Solomon(255,225)**	
packets/sec	Static	Adaptive	Static	Adaptive
0.5	1.2%	3.0%	8.2%	10.4%
2.0	2.5%	4.8%	12.4%	21.2%

Table 5.11: Successfully recovered packets.

The energy efficiency of adaptive FEC implementation using the CC2420 radio module is limited for the following three reasons:

- **CC2420 radio module only shows isolated packets with bit errors:** The majority of the adaptively encoded packets do not show any bit error in our experiments. This indicates that most bit errors are caused by short-term interferences and collisions. Adaptive FEC implementation using different FEC codes, as introduced by [41], would switch most of the time between no FEC and the weakest FEC Hamming code when using a CC2420 radio module. Moreover, most of the encrypted packets do not contain a bit error.

Figure 5.43 depicts an example of two different adaptive FEC implementations. Only few of the packets received by the CC2420 radio module show bit errors. The *FEC with immediate encryption adaption* encrypts a data packet after detecting a bit error. The next frame does not contain any bit error, therefore, the following packet is not anymore encoded. *FEC with immediate delayed adaption* encrypt several packets after detecting an erroneous transmission. In case of multiple FEC codes, only the weakest FEC code is applied, due to only one packet contained an error.

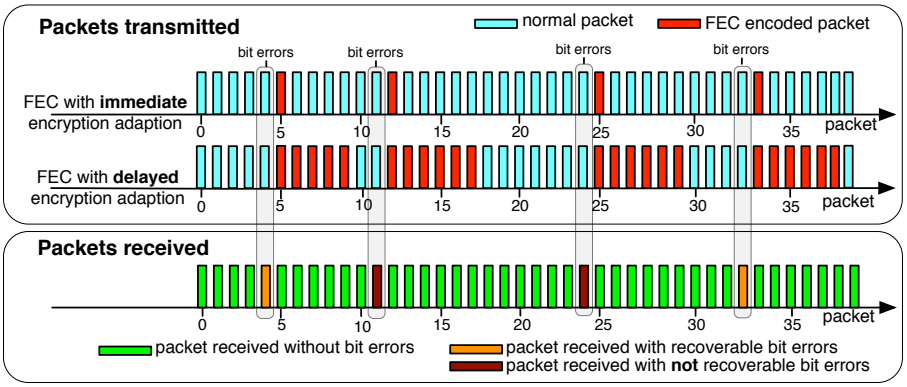


Figure 5.43: Adaptive FEC codes with CC2420 radio module.

- **Amount of bit errors in a corrupted packet:** Our Reed-Solomon(255,225) implementation shows a higher recovery potential than every FEC code used in [41]. Nevertheless, Reed-Solomon(255,225) is only able recover up to 21.2% of the corrupted packets in our measurement. Therefore, also a more detailed adaptive FEC implementation would not have recovered more than 21.2% of the corrupted packet in these measurements.

- **Radio duty cycle protocol:** ARQ mechanisms and FEC codes require a radio duty cycling protocol to save energy. ARQ based mechanisms are clearly more suitable to realize an energy efficient radio duty cycling protocol than FEC codes (see Subsection 4.1.5).

Table 5.12 shows the overall packet loss including end-to-end retransmissions. The end-to-end retransmission mechanism of UDP-E2E is able successfully forward all packets at a traffic load of 0.5 packets per second. Therefore, FEC codes do not have impact on the packet loss at low traffic load. At 2 packets per second, the FEC enabled protocols show a significantly higher end-to-end packet loss than the protocol *No FEC*. The adaptive FEC protocol versions show lower packet loss than their static versions. Additionally, Hamming(12,8) shows lower packet loss than Reed-Solomon(255,225). At high traffic load, all the FEC enabled protocols show

clearly higher traffic loss than *No FEC*. The results in Table 5.12 show that *No FEC* is superior to the FEC enabled protocols concerning reliability.

traffic load	**Hamming(12,8)**		**Reed-Solom(255,225)**		**BEAM/H2HR**
packets/sec	Static	Adaptive	Static	Adaptive	No FEC
0.5	0%	0%	0%	0%	0%
2.0	12.5%	4.3%	35.4%	6.2%	0.2%

Table 5.12: Overall packet loss.

Energy Consumption Evaluation

Besides reliability we analyzed the impact on energy consumption. We added an energy measurement mechanism to the FEC implementation to evaluate the energy required for FEC encoding and decoding calculations. This FEC energy measurement mechanism accumulates the time periods required by the microcontroller to execute FEC encoding and decoding operations. The used energy is calculated by multiplying the required time with the power required by the microcontroller at 100% load. Figure 5.44 shows the corresponding energy consumption.

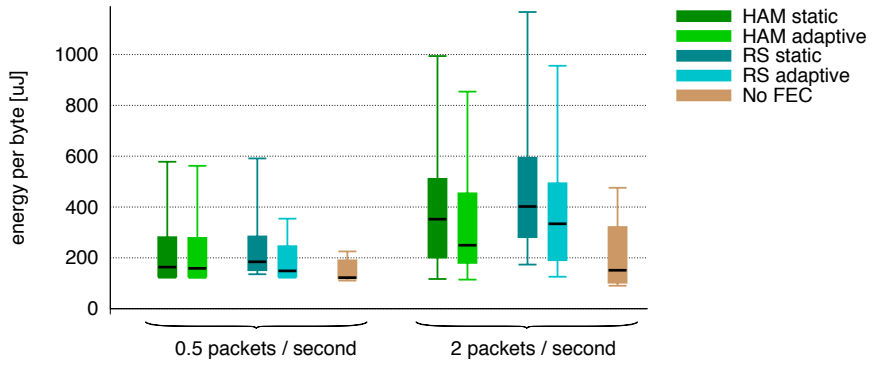


Figure 5.44: Energy required forwarding 66 bytes payload.

At low traffic load, the FEC enabled protocols require more energy than *No FEC*. This is caused by the short beacon strobes, the required FEC calculations and the additionally transmitted parity data. The difference between *No FEC* and *HAM adaptive* is caused by the beacon strobe mode and the longer payload. Table 5.13 shows the percentage of the energy required for the FEC calculations. The rest of the energy is required by the radio module. The higher the traffic load is the more packets have to be decoded due to bit errors caused by interferences. Adaptive FEC codes encode a lower amount of data packets.

traffic load	**Hamming(12,8)**		**Reed-Solomon(255,225)**	
packets/sec	Static	Adaptiv	Static	Adaptiv
0.5	1.41%	0.239%	16.5%	7.95%
2.0	5.17%	3.14%	27.4%	16.1%

Table 5.13: Percentage of the energy required for the FEC calculations.

5.4.4 Throughput

This subsection takes an additional look to the impact of FEC codes on the throughput. The previous subsections showed that FEC mechanisms reduce the throughput of a real world WSN network stack featuring additional functions such as radio duty cycling or ARQ based retransmission. The intention of this subsection is to measure the throughput using telosB nodes and Contiki without duty cycling and the overhead of ARQ. For the following throughput evaluations we create an additional network stack. We use nullRDC and nullMAC on the link layer to achieve high throughput on the link layer. We completely removed μIP from the network stack to minimize the delay caused by the network stack. We added the FEC codes Hamming(12,8) and Reed-Solomon(255,225) to nullRDC. As an ARQ reference, we modified the nullMAC protocol to retransmit a lost packet up to three times with a short random back-off delay. Figure 5.45 depicts the network stack designed for throughput evaluation.

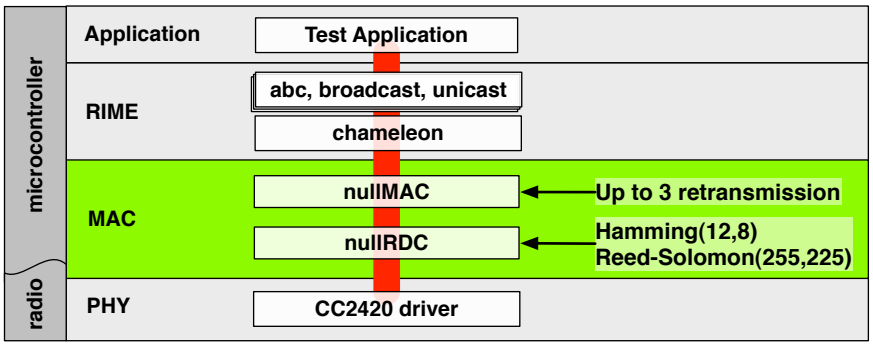


Figure 5.45: FEC enabled network stack for throughput evaluation.

We use two telosB nodes placed within a distance of 20 cm as depicted in Figure 5.46. The sensor node is connected over a serial line to a notebook to control the experiment. The second node is connected to the RIGOL multimeter. The first node, which is directly connected to the notebook, initially sends a packet with 66 bytes payload to the second node. This second node sends it back to the first node one and the first again to the second. The packets now go back and forth as fast as possible between the two nodes. This enables the measurement of the time required to forward a packet over two hops with the RIGOL multimeter and the

notebook.

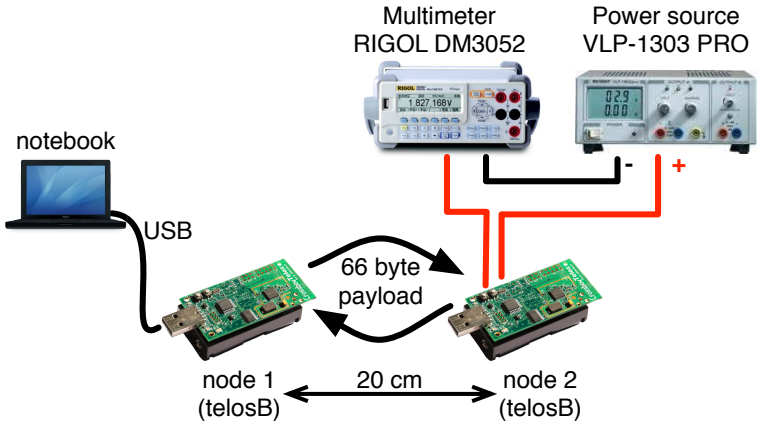


Figure 5.46: Testbed for throughput evaluations.

The high SNR results in over 99% error free transmissions. We created three experiments with this testbed. In every experiment, we compared the performance of nullRDC with enabled Hamming(12,8) or Reed-Solomon(255,225) support with nullRDC without FEC support. In experiment 1, we measured the required time to forward a packet if no bit errors occur. In this case, the FEC enabled protocols only have to encode the payload. The decoding is not required due to no bit error occurred. In experiments 2 and 3 we measured the impact of bit errors by injecting a single bit error into the encoded packets. In experiment 2, only one of the two nodes makes this injection. In experiment 3, both nodes inject a single bit error. By injecting only a single bit error, both FEC codes are able to repair the packets. In experiment 2 and 3, the protocol without FEC support executes one retransmission due to the bit error injected in the first packet. No bit error is injected to the retransmitted packet.

The results are shown in Figure 5.47. In all experiments, FEC codes show always a lower throughput, although the FEC codes do not need any retransmission. Both FEC codes and ARQ handled exact same amount of bit errors. Due to the CC2420 acknowledgment support, the retransmission of an erroneous packet is clearly faster than encoding and successfully recovering a packet by Reed-Solomon. Therefore, Reed-Solomon shows a very low throughput due time required for recovering an erroneous packet. ARQ on the CC2420 radio module is able transmit around seven additional data packets during the time period required by our Reed-Solomon implementation to encode and recover a single packet.

5.4.5 FEC Summary

The performance of FEC codes depends on the used radio module. FEC codes show promising results on bit/byte oriented radio modules featuring only low en-

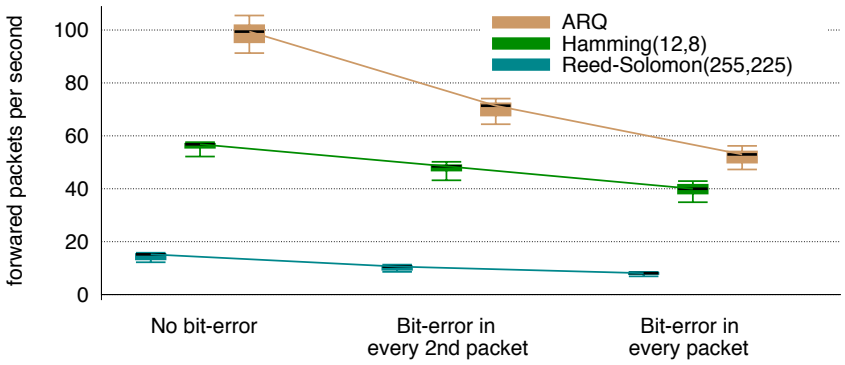


Figure 5.47: Throughput with and without FEC codes.

ergy efficiency and low reliability. But with packet oriented radio modules using a DSSS spreading mechanism, FEC codes require more time and energy than the ARQ mechanism offered by the radio module. The evaluated FEC codes are neither able to reduce the energy usage nor to enhance the reliability of our real world WSN network stack using a packet oriented radio module. Although FEC codes are able to reduce the ETX count under low traffic load on these radio modules, a standard ARQ mechanism is able to recover the erroneous packets using less energy. Under high traffic load values, ARQ mechanisms show a lower end-to-end packet loss than FEC enabled protocols. In our measurements, the more energy efficient Hamming(12,8) code recovered less than 5% of the corrupted packet. The advanced FEC code Reed-Solomon(255,225), which has a high recovery potential, was able to recover up to 21.2% of the erroneous packets. Unfortunately, the percentage of packets recovered by all our implemented FEC codes are insufficient to compensate for their additional time requirements and energy costs. Therefore, we do not use FEC support in our final protocol versions of BEAM and H2HR designed for the CC2420 radio module. Maybe a more complex adaptive FEC mechanism with a radio duty cycle mechanism optimized for adaptive FEC code is able to save energy with the CC2420 radio module.

5.5 Comparing BEAM to Existing WSN Protocols

This section compares our final protocol versions of BEAM and H2HR with already existing link layers protocols for packet oriented radio modules. The evaluated protocols included in the network stacks are shown in Figure 5.11 and 5.12. The following performance characteristics are analyzed:

- **Energy consumption:** We compare the energy required to forward data as well as the energy consumed by idle neighbor nodes. The adapted Contiki energy profiler measures the consumed energy.

- **Reliability and Throughput:** End-to-end packet loss measurements are used to compare the data delivery reliability performance of the used protocols. Reliability depends strongly on the used traffic load. Therefore, the traffic load limit is evaluated in addition.
- **Packet delivery time:** Finally, we compare the required packet delivery time of the network stacks. The packet delivery time is defined as the time between the first transmission of a generated packet by the sender node and the successful reception by the addressed receiver node.

The presented measurements are all made in the WISEBED testbed. We use the three large-scale scenarios described in Subsection 5.1.2. The next subsection presents the energy evaluation.

5.5.1 Energy Consumption

In this subsection we compare the energy consumption of BEAM, XMAC and ContikiMAC in the large-scale scenarios defined in the WISEBED testbed. Moreover, we analyze the energy consumption of the noninvolved neighbor nodes in the streaming scenario.

Energy Consumption Required in the Streaming Scenario

First, we compare the energy required to forward traffic in the streaming scenario (Figure 5.4a). We analyze different generated traffic rates from 0.125 to 8 packets per second on each of the four end-to-end flows. We use UDP packets with a link layer payload of 40 bytes. For each configuration, we send 50'000 packets. Figure 5.48 shows how much energy was required to forward one byte for different generated traffic load values of BEAM, XMAC and ContikiMAC. Moreover, Figure 5.49 depicts only BEAM and ContikiMAC for a better comparability of the two protocols.
Below 0.25 packets per second, only one single packet is usually forwarded in the entire network at the same time. The reliability mechanism handles external interferences. This results in quite predictable energy consumption for low traffic load values. The **total energy** consumed by the radio module increases with additional traffic load while the required **energy per byte** decreases with additional traffic load. The energy per byte decreases due to periodic channel checks. Depending on the traffic load, dozens or hundreds of wake-up periods are performed between forwarding of 2 packets. This dramatically increases the energy costs required per byte at very low traffic rate.

Above 0.25 packets per second, there is an increasing probability of multiple packets at the same time in the WSN. This causes inter-flow interferences, which have to be handled by the reliability mechanism, i.e., by longer beacon strobe transmissions and additional hop-to-hop retransmissions. Therefore, the energy costs of ContikiMAC and BEAM clearly increase to above 0.25 packets per second. The

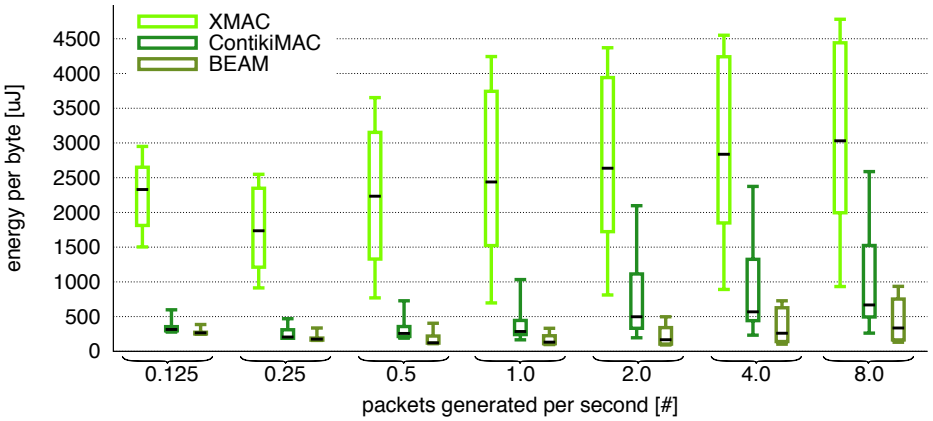


Figure 5.48: Energy per byte in UDP streaming scenario (including XMAC).

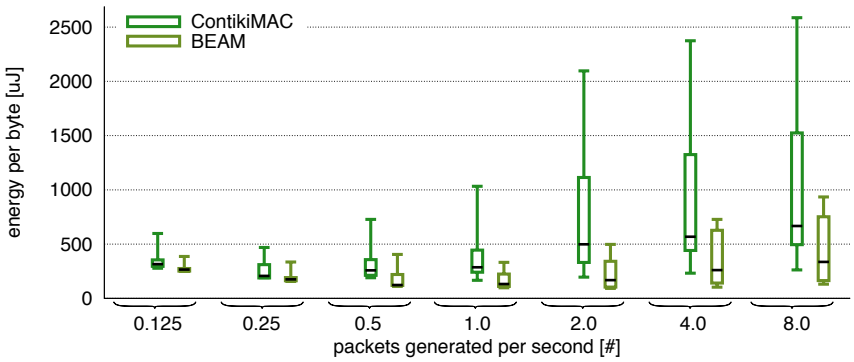


Figure 5.49: Energy per byte in UDP streaming scenario (without XMAC).

energy cost of XMAC only increases slightly, as the packet error ratio of XMAC is too high to be handled by the hop-to-hop reliability protocol (CSMA) of Contiki. Most of the packets are dropped by the hop-to-hop reliability protocol CSMA on the first hops.

The data rate can only be increased up to a specific radio channel load. At higher channel loads the resulting internal interferences cause congestion and decrease the forwarded traffic load. If considerably more packets are generated than the radio channel is able to handle, then the exceeding packets are usually dropped on the first hops. The quicker the exceeding packets can be dropped, the less they stress the radio channel. The reliability evaluation in Subsection 5.5.2 takes more detailed look on this topic.

XMAC requires clearly more energy than ContikiMAC and BEAM. First, as XMAC requires significantly longer listen intervals to execute periodic channel checks. Second, XMAC uses the radio channel capacity clearly less efficiently

than ContikiMAC and BEAM. Thus, XMAC requires clearly more beacon strobes to forward a packet. The beacon strobe mechanism used by XMAC is more vulnerable to interferences than the beacon strobe mechanisms of ContikiMAC and BEAM. This is basically caused by not supporting a beacon strobe transmission delay function and by not using the CC2420 radio module link layer support. This results in additional retransmissions by the hop-to-hop reliability protocol CSMA.

ContikiMAC and BEAM require nearly the same energy for periodic channel checks at low traffic load values. But BEAM requires less beacon strobe transmissions at low traffic rates than Contiki, due to a more precise beacon strobe transmission delay mechanism. BEAM and H2HR achieve a clearly more efficient channel usage than ContikiMAC with CSMA. Nodes using BEAM show a higher total energy consumption than nodes using ContikiMAC at data rates higher than 2 packets per second. But the higher throughput of BEAM results in a lower energy consumption per successfully received byte at the sink.

Energy Consumption of Noninvolved Neighbor Nodes

Next, we analyze the energy required by idle neighbor nodes. These nodes perform periodic wake-up periods to check the channel. They never have to forward any packet. Figure 5.50 shows the measured energy consumption. XMAC requires clearly more energy than the other protocols due to the longer listen intervals to execute periodic channel checks. At higher traffic load values, XMAC sometimes detects an incoming strobe at the beginning of the listen interval. This requires slightly less energy than listening to the channel for the required time to detect a potential beacon strobe. The idle nodes of ContikiMAC and BEAM show similar energy consumption.

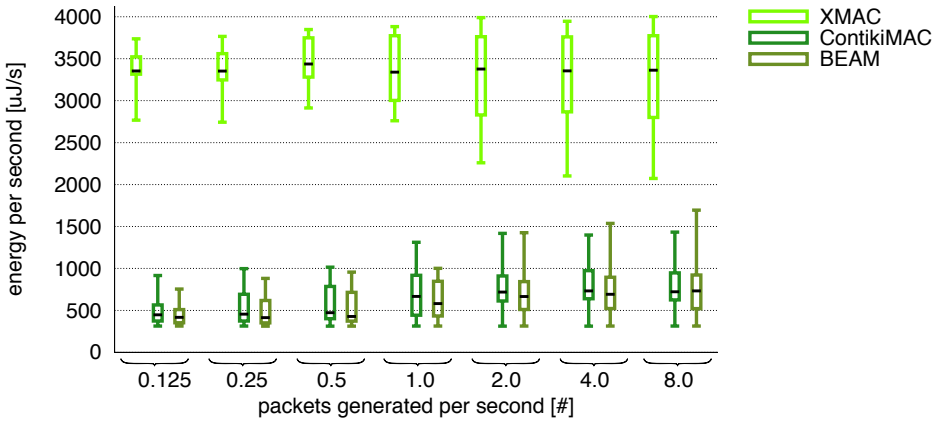


Figure 5.50: Energy per second of noninvolved nodes.

Energy Consumption Required in the Event and Burst Scenarios

Next, we compare the energy consumption of XMAC, ContikiMAC and BEAM within the event and burst scenario defined in Subsection 5.1.2. In the event scenario, four nodes try to forward one single packet nearly simultaneously to the sink (Figure 5.4b). We measure the energy required to forward the packet containing the event notification. The notification packet has a size of 43 bytes on the link layer. In the burst scenario, the sink sends 800 bytes application data to the four nodes (Figure 5.4c). Figure 5.51 shows the energy per byte required in the event and burst scenario. The event scenario does not trigger any end-to-end retransmis-

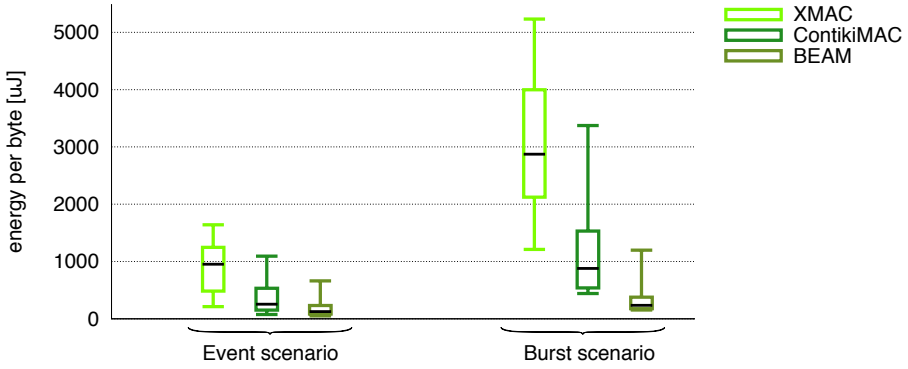


Figure 5.51: Energy per forwarded byte in event and burst scenario.

sions. The receiver cannot detect a missing packet without ever having received one. The missing packet cannot be detected before the next packet (generated by the next event) indicates the missing sequence number. But such a delayed packet is not useful for many real applications. Therefore, we do not make use of the possibility to detect packet loss in the event scenario. For the burst scenario, UDP-E2E is able to detected all packet losses. Like in the previous measurements, UDP-E2E only requests one lost packet at a time.

The event and burst scenarios do only properly work with UDP-E2E in these experiment setups. Most of the burst scenarios with executed TCP did not successfully deliver 800 bytes to all the four target nodes. The TCP acknowledgments for the SYN and data packets generated too much intra-flow interferences. XMAC and ContikiMAC additionally show problems with TCP in the event scenario. In most cases, either the first TCP SYN or some data packets get lost.

Event scenario: XMAC requires most energy per received byte. This is caused by more beacon strobe transmissions and higher packet loss. ContikiMAC shows some more beacon strobe transmissions than BEAM and a longer total time to transmit packets. XMAC and ContikiMAC generate clearly longer congestion periods than BEAM/H2HR on the first two hops where all four nodes concurrently try to send the notification packet.

Burst scenario: With XMAC, most of the packets are dropped during the first try. They have to be retransmitted by UDP-E2E. Therefore, XMAC requires clearly more energy than ContikiMAC and BEAM. ContikiMAC requires a longer packet delivery time and more end-to-end retransmissions than BEAM.

5.5.2 Reliability and Throughput

This subsection compares the reliability of the different link layer protocols. Reliability strongly depends on traffic load. The lower the traffic load is the lower are the internal interferences and resulting collisions as well as bit errors. At very low traffic load values, only one single packet is concurrently forwarded in the network. Under this condition only external interferences have to be handled. Local and end-to-end retransmission mechanisms increase interferences.

Both retransmission mechanisms require appropriate retransmission delays to not overload the already crowded radio channel. First, we analyze the **packet loss** ratio in the streaming scenarios. Figure 5.52 shows the measured packet loss. XMAC/CSMA shows the highest packet loss ratio. This link layer protocol combination shows a too high hop-to-hop packet loss ratio. UDP-E2E is not able to retransmit all lost packets. ContikiMAC/CSMA with UDP-E2E is able to forward over 99.8% of the packets up to a traffic load of 0.25 packets per second on each of the four paths. BEAM/H2HR is able to forward up to 2 packets per second.

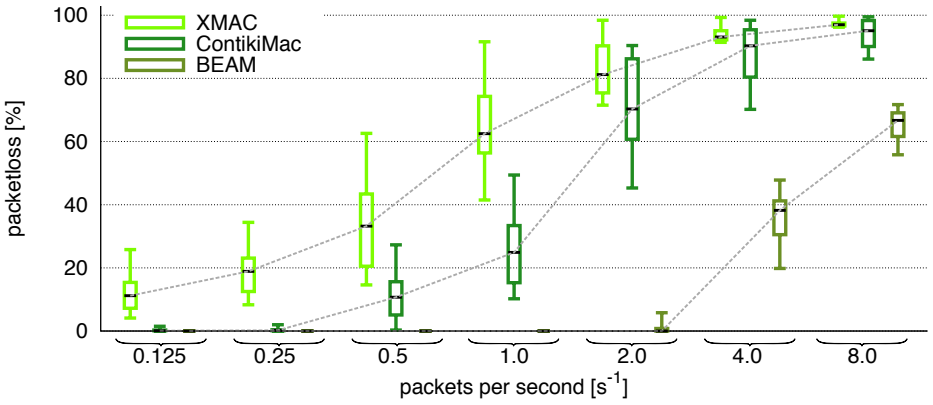


Figure 5.52: Packet loss ratio under different traffic load values.

End-to-end reliability mechanisms are only able to successfully retransmit packets if the hop-to-hop reliability mechanism does not drop too many packets. Every dropped packet generates additional interferences by the triggered end-to-end retransmissions. The performance of the hop-to-hop reliability mechanisms is essential for the overall reliability performance of the network stack. End-to-end reliability mechanisms should be used very carefully. They waste energy and can increase the total packet loss ratio at critical traffic load.

Throughput is closely related to packet loss. We analyze the throughput at different traffic load values. We generated traffic load values, which are much higher than the throughput the underlying protocols can handle. Figure 5.53 shows the measured traffic flows. XMAC and ContikiMAC show a significantly lower throughput. All three protocols experience a growth in throughput to an offered load of 1 packet per second. With a higher offered load, the throughput of XMAC and ContikiMAC decreases. The throughput of BEAM increases up to an offered load of 4 packets per second. Moreover, with BEAM the throughput does not decrease with a higher offered load.

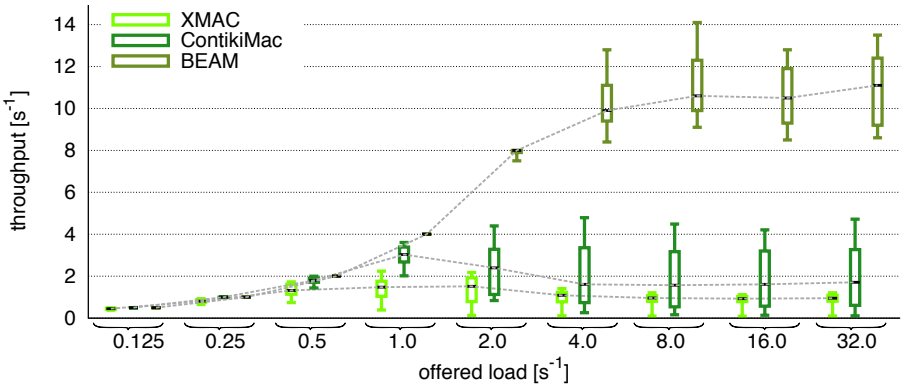


Figure 5.53: Maximum throughput.

5.5.3 Packet Delivery Time

This subsection takes a look at the required end-to-end packet delivery time of the experiments discussed in the previous subsections. Packet delivery time is defined as the time between the first transmission of a generated packet by the sender node and the successful reception by the addressed receiver node. Figure 5.54 shows the measured packet delivery times in the streaming scenario. Most of the packets show a quite similar average packet delivery time. Packets delivered without any end-to-end retransmission are usually delivered in less than one second. BEAM reduces the duty cycle duration with increasing traffic load. This results in a shorter packet delivery time for BEAM packets at high traffic load.

The longest packet delivery times of several seconds are caused by successful end-to-end retransmissions. Up to 2 packets per second over 99% of the packet can be successfully forwarded to the sink. UDP-E2E sometimes requires several retransmission attempts for a packet. Above this critical traffic load, UDP-E2E must start dropping packets from the packet buffer. The higher the traffic load above the critical traffic load is, the lower is the probability that UDP-E2E is able to successfully retransmit a packet. The longest packet delivery times above eight packets per second are caused by local retransmission delays to avoid congestion. ContikiMAC

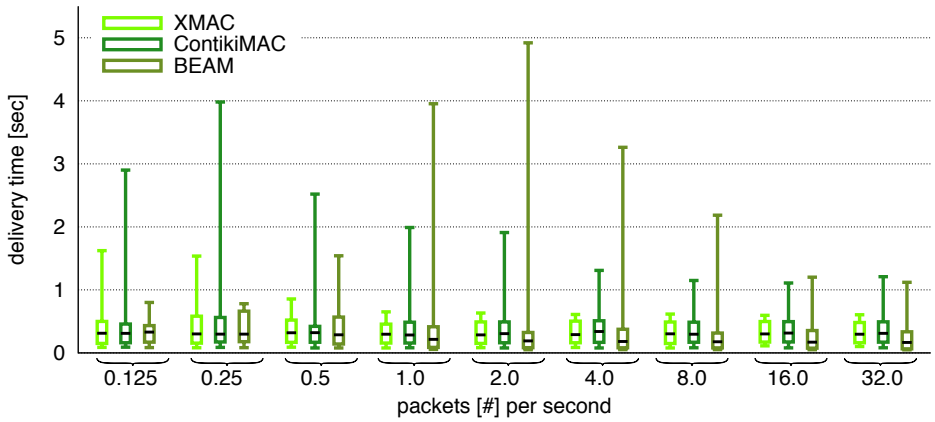
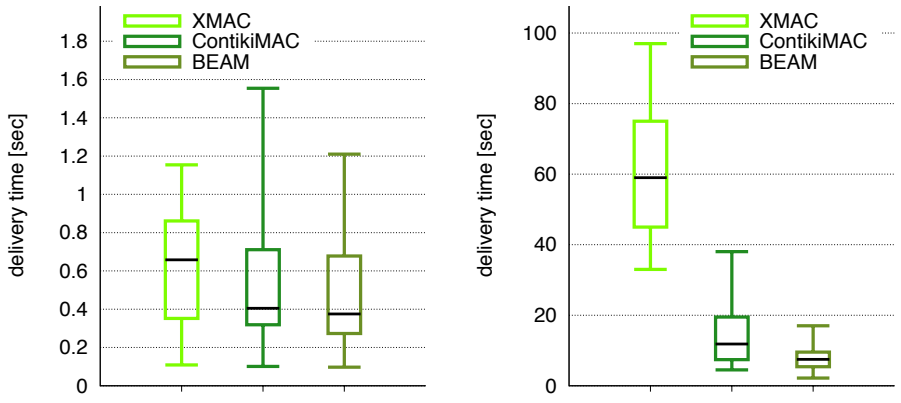


Figure 5.54: Packet delivery time for the stream scenario.

with UDP-E2E shows a similar behavior.

In the **event scenario**, we measured the time required by the first frame to reach the sink. The results are shown in Figure 5.55a. XMAC, ContikiMAC and BEAM are using the same default duty cycle duration during the event is detected. With XMAC, 55.3% of the events were not reported to the sink due to too strong interferences and resulting packet loss. This limits the outliers of XMAC. The beacon strobe mechanism of XMAC is more vulnerable to the internal interferences caused in the scenario. ContikiMAC and BEAM show almost the same packet delivery times.

For the **burst scenario**, we measured the time required to successfully forward the data to four nodes. The measured times are shown in Figure 5.55b. With



(a) Event scenario.

(b) Burst scenario.

Figure 5.55: Packet delivery time for the event and burst scenarios.

XMAC/CSMA most packets have to be retransmitted by UDP-E2E. This requires a lot of additional time.

5.5.4 Summary Comparing BEAM to Existing WSN Protocols

In this section we compared the performance of our contributed link layer stack to existing link layer stacks for energy efficient packet oriented radio modules. Our link layer stack, containing BEAM and H2HR, was the most energy efficient and reliable of the evaluated link layer stacks. Moreover, our contributed link layer protocols shows a significant higher throughput than the other evaluated protocols. This is due to the fact that our contributed link layer stack is able to handle intra-flow and inter-flow interferences. The other protocols struggle with handling intra-flow and inter-flow interferences caused by concurrently forwarded packets. All link layer stacks evaluated in this section are designed for packet oriented radio modules. The following section additionally compares our contributed protocols to energy efficient MAC protocols on bit/byte oriented radio modules.

5.6 BEAM Compared to Energy Efficient MAC Protocols for Bit/Byte Oriented Radio Modules

Within a real world testbed, a direct comparison of BEAM to energy efficient MAC protocols, which use adapted physical preambles or specific link frame headers such as MaxMAC is extremely difficult. Those energy efficient MAC protocols require different types of radio modules and do usually neither handle hop-to-hop nor end-to-end reliability mechanisms. We made the following assumptions to perform a reasonable comparison of BEAM to other energy efficient MAC protocols using bit/byte oriented radio modules.

- Energy efficient MAC protocols for bit/byte oriented radio modules require adapting the link or physical headers and cannot be implemented on packet oriented radio modules such as the CC2420.
- The energy usage of the reference link layer protocol introduced in Subsection 3.1.4 depicts the maximum energy efficiency that can be reached by the required bit/byte oriented radio modules. Real world protocol implementations usually require several magnitudes more energy than the reference link layer protocol. This is due to real world protocols additionally require energy to check the channel and perform retransmissions of lost packets. Moreover, energy efficient link layer protocols usually require additional messages such as beacon strobes, long preambles or synchronization messages to handle the duty cycles.

Figure 5.56 shows the energy required by the reference protocol to forward 50 bytes payload. The energy used by bit/byte oriented radio modules is calculated

for a physical data rate of 38.4 kb/s. Next to the reference protocol calculations we performed a real world experiment with our contributed network stack including BEAM, H2HR and UDP-E2E. We forwarded 15'000 packets including 50 bytes payload, with a packet rate of 2 packets per second in a line scenario with three nodes. We used the testbed including the RIGOL multimeter described in Figure 5.6 to determine the energy usage of the middle node, which is forwarding the packets. As expected, the energy consumption measured in the real world experiment is several magnitudes higher than the energy required by the hypothetical reference link layer protocol. Nevertheless, the real world implementation of BEAM including H2HR reliability support still requires less energy than every other energy efficient MAC protocols using bit/byte oriented radio modules.

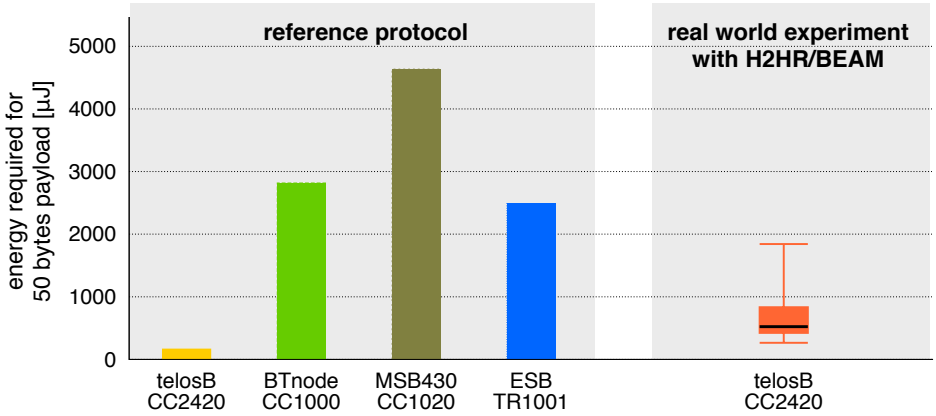


Figure 5.56: Energy usage of BEAM compared to the reference protocol.

5.7 Experiences with Different Evaluation Methodologies

This section describes the experiences made with different evaluation methodologies. In the first part, we describe how the results achieved by the OMNeT++ simulator match to the results delivered by the WISEBED testbed. The second part discusses the impact of the used traffic load on the achieved results.

5.7.1 Simulation versus Real World Experiences

The results of most real world experiments presented in this chapter confirm results achieved with the OMNeT++ simulator. Especially the line, parallel and merge scenarios show similar results, which lead to the same conclusions. However, in some cases the evaluated protocols show a better performance in the OMNeT++ simulator than in real world testbeds. Besides the better results, the individual

measurements show a smaller standard deviation and less outliers. This happens in some of the cross scenarios as well as during operations when the radio channel is highly allocated. In these cases, the used models in the OMNeT++ simulator are not able to properly simulate the internal interferences. Real world experiments show more retransmissions under high traffic load and in complex scenarios.

Extensive and complex optimization algorithms perform better with the OMNeT++ simulator than in real world testbeds. They show an improvement in the OMNeT++ simulator, but fail in real world experiments. The main reason for the difference are simplifications at the physical layer. These simplifications do not consider delays caused by radio modules and microcontrollers, i.e., simplified radio models are not able to simulate different kind of interferences caused at high traffic loads. Therefore, we propose to use OMNeT++ simulator models that simulate microcontroller delays, radio module behavior as well as interferences as realistically as possible. This includes implementing the state machine, delays and interface of the radio module. In addition, network topologies generating inter-flow and intra-flow interferences should be used. Otherwise a proposed optimization could improve the behavior within a simulated environment, but fail in real world under high traffic load and interferences. A good example for this is the duty cycle duration adaptation mechanism based on monitoring used by XMAC and MaxMAC. This works perfectly fine for basic simulations or simple real world implementations, but it completely fails in more challenging environments.

All protocol implementations show significantly better results with the simple radio model delivered by OMNeT++. The developed CC2420 radio module including a customized radio model based on real world SNR measurements delivers results that are closer to the results achieved by the WISEBED real world testbed than the radio model delivered by OMNeT++. Compared to bit/byte oriented radio modules, packet oriented radio modules have the advantage that receiving and sending a frame is independent from the microcontroller load. Controlling and timing of the radio module is a state machine, which is well defined and can be accurately simulated. No challenging UART bus transmission delay problems have to be simulated when sending or receiving a packet while processing other data. Therefore, the results achieved in real world test beds are rather consistent with the results predicted by emulation of packet oriented radios than bit/byte radio modules.

5.7.2 Impact of Traffic Load

The used network topology and the applied traffic load influence the level of inter-flow and intra-flow interferences. Simulations and real word experiments show that there are two different traffic load thresholds in a WSN.

Up to the first traffic load threshold, neither significant inter-flow nor intra-flow interferences appear. This is due to the fact that, up to this traffic load, only one single packet is concurrently forwarded in the WSN. The measured packet loss is almost constant. The concrete packet loss ratio primarily depends on the SNR of

the individual links. Evaluations with traffic load values up to the first threshold do not deliver significant results. They do not analyze the robustness of the protocol against inter-flow and intra-flow interferences.

At higher data rates, several packets are concurrently forwarded in the network. The protocols have to handle the increasing inter-flow and intra-flow interferences as well as congestion by an adequate mechanism. The second traffic load threshold is reached when the hop-to-hop reliability mechanism must drop packets. Up to this traffic load threshold, an efficient protocol is able to handle the generated packers without significant packet loss. The more efficient the protocol in forwarding packets is, the higher is the second traffic load threshold.

With further increasing traffic load, the radio channel will reach the maximum channel load. Packets have to be dropped to prevent congestion. The maximum channel load depends on the used protocol mechanism.

Chapter 6

Conclusions and Outlook

Wireless Sensor Networks have experienced an increasing degree of research interests and a growing number of industrial applications in the last decade. Various real world WSNs are individually customized to fulfill their intended purposes with limited energy resources and small processing power of a microcontroller. These real world WSNs have to be cost-efficient and to ensure functional reliability during operation.

6.1 Addressed Challenges

The following challenges for real world WSNs are addressed in this thesis :

- **Energy Efficiency:** Energy efficiency is a key challenge in the realization of a real world WSN. Battery powered sensor nodes offer only a limited amount of energy, which determines the node and network lifetime. Energy preserving mechanisms enhance network lifetime by turning off energy consuming components such as the radio module, the microcontroller and attached sensing devices. The longer the sleeping period in relation to the wake-up period is, the more energy can be saved. Most effective but also most challenging is to save energy by duty cycling the radio module. Another technique for preserving energy is to reduce the amount of packet transmissions in a WSN. Reducing the amount of transmissions can be realized by reducing intra-flow and inter-flow interferences to decrease the bit error probability and by adding redundant information to enable data recovery at receiver nodes.

- **Reliable Data Transmission:** Current real world WSN protocol stacks only offer weak data reliability support. Usually, they struggle when handling inter-flow and intra-flow interferences generated by concurrently forwarded packets. Transmission attempts have to be scheduled carefully in order to not interfere with transmissions of other, maybe hidden, nodes. Otherwise, periods with high traffic load generating inter-flow and intra-flow interferences result in congestion and significant packet loss.

- **Network Connectivity:** Usually, real world WSNs consist of heterogeneous sensor nodes, equipped with individual sensing devices to detect different environmental characteristics. The individual heterogeneous sensor nodes should be interconnected among themselves and connected to the Internet for remote management applications.

6.2 Contributed and Applied Protocols

To provide a solution for all these challenges we contribute an energy efficient real world WSN protocol stack supporting reliable data transmission. We use standard network and transport protocols to establish and handle end-to-end connection between sensor nodes in a WSN and servers in the Internet. Moreover, we use the standardized physical and data link layer protocol IEEE 802.15.4 to enable direct communication among heterogeneous neighbor nodes. Bellow we provide an overview of the individual protocols used in our WSN network stack.

- **BEAM:** BEAM supports asynchronous adaptive duty cycles to adapt the frequency of the wake-up periods according to current traffic load. The contributed **traffic prediction mechanism** is able to predict incoming traffic load even during congestion and heavy interferences. The higher the incoming traffic load is, the shorter are the applied sleeping periods. BEAM is designed to work best with the energy efficient and IEEE 802.15.4 compliant packet-oriented radio module CC2420. IEEE 802.15.4 compliant radio modules enable the interconnection of heterogeneous sensor nodes and they support the DSSS spread spectrum technique, which significantly enhances the robustness against interferences. Moreover, BEAM offers a detailed transmission report to the hop-to-hop reliability protocol to detect congestion.

- **H2HR:** H2HR enables hop-to-hop reliability including a **congestion detection and control mechanism**. The contributed congestion detection and control mechanism is able to detect and properly react on intra-flow and inter-flow interferences as well as congestion by adapting the frequency of individual transmission attempts. This results in higher throughput and reduces the number of energy consuming retransmission attempts caused by packet collisions.

- **UDP-E2E:** UDP-E2E is an application layer protocol supporting end-to-end reliability based on UDP. Using UDP-E2E, a reliable and energy efficient end-to-end transport service between sensor nodes and machines in the Internet may be offered. The application layer protocol employs standardized network sockets to support a regular application programming interface.

- **μIP :** Our contributed network stack extends the existing modular Contiki network stack including μIP. μIP enables the support of IP, TCP, UDP and

ICMP, which build the de facto standard protocol suite for Internet communication. By running these protocols in the WSN, it is possible to directly connect the WSN to a wired network infrastructure without proxies or middle-boxes.

6.3 Main Results and Conclusions

Our main contribution is an energy efficient real world WSN protocol stack supporting reliable data transmission in heterogenous WSNs. We contribute three protocols BEAM, H2HR and UDP-E2E to improve energy efficiency, data reliability and throughput of the existing Contiki network stack supporting μIP.

BEAM is an energy preserving, adaptive RDC protocol with a forward-looking traffic load prediction mechanism to determine an appropriate duty cycle also during periods of congestion. Congestion requires short sleeping periods to increase network bandwidth for high traffic load. Existing adaptive duty cycle protocols use traffic monitoring mechanisms to adapt sleeping periods to traffic load. Traffic monitoring mechanisms count the recently forwarded amount of packets to determine the current traffic load. Congestion results in a lower forwarded number of packets. This causes a lower number of counted packets and, therefore, traffic monitoring will increase the sleeping period length in case of congestion. The traffic prediction mechanisms enables BEAM to apply short sleeping periods also during periods with congestion to offer sufficient network bandwidth.

The BEAM frame headers are fully compliant with IEEE 802.15.4. This enables the execution of common link layer tasks such as sending acknowledgments or CRC calculations directly by the CC2420 radio module without interaction with the microcontroller.

H2HR is a hop-to-hop reliability protocol including a congestion detection and control mechanism. H2HR uses the explicit acknowledgment mechanism of BEAM to detect packet loss. Local retransmissions of a lost packets avoid energy-costly end-to-end retransmissions. The individual packet transmissions have to be carefully timed to reduce intra-flow or inter-flow interferences generated by concurrently forwarded packets. Moreover, the hidden node problem represents a major problem in network scenarios with multiple networks paths. Especially periods with numerous concurrently forwarded packets may result in congestion due to too heavy interferences. The congestion detection and control mechanism enables H2HR to calculate according transmission delays to reduce the number of collisions caused by concurrently forwarded packets.

BEAM and H2HR feature a packet aggregation mechanism to prevent or at least decrease congestion by reducing the number of concurrently forwarded packets. The reduction of concurrently forwarded packets reduces interferences and increases throughput.

UDP-E2E supports end-to-end reliability for UDP streams. It is based on selective acknowledgments using sequence numbers to request end-to-end retransmission of

lost packets. UDP-E2E is able to successfully retransmit lost packets if the hop-to-hop reliability mechanism does not drop more than 2-5% of the packets between source and sink. Higher end-to-end packet loss rates cannot be recovered, because intra-flow interferences caused by the selective acknowledgments increase packet loss instead of reducing it.

Our **evaluation of available radio modules** presented in Chapter 3 shows that the CC2420 radio module is the most energy efficient and robust of the evaluated radio modules. The DSSS mechanism of the CC2420 is significantly more robust against interferences than the modulation techniques used by bit/byte-oriented radio modules. Moreover, the CC2420 was the only radio module that supports a standardized physical and link layer to enable direct communication among heterogeneous sensor node platforms. The CC2420 is widely used in various sensor nodes such as telosB, micaZ, sensinode or imote2. The link layer functions offered by the CC2420 radio module reduce the load of the microcontroller and ensure a precisely predictable transmission timing, which is required to realize energy efficient radio duty cycles.

In addition, we evaluated the impact of **FEC codes** on the energy efficiency and data reliability in real world WSNs. We added FEC support to BEAM to enable real energy measurements on sensor nodes in a WSN. Usually, the impact of FEC codes to energy efficiency is evaluated without a radio duty cycling protocol.

The reliability performance of FEC codes depends on the used radio module. The evaluated FEC codes show promising results on bit/byte oriented radio modules featuring low energy efficiency and low data reliability. However, they fail on energy efficient packet oriented radio modules using frequency spreading techniques such as DSSS. Although FEC codes are able to reduce the ETX count under low traffic load, H2HR is able to recover the erroneous packets using less energy. Under high traffic load values, the FEC enabled protocols shows higher end-to-end packet loss than BEAM and H2HR without FEC support. The evaluated FEC codes are neither able to reduce the energy usage nor to enhance the reliability of our real world WSN network stack using a packet oriented radio module. In our measurements, the more energy efficient Hamming(12,8) code recovered less than 5% of the corrupted packets. The advanced FEC code Reed-Solomon(255,225), which has a high recovery potential, was able to recover up to 21.2% of the erroneous packets. This recovery rates are insufficient to compensate the additional time requirements and energy costs of the FEC coding.

6.4 Outlook

Our contributed WSN protocol stack improves the energy efficiency, data reliability and network connectivity in WSNs. However, there are some additional ideas to improve our WSN protocol stack for real world WSN applications:

Lifetime defined link layer protocol: Usually, adaptive radio duty cycling protocols use a fixed default duty cycle duration length. For example ContikiMAC

comes with a default duty cycle duration of 125ms. Long predefined default duty cycle durations enhance the network lifetime but also increase the packet delivery time. A lifetime defined link layer protocol supports an adaptive default duty cycle duration. The length is adapted according to the remaining battery energy and recommended lifetime. Software based energy profilers enable the recording of the already consumed energy. By knowing the total amount of energy provided by the used battery and the already consumed energy, the remaining lifetime can be estimated. If the remaining energy is too low, then BEAM and H2HR are able to save additional energy. BEAM can increase the default duty cycle duration to save more energy during periods with low traffic load. Moreover, H2HR can reduce the retransmission limit to reduce the energy during high traffic load. Allowing packet loss during congestion, avoids plenty of energy consuming retransmission attempts. Moreover, reducing the retransmission attempts during periods with too many concurrently forwarded packets, decrease the inter-flow and intra-flow interferences. Simulations can be used to calculate the energy required for an intended traffic load, data reliability and lifetime to determine small and cost-efficient batteries.

Hardware support: Our evaluations show that link layer tasks executed by the radio module require significantly less execution time and energy than handling by a software based application on the microcontroller. Moreover, link layer tasks executed by the radio module reduce the complexity of WSN protocol implementations. Spectrum spread techniques on a radio module significantly increases the robustness against interference and resulting bit errors. Future radio modules may support frequency-hopping spread spectrum (FHSS) techniques, which use different carrier waves frequencies to increase robustness against interference. FHSS switches between the channels in a pseudorandom sequence, which is known to the transmitter and receiver. Next to security advantages, FHSS shows a high resistants to interferences. The resistants to interference can be enhance by using Adaptive Frequency Hopping Spread Spectrum (Adaptive FHSS). Adaptive FHSS tries to use only frequencies with low bit error rates. Perhaps future packet oriented radios for WSNs will support FHSS.

Predefined scenarios and traffic patterns in real world testbeds: Comparing own protocol implementations to existing ones is challenging. Usually every protocol implementation is evaluated in a different testbed with different traffic loads and network scenarios. Protocols evaluated in basic network scenarios or with low traffic load do neither consider intra-flow and inter-flow interferences nor the impact of congestion. Defining a set of experiment setups with specific network scenarios and traffic loads in real world WSN testbeds such as WISEBED may improve comparability of different protocols. By using such predefined experiment setups, protocols could be better compared to each other.

Chapter 7

Acronyms

A-MPDU Aggregated MAC Protocol Data Units

A-MSDU Aggregated MAC Service Data Units

A-PPDU Aggregated Physical Protocol Data Units

abc Anonymous Best Effort Single-hop Broadcast

AFH Adaptive Frequency Hopping spread Spectrum

AFH Adaptive Frequency Hopping spread Spectrum

ARQ Automatic Repeat Request

ASK Amplitude Shift Keying

Adaptive FHSS Adaptive Frequency Hopping Spread Spectrum

BCH Bose-Chaudhuri-Hocquenghem

BEAM Burst-aware Energy-efficient Adaptive MAC

BFSK Binary Frequency Shift Keying

CCA Clear Channel Assessment

CNS Center at Nearest Source

CRC Cyclic Redundancy Check

CSMA Carrier Sense Multiple Access

DSSS Direct Sequence Spread Spectrum

ECC Error Correction Codes

EEPROM Electrically Erasable Programmable Read-Only Memory

ESB	Embedded Sensor Board
ETH	Eidgenössische Technische Hochschule
ETX	Expected Transmission Count
FCF	Frame Control Field
FCS	Frame Check Sequence
FEC	Forward Error Correction
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FSK	Frequency shift keying
GFSK	Gaussian Frequency Shift Keying
GIT	Greedy Incremental Tree
GUI	Graphical User Interface
H2H	Hop-to-hop Reliability
H2H	Hop-to-hop
IAM	Institute of Computer Science and Applied Mathematics
ibc	Identified Best Effort Single-hop Broadcast
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISM	Industrial, Scientific and Medical
LMAC	Lightweight Medium Access
LPL	Low Power Listening
LPP	Low Power Probing
LQI	Link Quality Indication
MFR	MAC Footer
mh	Best Effort Multi-hop Unicast
MHR	MAC Header

MPDU	MAC Protocol Data Units
MPDU	MAC Protocol Data Unit
MSB430	Modular Sensor Board 430
MSDU	MAC Service Data Units
NED	Network Description
nf	Best Effort Multi-hop Flooding
NRZ	Non Return to Zero
OMNeT++	Objective Modular Network Testbed
OOK	On Off Keying
OQPSK	Offset Quadrature Phase Shift Keying
PSDU	Physical Service Data Unit
PSK	Phase Shift Keying
RAM	Random Access Memory
RDC	Radio Duty Cycle
RFC	Requests for Comments
RMST	Reliable Multi-Segment Transport
ROM	Read-Only Memory
RSSI	Received Signal Strength Indicator
RTT	Round-Trip-Time
ruc	Reliable Single-hop Unicast
S-MAC	Sensor-MAC
SAW	Surface Acoustic Wave
SFD	Start of Frame Delimiter
SHR	Synchronization Header
sibc	Stubborn identified Best Effort Single-hop Broadcast
SNR	Signal to Noise Ratio
SPT	Shortest Paths Tree

suc	Stubborn Best Effort Single-hop Unicast
TARWIS	Testbed Management Architecture for Wireless Sensor Networks
TCP	Transport Control Protocol
TDMA	Time Division Multiple Access
TIK	Computer Engineering and Networks Laboratory
trickle	Reliable Multi-hop Flooding
TSS	TCP Support for Sensor Networks
uabc	Unique Anonymous Best Effort Single-hop Broadcast
uc	Best Effort Single-hop Unicast
UDP-E2E	UDP End-to-end Reliability
UDP	User Datagram Protocol
uibc	Unique Identified Best Effort Single-hop Broadcast
USB	Universal Serial Bus
WPAN	Wireless Personal Area Networks
WSN	Wireless Sensor Network
WiseML	Wireless Sensor Network Markup Language

Bibliography

[1] "A4-Mesh: Authentication, Authorization, Accounting, and Auditing in Wireless Mesh Networks." [Online]. Available: a4-mesh.unibe.ch

[2] A. Varga, "The OMNeT++ Discrete Event Simulation System." European Simulation Multiconference (ESM), Prague, Czech Republic, June 2001, pp. 319–324. [Online]. Available: http://www.omnetpp.org

[3] M. Anwander and T. Braun, "A Reliable, Traffic-adaptive and Energy-efficient Link Layer for Wireless Sensor Networks." Submitted for Publication to European Conference on Wireless Sensor Networks (EWSN), Ghent, Belgium, February 2013.

[4] M. Anwander, G. Wagenknecht, and T. Braun, "Management of Wireless Sensor Networks using TCP/IP." International Workshop on Sensor Network Engineering (IWSNE), Santorini, Greece, June 2008, pp. II.1–II.8.

[5] M. Anwander, G. Wagenknecht, T. Braun, and K. Dolfus, "BEAM: A Burst-Aware Energy-Efficient Adaptive MAC Protocol for Wireless Sensor Networks." International Conference on Networked Sensing Systems (INSS), Kassel, Germany, May 2009, pp. 61–72.

[6] Atmel, "AVR 8-bit Microcontrollers: Datasheets, Specifications and Reference Manuals." [Online]. Available: http://www.atmel.com/products/avr/

[7] N. I. Australia, "Castalia - a simulator for Wireless Sensor Networks." [Online]. Available: http://castalia.npc.nicta.com.au

[8] M. Baar, E. Koeppe, A. Liers, and J. Schiller, "The ScatterWeb MSB-430 Platform for Wireless Sensor Networks." SICS Contiki Workshop, Kista, Sweden, March 2007.

[9] S. Barthlomé, "Investigating Forward Error Correction Strategies on MSB430 Sensor Nodes." Master Thesis, University of Bern, Switzerland, May 2011.

[10] U. Berkeley., "Crossbow TelosB mote (TPR2400)." [Online]. Available: www.xbow.com/pdf/Telos_PR.pdf

[11] J. Beutel, M. Dyer, O. Kasten, M. Ringwald, and K. Römer, "BTnodes - A Distributed Environment for Prototyping Ad Hoc Networks." [Online]. Available: www.btnode.ethz.ch

[12] T. Braun, T. Voigt, and A. Dunkels, "TCP Support for Sensor Networks." Wireless On demand Network Systems and Services (WONS), Obergurgl, Austria, January 2007, pp. 162–169.

[13] M. Buettner, V. Gary, E. Anderson, and R. Han, "X-MAC: A Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks," in *in SenSys*. ACM Conference on Embedded Networked Sensor Systems (SenSys), Boulder, USA, November 2006, pp. 307–320.

[14] M. Busse, T. Haenselmann, T. King, and W. Effelsberg, "The Impact of Forward Error Correction on Wireless Sensor Network Performance." ACM Workshop on Real-World Wireless Sensor Networks (REALWSN), Uppsala, Sweden, June 2006.

[15] R. G. C. Intanagonwiwat and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proceedings of the sixth annual international conference on Mobile computing and networking*, Boston, MA USA, 2000, pp. 56–67.

[16] C. Campbell, *Surface Acoustic Wave Devices for Mobile and Wireless Communication*. Academic Press Inc, 1998.

[17] Crossbow Technologies. [Online]. Available: http://www.xbow.com

[18] W. Dargie and C. Poellabauer, *Fundamentals of Wireless Sensor Networks: Theory and Practice*. John Wiley & Sons, 2010.

[19] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proceedings of the 9th ACM International Conference on Mobile Computing and Networking (MobiCom 03)*, San Diego, California, September 2003.

[20] Deliverable D4.1: First Set of well-designed Simulations, "Experiments and possible Benchmarks. Technical Report," June 2008. [Online]. Available: http://www.wisebed.eu

[21] A. Dunkels, "Full TCP/IP for 8-Bit Architectures." International Conference on Mobile Systems, Applications, and Services (MobiSys), San Francisco, USA, May 2003, pp. 85–98.

[22] A. Dunkels, J. Alonso, T. Voigt, and H. Ritter, "Distributed TCP Caching for Wireless Sensor Networks." Mediterranean Ad-Hoc Networks Workshop (Med-Hoc-Net), Bodrum, Turkey, June 2004, pp. 13–28.

[23] A. Dunkels, B. Groenvall, and T. Voigt, "Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors." IEEE Workshop on Embedded Networked Sensors (EmNets), Tampa, Florida, November 2004, pp. 455–462. [Online]. Available: http://www.sics.se/contiki/

[24] A. Dunkels, L. Mottola, N. Tsiftes, F. Osterlind, J. Eriksson, and N. Finne, "The Announcement Layer: Beacon Coordination for the Sensornet Stack." European Conference on Wireless Sensor Networks (EWSN), Bonn, Germany, February 2011, pp. 211–226.

[25] A. Dunkels, F. Österlind, and Z. He, "An Adaptive Communication Architecture for Wireless Sensor Networks." ACM Conference on Embedded Networked Sensor Systems (SenSys), Sydney, Australia, November 2007, pp. 335–349.

[26] A. Dunkels, "The contikimac radio duty cycling protocol," 2011.

[27] A. Dunkels, J. Alonso, T. Voigt, H. Ritter, and J. H. Schiller, "Connecting wireless sensornets with tcp/ip networks," in *WWIC'04*, 2004, pp. 143–152.

[28] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based on-line energy estimation for sensor nodes," in *Proceedings of the Fourth Workshop on Embedded Networked Sensors (Emnets IV)*, Cork, Ireland, Jun. 2007. [Online]. Available: http://www.sics.se/~adam/dunkels07softwarebased.pdf

[29] A. El-Hoiydi and J. D. Decotignie, "WiseMAC: An Ultra Low Power MAC Protocol for Multihop Wireless Sensor Networks." International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS), Turku, Finland, July 2004, pp. 18–31.

[30] E. Ertin, A. Arora, R. Ramnath, and M. Nesterenko, "Kansei: A Testbed For Sensing At Scale." ACM/IEEE International Conference on Information Processing In Sensor Networks (IPSN), Nashville, Tennessee, USA, April 2006, pp. 399–406.

[31] S. O. S. F. U. B. . S. GmbH. [Online]. Available: http://scatterweb.mi.fu-berlin.de/

[32] T. A. Gulliver and V. K. Bhargava, "A Systematic (16,8) Code for Correcting Double Errors and Detecting Triple-Adjacent Errors," *IEEE Transactions on Computers*, vol. 42, January 1993.

[33] R. Hamming, "Error Detecting and Error Correcting Codes," vol. 26, no. 2, pp. 147–160, April 1950.

[34] V. Handziski, A. Koepke, A. Willig, and A. Wolisz, "TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor

Network." ACM/SIGMOBILE International Workshop on Multi-hop Ad Hoc Networks (REALMAN), Florence, Italy, May 2006.

[35] HART communication. [Online]. Available: http://www.hartcomm.org

[36] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building efficient wireless sensor networks with low-level naming," 2001.

[37] P. Horowitz and W. Hill, *The Art of Electronics*. Cambridge University Press;, 1989.

[38] P. Hurni, M. Anwander, and G. Wagenknecht, "TARWIS - Testbed Management Architecture for Wireless Sensor Network Testbeds." [Online]. Available: http://rvs.unibe.ch/research/software.html

[39] P. Hurni and T. Braun, "Real-World Experiences with the Maximally Traffic Adaptive Medium Access Control Protocol." Technical Report IAM-11-001, Institute of Computer Science and Applied Mathematics, University of Bern, Switzerland, June 2011.

[40] P. Hurni and S. Barthlomé, "libECC: An Open-Source Library of Error Correcting Codes (ECCs) for the MSP430 Microcontroller." [Online]. Available: http://rvs.unibe.ch/research/software.html

[41] P. Hurni, S. Barthlomé, and T. Braun, "Link-Quality Aware Run-Time Adaptive Forward Error Correction Strategies in Wireless Sensor Networks." European Conference on Wireless Sensor Networks (EWSN), Trento, Italy, submitted, February 2012.

[42] P. Hurni, U. Bürgi, and M. A. T. Braun, "TCP Performance Optimizations for Wireless Sensor Networks." European Conference on Wireless Sensor Networks (EWSN), Trento, Italy, submitted, February 2012.

[43] IEEE, "Ieee standard for information technology- telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements-part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-1997*, pp. i –445, 1997.

[44] IEEE, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (ANSI/IEEE Std 802.11, 1999 Edition (R2003))*, Institute of Electrical and Electronics Engineers, Inc., Jun. 2003.

[45] IEEE, *IEEE 1 Std. 802.15.1, IEEE Standard for Information Technology Local and Metropolitan Area Networks Specific Requirements Part 15.1*, Institute of Electrical and Electronics Engineers, Inc., 2005.

[46] ISA. [Online]. Available: http://www.isa.org

[47] V. K. John Bard, *Software Defined Radio*. John Wiley & Sons, 2007.

[48] Kalsi, *Electronic Instrumentation*. Mcgraw Hill Higher Education, 2010.

[49] B. Krishnamachari, D. Estrin, and S. B. Wicker, "The impact of data aggregation in wireless sensor networks," in *22nd International Conference on Distributed Computing Systems, Workshops (ICDCSW 02) July 2-5, 2002, Vienna, Austria, Proceedings*. IEEE Computer Society, 2002, pp. 575–578.

[50] P. Kumar, M. Günes, Q. Mushtaq, and J. Schiller, "Optimizing duty-cycle for delay and energy bound wsn applications," in *Proceedings of the 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, ser. WAINA '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 692–697.

[51] P. Kumar, M. Günes, Q. Mushtaq, and B. Blywis, "A real-time and energy-efficient mac protocol for wireless sensor networks," in *6th IEEE International Conference on Wireless and Optical Communications Networks*, Cairo, Egypt, April 2009.

[52] H. Lee, A. Cerpa, and P. Levis, "Improving wireless simulation through noise modeling," in *In IPSN 07: Proceedings of the 6th international conference on Information processing in sensor networks*. ACM Press, 2007, pp. 21–30.

[53] C.-J. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis, "Surviving Wi-fi Interference in Low Power ZigBee Networks." ACM Conference on Embedded Networked Sensor Systems (SenSys), Zurich, Switzerland, November 2010, pp. 309–322.

[54] Y.-D. Lin, J.-H. Yeh, T.-H. Yang, C.-Y. Ku, S.-L. Tsao, and Y.-C. Lai, "Efficient dynamic frame aggregation in ieee 802.11s mesh networks," *Int. J. Commun. Syst.*, vol. 22, no. 10, pp. 1319–1338, Oct. 2009. [Online]. Available: http://dx.doi.org/10.1002/dac.v22:10

[55] Y. Lin and V. W. S. Wong, "Frame aggregation and optimal frame size adaptation for ieee 802.11n wlans," in *GLOBECOM'06*, 2006, pp. –1–1.

[56] R. R. R. M. Zorzi, "Is TCP energy efficient?" IEEE International Workshop on Mobile Multimedia Communications (MoMuC'99): San Diego, California, USA, November 1999, pp. 198–201.

[57] M. A. Mahmood and W. Seah, "Reliability in Wireless Sensor Networks: Survey and Challenges Ahead." School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand, April 2006.

[58] L. A. N. Man and S. Committee, "Ieee std 802.15.4c-2009 (amendment to ieee std 802.15.4-2006) ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 15.4: Wireless lan medium ac," *IEEE Std 802154c2009 Amendment to IEEE Std 8021542006*, vol. 2009, no. April 2009, pp. 1–21.

[59] Matthias Ringwald and Kay Roemer, "BitMAC: a deterministic, collision-free, and robust MAC protocol for sensor networks," in *EWSN'05*, 2005, pp. 57–69.

[60] E. McCune, *Practical Digital Wireless Signals*. Cambridge University Press, 2010.

[61] R. Morelos-Zaragoza, *The Art of Error Correcting Coding*. Second Edition, John Wiley and Sons, 2006.

[62] MoteWeb: Harvard Sensor Network Testbed Software. [Online]. Available: http://motelab.eecs.harvard.edu/

[63] L. Mottola, G. P. Picco, M. Ceriotti, c. Gună, and A. L. Murphy, "Not all wireless sensor networks are created equal: A comparative study on tunnels," *ACM Trans. Sen. Netw.*, vol. 7, no. 2, pp. 15:1–15:33, Sep. 2010. [Online]. Available: http://doi.acm.org/10.1145/1824766.1824771

[64] R. Musaloiu, C. Liang, and A. Terzis, "Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks." ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), St. Louis, USA, April 2008, pp. 421–432.

[65] M. Z. H. Y. C. A. B. Nouha Baccour, Anis Koubaa and M. Alves, "Radio link quality estimation in wireless sensor networks: a survey," in *In ACM Transactions on Sensor Networks. Volume 8, Issue 4*. ACM, November 2012.

[66] P. Hurni and T. Braun, "MaxMAC: a Maximally Traffic-Adaptive MAC Protocol for Wireless Sensor Networks." European Conference on Wireless Sensor Networks (EWSN), Coimbra, Portugal, February 2010, pp. 289–305.

[67] C. Pazos, J. Sanchez Agrelo, and M. Gerla, "Using back-pressure to improve tcp performance with many flows," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, mar 1999, pp. 431 –438 vol.2.

[68] W. W. Peterson and E. J. Weldon, *Error-Correcting Codes*. MIT Press, 1998.

[69] M. Petrova, L. Wu, P. Mahonen, and J. Riihijarvi, "Interference measurements on performance degradation between colocated ieee 802.11g/n and ieee 802.15.4 networks," in *Networking, 2007. ICN '07. Sixth International Conference on*, april 2007, p. 93.

[70] I. Reed and G. Solomon, "Polynomial Codes over certain finite Fields," *Society for Industrial and Applied Mathematics (SIAM) Journal*, vol. 8, no. 2, pp. 300–304, June 1960.

[71] RF Monolithics Incorporated: TR1001 868.35 MHz Hybrid Transceiver. [Online]. Available: www.rfm.com/products/data/tr1001.pdf

[72] RIGOL, "Digital Multimeter DM3052," 2012. [Online]. Available: http://www.rigol.com/prodserv/DigitalMultimeters/

[73] P. Roshan, *Wireless Local-area Network Fundamentals*. Macmillan Technical Publishing, 2003.

[74] A. Saif, M. Othman, S. Subramaniam, and N. Hamid, "An enhanced a-msdu frame aggregation scheme for 802.11n wireless networks," in *Wireless Personal Communications*. Springer US, 2011, pp. 1–24.

[75] J. Schiller, A. Liers, H. Ritter, R. Winter, and T. Voigt, "ScatterWeb - Low Power Sensor Nodes and Energy Aware Routing." Hawaii International Conference on System Sciences (HICSS), Hawaii, USA, January 2005, pp. 1–9.

[76] J. H. Schiller, A. Liers, and H. Ritter, "ScatterWeb: A Wireless Sensornet Platform for Research and Teaching," *Elsevier Computer Communications*, vol. 28, pp. 1545–1551, August 2005.

[77] T. Selvam and S. Srikanth, "A frame aggregation scheduler for ieee 802.11n," in *Communications (NCC), 2010 National Conference on*, 2010, pp. 1–5.

[78] Sentilla. [Online]. Available: www.sentilla.com

[79] Seventh Framework Programme FP7 - Information and Communication Technologies, "Wireless Sensor Networks Testbed Project (WISEBED)," FP7 Project 2008-2011. [Online]. Available: http://www.wisebed.eu

[80] A. Sikora and V. Groza, "Coexistence of ieee802.15.4 with other systems in the 2.4 ghz-ism-band," in *Instrumentation and Measurement Technology Conference, 2005. IMTC 2005. Proceedings of the IEEE*, vol. 3, may 2005, pp. 1786 –1791.

[81] H. Sizun, *Radio Wave Propagation for Telecommunication Applications*. Springer Berlin Heidelberg, 2004.

[82] D. R. Smith, *Digital Transmission Systems*. Springer US, 2003.

[83] D. Son, B. Krishnamachari, and J. Heidemann, "Experimental study of concurrent transmission in wireless sensor networks," in *Proceedings of the 4th international conference on Embedded networked sensor systems*, ser. SenSys '06. New York, NY, USA: ACM, 2006, pp. 237–250. [Online]. Available: http://doi.acm.org/10.1145/1182807.1182831

[84] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis, "An empirical study of low-power wireless," *ACM Trans. Sen. Netw.*, vol. 6, no. 2, pp. 16:1–16:49, Mar. 2010. [Online]. Available: http://doi.acm.org/10.1145/1689239.1689246

[85] K. Srinivasan, M. Jain, J. I. Choi, T. Azim, E. S. Kim, P. Levis, and B. Krishnamachari, "The κ factor: inferring protocol performance using inter-link reception correlation," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, ser. MobiCom '10. New York, NY, USA: ACM, 2010, pp. 317–328. [Online]. Available: http://doi.acm.org/10.1145/1859995.1860032

[86] F. Stann and J. Heidemann, "Rmst: Reliable data transport in sensor networks," in *Proceedings of the First International Workshop on Sensor Net Protocols and Applications*, Anchorage, Alaska, USA, 2003, pp. 102–112. [Online]. Available: 049

[87] P. Suarez, C. Renmarker, T. Voigt, and A. Dunkels, "Increasing ZigBee network lifetime with X-MAC." ACM Workshop on Real-World Wireless Sensor Network (REALWSN), Glasgow, Scotland, November 2008, pp. 13–18.

[88] Texas Instruments, "16-Bit Ultra-Low Power MSP430 Microcontrollers: Datasheets, Specifications and Reference Manuals." [Online]. Available: http://focus.ti.com/

[89] Texas Instruments CC1000: Single Chip Very Low Power RF Transceiver. [Online]. Available: http://www.ti.com/lit/gpn/cc1000

[90] Texas Instruments CC1020: Single-Chip FSK/OOK CMOS RF Transceiver. [Online]. Available: http://www.ti.com/lit/gpn/cc1020

[91] Texas Instruments CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. [Online]. Available: http://www.ti.com/product/cc2420

[92] J. Thelen and D. Goense, "Radio wave propagation in potato fields," in *In 1st Workshop on Wireless Network Measurements*, 2005.

[93] TWISTv1: The TWIST testbed software suite, 2011. [Online]. Available: http://www.twist.tu-berlin.de/wiki/TWIST/Software/TWISTv1

[94] L. H. van and P. Havinga, "A lightweight medium access protocol (lmac) for wireless sensor networks: Reducing preamble transmissions and transceiver state switches," in *1st International Workshop on Networked Sensing Systems, INSS 2004*. Tokio, Japan: Society of Instrument and Control Engineers (SICE), 2004, pp. 205–208. [Online]. Available: http://doc.utwente.nl/64756/

[95] A. Varga, "INET Framework, an open-source communication networks simulation package for the OMNeT++ simulation environment." [Online]. Available: http://inet.omnetpp.org/

[96] VOLTCRAFT, "Power supply VLP-1303 PRO," 2012. [Online]. Available: http://www.voltcraft.ch/index.php?site=netztechnik

[97] G. Wagenkecht, M. Anwander, and T. Braun, "Hop-to-Hop Reliability in IP-based Wireless Sensor Networks - a Cross-Layer Approach." International Conference on Wired/Wireless Internet Communications 2009 (WWIC'09) , Enschede, The Netherlands, June 2010, pp. 195–202.

[98] G. Wagenknecht, M. Anwander, and T. Braun, "MARWIS: A Management Platform for Heterogeneous Wireless Sensor Network." Ercim News, Vol. 5031/2008, Nr. 76, January 2009, pp. 18–19.

[99] G. Werner-Allen, P. Swieskowski, and M. Welsh, "MoteLab: a Wireless Sensor Network Testbed." ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Los Angeles, USA, April 2005, pp. 483–488.

[100] D. Yang, Y. Xu, and M. Gidlund, "Coexistence of ieee802.15.4 based networks: A survey," in *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*, nov. 2010, pp. 2107 –2113.

[101] W. Ye, J. Heidemann, and D. Estrin, "An Energy Efficient MAC Protocol for Wireless Sensor Networks." IEEE International Conference on Computer Communications (INFOCOM), New York, USA, June 2002, pp. 1567–1576.

[102] Yifeng Yang, "AN1066 MiWi Wireless Networking Protocol Stack, describing the Microchip MiWi stack," 2009. [Online]. Available: http://ww1.microchip.com/downloads/en/appnotes/01066a.pdf

[103] ZigBee Alliance. [Online]. Available: http://www.zigbee.org

List of Publications

Refereed Papers (Journals, Conferences, Workshops)

- M. Anwander and T. Braun, "A Reliable, Traffic-adaptive and Energy-efficient Link Layer for Wireless Sensor Networks," *Submitted for Publication to European Conference on Wireless Sensor Networks (EWSN), Ghent, Belgium*, February 2013.

- S. Morgenthaler, T. Braun, Z. Zhao, T. Staub, M. Anwander, "UAVNet: A Mobile Wireless Mesh Network Using Unmanned Aerial Vehicles," in *3rd International Workshop on Wireless Networking and Control for Unmanned Autonomous Vehicles* , Anaheim, CA, USA, December 3 - 7, 2012

- M. Anwander, T. Braun, A. Jamakovic, T. Staub, "Authentication and Authorisation Mechanisms in support of Secure Access to WMN Resources," in *The Fourth IEEE International Workshop on Hot Topics in Mesh Networking (HOTMESH)*, San Francisco, USA, June 25, 2012, IEEE, ISBN 978-1-4673-1239-4

- G. Wagenknecht, M. Anwander, T. Braun, "Performance Evaluation of Reliable Overlay Multicast in Wireless Sensor Networks," in *10th International Conference on Wired/Wireless Internet Communications (WWIC)*, Santorini, Greece, June 6 - 8, 2012, pp. 114-125, Springer Berlin Heidelberg, ISBN 978-3-642-30629-7

- P. Hurni, M. Anwander, G. Wagenknecht, T. Staub, T. Braun, "TARWIS - A testbed management architecture for wireless sensor network testbeds," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Maui, Hawaii, USA, April 16 - 20, 2012, pp. 611-614, IEEE Xplore, ISBN 978-1-4673-0267-8

- A. Jamakovic, M. Anwander, T. Braun, P. Kropf, E. Schiller, J. Schwanbeck, T. Staub, "A4-Mesh: Connecting Remote Sites," in *Switch Journal*, March, 2012, pp. 15-17, Switch

- P. Hurni, U. Buergi, M. Anwander, T. Braun, "TCP Performance Optimizations for Wireless Sensor Networks," in *9th European Conference on Wire-*

less Sensor Networks, Trento, Italy, February 15 - 17, 2012, pp. 17-32, Springer-Verlag Berlin, Heidelberg, ISBN 978-3-642-28168-6

- G. Wagenknecht, M. Anwander, T. Braun, "SNOMC: An Overlay Multicast Protocol for Wireless Sensor Networks," in *9th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, Courmayeur, Italy, January 9 - 11, 2012, pp. 75-78, IEEE Xplore, ISBN 978-1-4577-1721-5

- G. Coulson, B. Porter, I. Chatzigiannakis, C. Koninis, S. Fischer, D. Pfisterer, D. Bimschas, T. Braun, P. Hurni, M. Anwander, G. Wagenknecht, S. Fekete, A. Kroeller, T. Baumgartner, "Flexible Experimentation in Wireless Sensor Networks," in *Communications of the ACM*, Vol. 55, Nr. 1, January, 2012, pp. 82-90, ACM New York, USA, ISSN 0001-0782

- P. Hurni, M. Anwander, G. Wagenknecht, T. Staub, and T. Braun, "TARWIS - A Testbed Management Architecture for Wireless Sensor Network Testbeds," in *7th International Conference on Network and Service Management (CNSM 2011)*, Paris, France, October 24-28, 2011, submitted for publication.

- M. Anwander, G. Wagenknecht, T. Braun, K. Dolfus, "BEAM: A Burst-Aware Energy-Efficient Adaptive MAC Protocol for Wireless Sensor Networks," in *Seventh International Conference on Networked Sensing Systems*, Kassel, Germany, June 15 - 18, 2010, pp. 195-202, IEEE Press, ISBN 978-1-4244-7909-2

- T. Staub, M. Anwander, K. Baumann, T. Braun, M. Brogle, K. Dolfus, C. Félix, and P. K. Goode, "Connecting Remote Sites to the Wired Backbone by Wireless Mesh Access Networks," in *16th European Wireless Conference*, Lucca, Italy, April 12 - 15, 2010, pp. 675 - 682, IEEE Xplore, ISBN 978-1-4244-5999-5.

- T. Staub, M. Anwander, K. Baumann, T. Braun, M. Brogle, P. Dornier, C. Félix, and P. K. Goode, "Wireless Mesh Networks - Connecting Remote Sites," in *SWITCH Journal*, Zurich, Switzerland, March, 2010, pp. 10-12.

- P. Hurni, G. Wagenknecht, M. Anwander, T. Braun, "A Testbed Management Architecture for Wireless Sensor Network Testbeds (TARWIS)," in *7th European Conference on Wireless Sensor Networks (EWSN)*, Coimbra, Portugal , February 17 - 19, 2010, pp. 33-35, Springer, ISBN 978-989-96001-3-3

- G. Wagenknecht, M. Anwander, T. Braun, "Hop-to-Hop Reliability in IP-based Wireless Sensor Networks - a Cross-Layer Approach," in *International Conference on Wired/Wireless Internet Communications 2009 (WWIC'09)*, Enschede, The Netherlands, Vol. 5546/2009, May 27 - 29, 2009, pp. 61-72, Springer LNCS, ISBN 978-3-642-02117-6

- P. Hurni, T. Braun, M. Anwander, "Evaluation of WiseMAC and Extensions on Wireless Sensor Nodes," in *Springer Telecommunication Systems Journal*, Vol. 43, Nr. 1-2, September 4, 2009, Springer US, ISSN 1018-4864

- P. Hurni, T. Staub, G. Wagenknecht, M. Anwander, and T. Braun, "A Secure Remote Authentication, Operation and Management Infrastructure for Distributed Wireless Sensor Network Testbeds," in *1st Workshop on Global Sensor Networks (GSN'09) co-located with KiVS'09*, Kassel, Germany, Vol. 17, March 6 - 7, 2009, pp. 1-6, Electronic Communications of the EASST, ISSN 1863-2122.

- G. Wagenknecht, M. Anwander, T. Braun, "MARWIS: A Management Platform for Heterogeneous Wireless Sensor Networks," in *Ercim News*, Vol. 5031/2008 , Nr. 76, January, 2009, pp. 18-19, ERCIM EEIG, ISSN 0926-4981

- G. Wagenknecht, M. Anwander, Marc Brogle, T. Braun, "Reliable Multicast in Wireless Sensor Networks," in *7. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze*, Berlin, Germany, September 25 - 26, 2008, pp. 69-72, Freie Universitaet Berlin, Fachbereich Mathematik und Informatik, Tech. Report B 08-12

- M. Anwander, G. Wagenknecht, T. Braun, "Management of Wireless Sensor Networks using TCP/IP," in *International Workshop on Sensor Network Engineering (IWSNE) at the 4th IEEE/ACM International Conference on Distributed Computing in Sensor Systems*, Santorini Island, Greece, June 11, 2008, pp. II.1-II.8, ISBN 978-90-9023209-6

- G. Wagenknecht, M. Anwander, T. Braun, T. Staub, J. Matheka, and S. Morgenthaler, "MARWIS: A Management Architecture for Heterogeneous Wireless Sensor Networks," in *6th International Conference on Wired/Wireless Internet Communications (WWIC'08)*, Tampere, Finland, Vol. LCNS, Nr. 5031, May 28 - 30, 2008, pp. 177-188, Springer, ISBN 978-3-540-68805-1.

- M. Anwander, G. Wagenknecht, T. Staub, and T. Braun, "Management of Heterogenous Wireless Sensor Networks," in *6. Fachgespräch 'Drahtlose Sensornetze' der GI/ITG-Fachgruppe 'Kommunikation und Verteilte Systeme'*, Aachen, Germany, July 16 - 17, 2007, pp. 63-66, Distributed Systems Group, RWTH Aachen University, ISSN 0935-3232.

- T. Staub, T. Bernoulli, M. Anwander, M. Wälchli, and T. Braun, "Experimental Lifetime Evaluation for MAC Protocols on Real Sensor Hardware," in *ACM Workshop on Real-World Wireless Sensor Networks (REALWSN'06)*, Uppsala, Sweden, June 19, 2006, pp. 25-29, ACM Press, ISBN 1-59593-431-6.

Books

- T. Braun, M. Anwander, P. Hurni, M. Waelchli, "MAC Protocols for Wireless Sensor Networks, Next Generation Mobile Networks and Ubiquitous Computing" Hershey, New York, USA, October, 2010, pp. 165 - 174, IGI Global, ISBN 978-1-60566-250-3

Unrefereed Papers (Technical Reports, Project Deliverables)

- G. Coulson, G. Wagenknecht, M. Anwander, et al., "Report on the Integration of the Software Infrastructure," WISEBED Deliverable D2.3, June, 2010

- T. Staub, M. Anwander, M. Brogle, K. Dolfus, T. Braun, K. Baumann, C. Félix, and P. Dornier, "Wireless Mesh Networks for Interconnection of Remote Sites to Fixed Broadband Networks (Feasibility Study)," Universität Bern, Institut für Informatik und angewandte Mathematik, Bern, Switzerland, December 18, 2009, IAM-09-007.

- M. Brogle, S. Serbu, D. Milic, M. Anwander, P. Hurni, C. Spielvogel, C. Fautsch, D. Harmanci, L. Charles, H. Sturzrehm, G. Wagenknecht, T. Braun, T. Staub, C. Latze, and R. Standtke, "BeNeFri Summer School 2009 on Dependable Systems," Münchenwiler, Switzerland, September 8, 2009, IAM-09-006.

- M. Brogle, D. Milic, M. Anwander, G. Wagenknecht, M. Wälchli, T. Braun, R. Kummer, M. Wulff, R. Standtke, H. Sturzrehm, E. Riviere, P. Felber, S. Krenn, C. Ehret, C. Latze, P. Hurni, and T. Staub, "BeNeFri Summer School 2008 on Dependable Systems," Quarten, Switzerland, November 18, 2008, IAM-08-003.

- T. Braun, U. Ultes-Nitsche, M. Brogle, D. Milic, P. Lauer, T. Staub, G. Wagenknecht, M. Anwander, M. Wälchli, M. Wulff, C. Latze, M. Hayoz, C. Ehret, and T. Nicola, "RVS Retreat 2007," Quarten, Switzerland, December, 2007, IAM-07-004.

- M. Anwander, G. Wagenknecht, T. Braun, "Sensor Node Platform and Middleware for Management of Wireless Sensor Networks,"June 19, 2007, iam-07-003

- M. Anwander, G. Wagenknecht, T. Braun, "Energy-efficient Management of Heterogeneous Wireless Sensor Networks," April 30, 2007, IAM-07-002

Erklärung

gemäss Art. 28 Abs. 2 RSL 05

Name/Vorname: Anwander Markus

Matrikelnummer: 00-102-921

Studiengang: Informatik

Bachelor ☐ Master ☐ Dissertation ☒

Titel der Arbeit: Reliability in Energy Efficient Wireless Sensor Networks

LeiterIn der Arbeit: Prof. Dr. Torsten Braun

Ich erkläre hiermit, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andernfalls der Senat gemäss Artikel 36 Absatz 1 Buchstabe o des Gesetztes vom 5. September 1996 über die Universität zum Entzug des auf Grund dieser Arbeit verliehenen Titels berechtigt ist.

Bern, 15.11.2012

Ort/Datum

.....

Unterschrift

Curriculum Vitae

Personal Details

Name	Markus Anwander
Date of Birth	December 29, 1974
Address	Wilkerstr 24
	CH-3097 Liebefeld, Switzerland
Hometown	Untereggen SG, Switzerland
Nationality	Swiss

Education

2006 – 2012	Ph.D. student in Computer Science at the University of Bern, Switzerland
2006	Master of Science in Computer Science, University of Bern, Switzerland
2000 – 2006	Study of Computer Science at the University of Bern, minor fields in Mathematics and Communication science
1996 – 2000	General qualification for university entrance, IMSE Sargans, Switzerland
1991 – 1994	Apprenticeship as chemical laboratory worker, Ivoclar, Liechtenstein