Autonomous Framework for Supporting Energy Efficiency and Communication Reliability for Periodic Data Flows in Wireless Sensor Networks

OSAMA KHADER

Autonomous Framework for Supporting Energy Efficiency and Communication Reliability for Periodic Data Flows in Wireless Sensor Networks

vorgelegt von
Osama Khader
M.Sc. in Computer Engineering

von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
– Dr.-Ing. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr.-Ing. Thomas Sikora
Berichter:	Prof. Dr.-Ing. Adam Wolisz
Berichter:	Prof. Dr. Pedro Marron
Berichter:	Prof. Dr. Thiemo Voigt

Tage der wissenschaftlichen Aussprache: 5. Juni 2014

Berlin 2014

D83

Abstract

In this thesis we propose, design, and evaluate a novel decentralized, adaptive and self-learning framework to efficiently support low-rate periodic traffic applications in wireless sensor networks (WSNs). In many application areas like: environmental monitoring, building automation, industrial automation, aerospace applications wirelessly connected sensor nodes are expected to periodically report their sensor readings. One of the most widely used solutions for such applications is WirelessHART, featuring multi-hop configurations with strictly time- synchronized slotted operation and frequency hopping as its basic mechanisms. High packet delivery ratio is achieved, as the frequency diversity provides high robustness against frequency selective fading and interference, while time slotting provides the possibility to push nodes into power saving modes for most of the time (in fact, all the time while they are not scheduled to transmit or receive). A well-known disadvantage of this approach is the necessity of complex schedule computations and its implications (i.e. the complexity of deriving and deploying new schedules in case of traffic changes, such that changes of traffic periodicity).

This issue was the main motivation of this work. Traffic rates may change frequently due to the changes in the monitored environment (e.g. light intensity changes between day and night, with most traffic occurring during twilight times where the rate of change is the largest), which implies the need of an agile adaptive traffic policy. The key question is therefore, how energy efficient operation (i.e. high level of sleeping) can be achieved under such variable traffic conditions - while ensuring low delay and high packet delivery ratio. In our approach, no explicit time synchronization among the nodes is needed. Each traffic relaying node (forwarder) starts its operation with excessively long wake-up periods and estimates the suitable times for more aggressive sleeping and shorter wake- up periods by computing the statistical parameters of the passing traffic. Naturally, if the traffic periodicity and/or jitter changes, the node will properly adapt to the new situation. This basic approach has been extended to efficiently support numerous, and possibly intersecting packet flows. In full understanding of the reliability advantages offered by the frequency hopping

solution in an equally autonomous way (i.e. without explicit signaling), this feature has also been incorporated into our design.

For the purpose of performance evaluation, the newly developed approach and WirelessHART have been extensively simulated (using realistic channel traces available in literature); in addition measurements of real implementation have also been performed. The solution has been shown to achieve its major goal - flexibly and efficiently following the traffic changes. Our approach, parameterized for minimum energy consumption, could be proven as clearly superior to WirelessHART in this metric, as well as in end-to-end delay under several test scenarios. Such aggressive power-saving characteristic has resulted in a slightly worse packet delivery ratio. However, different parameterizations allowing balancing different power saving modes vs. delivery ratios are also possible.

Zusammenfassung

In dieser Arbeit entwickeln und untersuchen wir einen neuartigen dezentralen, selbstlernenden und adaptiven Ansatz, um hohe Zuverlässigkeit, geringe Latenz und Energieeffizienz für niederratigen periodischen Datenverkehr in drahtlosen Sensornetzen zu erreichen. Mit diesem Ansatz zielen wir auf Anwendungen in den Feldern der Umweltbeobachtung, Gebäudeautomation, Industrieautomatisierung, oder Anwendungen in der Luft- und Raumfahrt. In diesen Anwendungsklassen werden oft eine Vielzahl von Sensorknoten in einem vorgegebenen räumlichen Gebiet plaziert um periodisch eine Vielzahl physikalischer Signale zu übertragen.

Viele Sensornetze haben bestimmte Beschränkungen gemein. Ein Beispiel ist die relative geringe Übertragungsreichweite eines Sensorknotens, welche hilft, seine Lebensdauer zu verlängern. Eine andere wichtige Eigenschaft folgt aus der grossen Anzahl der Sensorknoten in vielen Anwendungen. Sensorknoten können dem Netz zu beliebigen Zeitpunkten beitreten oder es verlassen, und die drahtlosen Verbindungen zwischen den Knoten können sich über die Zeit verändern. Somit sollten die Kommunikationsprotokolle selbstorganisierend sein und ohne menschlichen Eingriff auskommen. Weiterhin sollten Protokolle für Sensornetze keine übermässigen Anforderungen in puncto Rechenleistung oder Speicherverbrauch stellen.

Darüber hinaus wird die Zuverlässigkeit der Kommunikation in Sensornetzen durch Phänomene wie Mehrwegeschwund und schmalbandige Interferenz eingeschränkt. Geringe Zuverlässigkeit führt zu Paketverlusten, was wiederum energetisch aufwändige Übertragungswiederholungen und grössere Latenzen nach sich zieht. Ausserdem haben Sensorknoten unterschiedliche Senderaten und diese können sich mit der Zeit ändern. Ein möglicher Grund für geänderte Senderaten sind Veränderungen in der Umwelt (zum Beispiel senden Helligkeitssensoren mehr Daten während der Dämmerung, weil die Veränderungsrate dann am grössten ist). Diese Variabilität in den Senderaten verlangt nach einer Adaption des Netzes.

Eine Schlüsselfrage ist somit, wie (energiesparende) Schlafaktivitäten am besten unterstützt werden können, während das Netz gleichzeitig eine hohe

Kommunikationszuverlässigkeit, geringe Latenzen, und eine gute Adaptivität an Laständerungen aufweist.

In dieser Arbeit entwickeln und untersuchen wir eine neuartige Lösung, um diese Ziele zu erreichen. Im Gegensatz zu aktuellen TDMA-basierten Systemen wie WirelessHART (welches einigen Verwaltungsaufwand für Zeitsynchronisation und andere Zwecke betreibt) benötigt unser Ansatz keine aufwändigen Management-Protokolle, sondern benutzt ausschliesslich die Periodizität des eigentlichen Datenverkehrs.

Zwei wesentliche Ideen werden eingeführt. Erstens: Sensorknoten, die Pakete weiterleiten tauschen keine Informationen über die Periode und über die Zeitbasis aus, sondern schätzen die Periode und die durchschnittlichen Schwankungen derselben direkt aus Beobachtungen des Datenverkehrs. Darauf basierend bestimmt der Knoten dann die Zeiten zu denen er schlafen kann und zu denen er wach sein muss, um das nächste Paket zu empfangen. Zweitens: Alle Knoten des Sensornetzes (Quellen und Weiterleitungsknoten) wechseln für jedes neue Paket die Übertragungsfrequenz. Hierbei sind Quellenknoten unabhängig voneinander, d.h. sie wählen ihre eigenen Perioden und Kanäle autonom. Ein Knoten, der Pakete weiterleitet, benutzt die geschätzten Perioden dann auch, um zu bestimmen wann er als nächstes den Kanal wechseln muss.

Unser Ansatz erzielt gegenüber einem Vergleichssystem mit nur einem einzigen Kanal erhebliche Verbesserungen in der Energieeffizienz und der Zuverlässigkeit. Auch im Vergleich zu einem WirelessHART-basierten Vergleichssystem ist unser Ansatz in puncto Energie, Latenz und Adaptierbarkeit überlegen. Für unsere Auswertungen haben wir Experimente und realistische trace-basierte Simulationen benutzt, und zwar sowohl für unseren eigenen Ansatz als auch für die Vergleichssysteme.

Contents

List of Figures

List of Tables

List of Abbreviations

6LoWPAN Low power Wireless Personal Area Networks.

ABCH asynchronous blind channel hopping.

ACK Acknowledgement.

ANOVA Analysis of Variance.

ATIM Ad hoc Traffic Indication Message.

BER Bit Error Rate.

CCA Clear Channel Assessment.

CoRE Constrained RESTful Environments.

CSMA Carrier Sense Multiple Access.

CSMA/CA Carrier Sense Multiple Access/Collision Avoidance.

CTP Collection Tree Protocol.

CTS Clear To Send.

DSSS Direct Sequency Spread Spectrum.

FFD Full-Function Devices.

FPFF Fastest Periodic Flow First.

HART Highway Addressable Remote Transducer.

ID identity.

IEC International Electrotechnical Commission.

ISA International Society of Automation.

LPL Low-Power Listening.

LQI Link Quality Indicator.

LR-WPANs Low-Rate Wireless Personal Area Networks.

MAC Medium Access Control.

O-QPSK Offset Quadrature Phase-Shift Keying.

OS Operating System.

PAN Personal Area Network.

PDR Packet Delivery Ratio.

PS Power Saving Mode.

RFD Reduced-Function Devices.

RICER Receiver Initiated Cycled Receiver.

ROLL Routing Over Low-power and Lossy networks.

RSM Response Surface Methodology.

RSSI Received Signal Strength Indicator.

RTS Request To Send.

SNR Signal to Noise Ratio.

TDMA Time Division Multiple Access.

TI Texas Instruments, Inc..

TinyOS Tiny Operating System.

TKN Telecommunication Network Group.

TOA Time Of Arrival.

TRAMA TRaffic-Adaptive Medium Access protocol.

TSCH Time Slotted Channel Hopping.

TSMP Time Synchronized Mesh Protocol.

TWIST TKN WIreless Sensor network Testbed.

WSNs Wireless Sensor Networks.

Acknowledgements

First and foremost, I would like to express sincere gratitude to my research advisors: Prof. Adam Wolisz and Prof. Andreas Willig. This was the most rewarding experience of my PhD research to work under the guidance of these two wonderful people.

Prof. Wolisz has always been a guiding beacon throughout the graduate studies. His invaluable inputs and approach towards work have always been a constant source of motivation. I am deeply grateful to him for giving me the opportunity to pursue my PhD research in his TKN group.

I would also like to thank my co-advisor Prof. Andreas Willig at the University of Canterbury Christchurch, New Zealand, for his invaluable inputs and suggestions during my PhD research. Prof. Willig has been my idol not only in research work but also on other aspects of life. Every meeting with him was a step forward, leading to exciting problems to solve.

I deeply thank the members of the thesis committee: Prof. Thomas Sikora, Prof. Pedro Marron, and Prof. Thiemo Voigt for agreeing to review my dissertation and for their valuable feedback.

I would also like to acknowledge the German Academic Exchange Service (DAAD) and the Technical University of Berlin-Telecommunication Networks Group (TKN) for the financial support to complete my PhD thesis.

I also would like to thank my colleagues and friends at TKN: Dr. Vlado Handziski, Dr. Jan Haur, Dr. Andreas Kopke, Dr. Ahmad Rostami, Dr. Murad Abusubaih, Dr. Mathias Bohge, Dr. Daniel Willkomm, Dr. Anatolij Zubow, Dr. Lukasz Budzisz, Manoj Rege, Konstantin Miller, Filip Idzikowski, Thomas Menzel, Niels Karowski, Tacettin Ayar, Hieu Le, Michael Doring, Berthold Rathke, Onur Ergin, Mikolaj Chwalisz, Daniel Happ for helping and making my life in TKN easier and full of fun.

A special thanks to Petra Hutt for nourishing a great work environment. I also thank all TKN staff, especially the technical members: Georgios Ainatzes, Sven Spuida and Peter-Rene Schroter.

I also would like to mention my friends: Ashraf Emawi, Dr. Sid Ahmed Attia, Dr. Georges Haboub, and Dr. Abdel-Karim Al Tamimi, without whom

my life in Berlin would have been more difficult.

Leaving my home country has been difficult for me; however I recognize it has been even much harder on my family members. I want to show my love and gratitude to all of them, especially my parents: Hamza and Iftekhar, and my brothers and sisters for all their constant moral support throughout these years.

Finally, I wish to thank my family; my wife Seham for all her support and patience throughout this long and hard journey, my son Hamza and my daughter Sara. Hoping that they get inspired by me.

Dedication

Dedicated to:

- My wonderful parents: Hamza and Iftekhar,
- My lovely wife, Seham,
- My lovely daughter, Sara
- My lovely son, Hamza,
- My family members and friends.

الإهداء

إلى من علمني العطاء بدون انتظار .. إلى من أحمل أسمه بكل افتخار ..إلى القلب الكبير (والدي العزيز)

إلى من بوجودها أكتسب قوة ومحبة لا حدود لها... إلى القلب الناصع بالبياض (والدتي الحبيبة)

إلى نصفي الآخر و رفيقة دربي زوجتي الغالية

إلى أولادي الأحبــاء

إلى إخواني وأخواتي الأعزاء

إلى جميع الأهل والأصدقاء

CHAPTER 1

Introduction

Wireless Sensor Networks (WSNs) are composed of an inexpensive embedded devices – called sensor nodes – capable of sensing, computation, and communicating together in an ad hoc manner [7, 87, 21]. They have changed the way we interact with our physical world, since they allow computation very close to physical events of interest. WSNs are being extensively used in different domains ranging from monitoring environments, controlling our houses, cars, manufacturing plants [31, 181, 154], etc. Often sensor nodes are significantly constrained in terms of available memory, computational power and, most importantly, the amount of energy available to them. Due to these constraints, sensor networks often share certain characteristics. One example is the relatively short transmit range of a sensor node, which is useful to save energy. An immediate implication of the short transmission range is that many sensor networks applications are in fact multi-hop wireless networks. Therefore, intermediate nodes (or forwarders) are needed to relay packets on behalf of other nodes to the destination node, which in many scenarios is a centralized sink node. Another important characteristic can be justified from the envisioned large-scale in terms of numbers of sensor nodes in a WSN deployment. Sensor nodes may join, move, be switched off or leave the network, and the wireless links between them can experience substantial fluctuations, so the communication protocols should be self-organizing and operating without (much) human intervention. Furthermore, WSN protocols should not impose excessive computational burden or require too much memory to save state information.

1.1 Motivation and Goals

In many application areas of multi-hop WSN's such as monitoring applications, building automation, industrial control, and aerospace applications, the network traffic is dominated by the presence of periodic data sources

[154, 159, 54, 32]. For example, temperature sensor nodes can transmit their sampled readings once a minute, light sensors can transmit their reading every few seconds, and so on. The generated data samples are often delivered through a set of forwarder nodes to a dedicated gateway or sink node, which analyzes the periodic data and presents it to a human user or computes responses. In many of these applications it is also required to achieve good timeliness and high reliability of data delivery, i.e. it is necessary that some large fraction of all packets reaches the sink within some specified maximum time. At the same time, there is possibly conflicting requirement of energy efficiency when most or all of the nodes operate on batteries. Therefore, communicating the sensed data timely and reliably while consuming the minimum amount of energy is very essential.

One of the key approaches to achieve energy-saving is to let the sensor nodes switch to an energy-conserving sleep state whenever possible. In this sleep state several parts of the node circuitry, including the wireless transceiver, are switched off, as often the transceiver consumes the most energy on a sensor node [9, 139, 149, 60]. This leads to substantial energy savings but disables the communication ability of a node. The fraction of time where the node is awake is called its *duty cycle*, and from the perspective of energy-efficiency this duty cycle should be kept as small as possible.

For a source node generating the periodic data there is no problem: the node wakes up, samples its sensor(s), transmits a packet and returns to sleep mode. However, in a multi-hop network other nodes are needed to forward the packets to a sink node. To be most energy-efficient, a forwarder should wake up just before a periodic packet arrives, do the necessary forwarding work and enters sleep mode again. However, in general the time difference between packet inter-arrival times (the jitter) as seen by a forwarder node is not ideally regular and so the arrival times are not know precisely. The random components in the inter-arrival time are for example due to usage of randomized Medium Access Control (MAC) protocols, time-varying cross-traffic resulting in queueing effects, retransmissions, blocking of interrupts by node operating systems, etc. Intuitively, one might expect that, the amount of jitter (for example expressed as the deviation from the perfect period) is a function of the number of hops a packet traverses.

One particular way to schedule the wakeup times for forwarders rests on the Time Division Multiple Access (TDMA) MAC protocol scheme, in which time is sub-divided into subsequent superframes, and these are further sub-divided into individual time slots, which then are assigned on an exclusive basis to pairs of nodes. The TDMA approach has, amongst others, been adopted for example in the recently standardized industrial wireless sensor network technologies WirelessHART [66] and ISA-100.11a [8]. With TDMA, each node is allowed to sleep in those time slots in which it neither transmits

nor receives, and these are known to the node through its TDMA schedule.

It is a long-standing debate in the realm of wireless sensor networks how the sleeping opportunities achieved through adopting the TDMA principle on the one hand compare to the complexity and energy cost of TDMA protocols on the other hand. TDMA protocols use an explicit time synchronization protocol in order to be able to switch between different channels and communicate. The existing WirelessHART and ISA-100.11a standards both use a centralized coordinator (an expensive hardware) for calculating TDMA schedules. This coordinator collects load and topology data, computes a TDMA schedule for each node and disseminates these schedules back into the network. This involves extensive signalling overheads, thus increasing energy-consumption [90, 94, 93]. There are further overheads for synchronization purposes. For example, in the WirelessHART standard nodes need to resynchronize every 30s even if there is no need to send data packets in the near future. Moreover, because nodes set-up schedules to communicate between each other in advance, adaptivity of network topology or to the changes of traffic demands is expected to be costly in terms of energy and delay [34, 143]. Does all the TDMA overhead pay out [12, 141, 183], or is it possible to carry out periodic data transmission without all the overheads of TDMA while still maintaining its main benefits of achieving sleep times for nodes and supporting periodic data?

1.2 Autonomous Framework Architecture

A key goal of this thesis is to shed light on this question by comparing the TDMA-based state-of-the-art WirelessHART industrial wireless sensor network (a commercially successful representative of this class of networks) against an alternative design developed in this thesis, called the **autonomous framework**, organized around several components and designed to support high reliability, low delay and low energy consumption for periodic traffic applications.

Instead of relying on pre-computed schedules and deterministic medium access, the key approach in the autonomous framework is to allow forwarders to autonomously *learn* and estimate the periods of all traffic flows going through them and to determine their wakeup and sleep times accordingly. In particular, a forwarder alternates between two different states: In the **learning state** a forwarder is switched on all the time and observes all packets from its neighbors. After a number of observations the forwarder is able to estimate the period and the relevant quantiles. Once these estimates are reliable enough, the forwarder enters the other state, called the **operational state**. In the operational state the forwarder follows the sleep/wakeup cycle, where it wakes up and sleep just at the right time.

Furthermore, the forwarder observes the packet loss rate in the operational state and continues to update the estimates of the period and the quantiles (we refer to this as **statistics update**). If the packet loss rate grows too large, the forwarder returns to the learning state in order to re-estimate period and quantiles. This allows forwarders to adapt to changes in topology or load scenario. This approach does not need centralized scheduling or time synchronization, and consequently does not employ a deterministic MAC protocol, instead, we rely on a CSMA-type MAC.

In designing the autonomous framework it clearly was not sufficient to focus on energy consumption alone, as transmission reliability is very important as well. Reliability in WSNs can be low because of path loss, multi-path fading, or narrow-band interference [172, 188, 175]. Low communication reliability causes packets to be lost, and therefore retransmission of lost packets is usually needed, which in turn leads to increased energy-consumption [138].

A popular approach to improve reliability is to exploit frequency diversity by channel hopping, i.e. periodically changing the communication channel. Channel hopping is known to substantially improve communication reliability in wireless networks [36, 89], and therefore it has been adopted in recent standards for industrial wireless sensor networks, for example Wireless-Highway Addressable Remote Transducer (HART) and International Society of Automation (ISA)-100.11a [99, 66, 83, 28, 58]. Both WirelessHART and ISA100.11a rest on a TDMA approach with slow frequency-hopping, i.e. slot-by-slot frequency hopping.

Figure 1.1 depicts the high level architecture of our autonomous framework which includes: estimation and identification of the flows, asynchronous channel hopping, local dynamic multiple sleep state scheduling, an on-the-fly traffic adaptation mechanism and an overlapping controller. We explain these components in more detail:

Multi-flows estimation and adaptation: Each node autonomously identifies periodic flows passing through it, estimates their periods and adapts its duty cycle accordingly. Each forwarder node acquires knowledge about the traffic characteristics by observing the mean packet arrival time and its jitter over time. These two parameters are estimated with the help of sequence numbers and timestamps and their corresponding values are updated after each packet arrival.

Asynchronous channel hopping: This novel mechanism allows the source nodes and all forwarders to switch channels for each new periodic packet. Source nodes are independent of each other, i.e. they choose their own transmission periods and channels autonomously. A forwarder uses the estimated traffic periods also for figuring out the times when it needs to switch the channel. The main idea of the asynchronous channel hopping

is to use both the flow period information and packet sequence number for selecting the next channel. Specifically, we use a translation function to map the packet sequence number and other parameters to the next channel. We also address the issue of collision due to frequencies overlap.

Local dynamic multiple sleep states scheduling: This mechanism exploits the several different sleep states provided by the chosen radio transceiver (which in this respect is a representative for a large class of transceivers supporting multiple sleep states) and utilizes them in efficient manner. The idea is to let each individual node determine the appropriate sleep mode that would still allow it to be awake when needed dynamically, based on local information of the expected traffic.

On-the-fly traffic adaptation mechanism: Depending on the underlying application, source nodes may increase or decrease their traffic periodicity. In order to enable the forwarders to react and adapt to the new change in an agile manner, we developed very efficient approach which relies only on local information and operates without any centralized components. Thus, it allows the source nodes to increase or decrease their packet generation rate based on their sensing requirements. The key idea is that the source node notifies its neighbor about the new traffic period by just setting up a single bit, abbreviated as LB (Learning Bit). When a forwarder receiving data packet with LB set, it immediately enters the learning phase to start the process of estimating a new traffic characteristics, otherwise it stays in the operational phase.

Multi-flows overlapping controller: Forwarders might be placed on the routes for several distinct sources and must adapt both the sleep/wakeup windows and also the frequency, especially in situations where packets of different source flows are expected to arrive at about the same time at a forwarder. The basic idea to eliminate such a collision due to the traffic overlap is to estimate the traffic characteristics for each flow separately; each flow is distinguished from the others by using a unique flow ID.

To detect whether there is a potential overlap in the next cycle, the node compares the next expected time intervals among the different flows. Upon a detection of a potential overlap the node tries to resolve it by piggybacking some information to the conflicting nodes beforehand.

We will compare the autonomous framework and WirelessHART for their energy consumption, packet delay and their achievable reliability in a range of scenarios with periodic traffic, and taking various real-world overheads into consideration. Our results indicate that for scenarios with low to modest overall traffic loads the autonomous framework requires substantially less energy

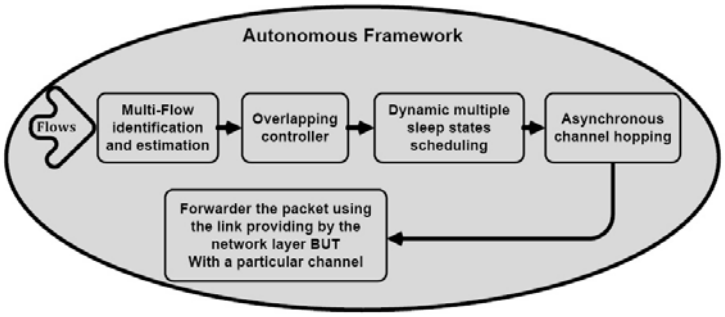


Figure 1.1: Autonomous components

and lower delay than WirelessHART while achieving similar reliability. Moreover the autonomous framework outperforms the centralized approach in terms of adaptability to varying traffic conditions.

1.3 Contributions of the Thesis

In order to achieve sufficient communication reliability, low delay and good energy-efficiency in resource constrained WSN devices in networks with periodic traffic, we have, as our first contribution, designed a decentralized, self-learning and self-adaptive approach called the **autonomous framework**. Our autonomous framework solution integrates several mechanisms, including: asynchronous channel hopping, multi-flow traffic estimation and adaptation, local dynamic multiple sleep states scheduling, a control mechanism for overlapping flows, and a mechanism to adapt to traffic changes. A more detailed presentation of design objectives and associated research challenges can be found in Chapter 3, the detailed design of the framework is presented in Chapters 4 and 5.

The second main contribution is a detailed analysis of the energy consumption characteristics of WirelessHART. More specifically, we have conducted a study using the response surface methodology to determine how the energy consumption depends on various system parameters like transceiver power parameters, synchronization overhead etc. Through this analysis we have identified the main factors contributing to the energy consumption of WirelessHART. This knowledge is not only of intrinsic interest, it is also useful to guide energy optimization of WirelessHART. By identifying the factors contributing most to the overall energy consumption we can focus our efforts to save energy to the most promising component. Based on the insights from this analysis we also propose and analyze an energy management policy for WirelessHART nodes, which allows them to autonomously exploit the multiple sleep states of a representative radio transceiver (the popular IEEE 802.15.4-

compliant ChipCon CC2420 [30]) and utilizes them in efficient manner. To carry out this analysis, we have developed a realistic simulation model for WirelessHART. We validate the WirelessHART simulation model using a real WirelessHART evaluation Kit.

The third main contribution is a similar analysis of the autonomous framework, again identifying the factors contributing most to its energy expenditure. Similar to the WirelessHART analysis, we use a simulation model to analyze the autonomous framework. It is important to note, however, that some of the underlying assumptions and approaches for this framework have been validated by experiments in a a real sensor network devices.

As our fourth main contribution we use the simulation models for both systems together with real-world measurements and real-world connectivity traces to conduct a detailed comparison study in terms of energy consumption, packet delay, adaptability to varying traffic loads and achieved reliability for different deployment and load scenarios.

1.4 Thesis Outline

The remainder of this thesis has the following structure:

Chapter 2: details the background and related literature for energy-efficiency techniques for WSNs. It starts by identifying the fundamental components that impact on the overall energy consumption, then it details and discusses the low-energy medium access control protocols identified in literature for both single and multi-channel solutions.

This chapter also provides an overview of the state-of-the-art WirelessHART technology, explains the WirelessHART components and presents the relevant details of its operation. This includes the channel-hopping and time synchronization mechanisms adopted in the WirelessHART protocol.

Chapter 3: firstly presents a detailed problem statement, describes the main research challenges and sketches the solution approach, together with the system metrics, assumption and considered scenarios. Furthermore, it contains an overview on the performance evaluation methodology used on this thesis.

Chapter 4: presents the basic design of the autonomous framework and evaluates its performance in a single-flow scenario on a single channel. We also present measurement results for the per-hop jitter, which help to motivate key decisions in the approach taken for estimating traffic periods.

Chapter 5: presents an extension of the autonomous framework to multi-flow and multi-channel scenarios. In the first part of the chapter we present an overview of the general autonomous framework, including the scheduling of wake-up times. Important parts of the design will be given as state diagrams. Then we present the design of the autonomous framework and its state diagrams which includes an asynchronous channel hopping scheme, estimation and adaptation algorithms, sleep/wakeup scheduling, adaptive traffic policy and the multi-flows overlapping mechanism. In the second part we conducted a trace based simulation to evaluate the autonomous framework. We also conducted a sensitivity analysis using the response surface methodology to analysis the impact of the autonomous framework parameters on the energy consumption.

Chapter 6: presents a comprehensive performance evaluation of the WirelessHART protocol, including a sensitivity analysis of the WirelessHART energy consumption parameters using the response surface methodology. We also propose and analyze an energy management scheme for TDMA systems in which more advanced sleeping capabilities of the chosen radio transceiver are utilized. In this scheme each node individually selects its next sleep state according to its transmission/reception schedule. With this scheme the energy consumption in the sleep state can be reduced substantially. In this chapter, we also validate the simulation models using real-word experiments.

Chapter 7: in this chapter we compare WirelessHART and the autonomous framework in a range of scenarios for their energy consumption, packet delay, adaptability to changing traffic rates and achieved reliability. This allows us to identify the advantages and disadvantages of each approach.

Chapter 8: this chapter summarizes the contribution of the thesis and discusses the lessons learned. We also outline several future directions and open issues for both our autonomous framework and the WirelessHART system.

Details about the simulation models for WirelessHART and the autonomous framework are given in Appendix A and Appendix B, respectively.

1.5 Publications

Journal Articles:

- **Osama Khader**, Andreas Willig and Adam Wolisz, "Self-Learning and Self-Adaptive Framework for Supporting High Reliability and Low En-

ergy Expenditure in WSNs" Special issue of Telecommunication Systems Journal, vol.x , no.x, To appear 2014.

- **Osama Khader** and Andreas Willig, "An energy consumption analysis of the Wireless HART TDMA protocol". Journal of Computer communications, vol. 36, no. 7, april 2013.

Conference Proceedings:

- **Osama Khader**, Andreas Willig and Adam Wolisz, "An Autonomous Framework for Supporting Energy Efficiency and Communication Reliability in WSNs". In Proc. of the 6th Joint IFIP Wireless and Mobile Networking Conference (WMNC2013), apr 2013 **(Best Paper Award)**.

- **Osama Khader**, Andreas Willig and Adam Wolisz, "WirelessHART TDMA Protocol Performance Evaluation Using Response Surface Methodology". In Proc. of the 6th International Conference on Broadband and Wireless Computing, Communication and Applications (6th BWCCA 2011), oct 2011.

- **Osama Khader**, Andreas Willig and Adam Wolisz, "Distributed Wakeup Scheduling Scheme for Supporting Periodic Traffic in WSNs". In Proc. of the European Wireless Conference 2009 (EW '09), ISBN: 978-3-8007-3167-1, pp. 287-292 Aalborg, Denmark, may 2009.

Technical Reports:

- **Osama Khader**, Andreas Willig and Adam Wolisz, "Self-learning and adaptive scheme for Supporting periodic Multi-flows in Wireless Sensor Networks", TKN Technical Report Series TKN-13-002, Telecommunication Networks Group, Technical University Berlin, mar 2013.

- **Osama Khader**, Andreas Willig and Adam Wolisz, "A Simulation Model for the Performance Evaluation of WirelessHART TDMA Protocol", TKN Technical Report Series TKN-11-001, Telecommunication Networks.

- **Osama Khader**, Andreas Willig and Adam Wolisz "Dynamic Adaptive Wake-up Scheduling Scheme for Supporting Periodic Traffic in WSNs", TKN Technical Report Series TKN-08-012, Telecommunication Networks Group, Technical University Berlin, nov 2008.

CHAPTER 2

Background and Related Literature

The constraints found in wireless sensor networks, like the restricted computational power and memory, the energy limitations and the reliance on wireless communications, often call for application-specific network and protocol designs. In this way the network architectures and operation of protocols can be tailored to the traffic types and load generated by a specific application.

This chapter provides a discussion and overview of the related work for low power approaches for both single and multi-channel solutions. In this chapter, we also provide an overview of the state-of-the-art solution (WirelessHART), which will be used as a benchmark against our autonomous framework solution.

2.1 Energy Consumption in Wireless Sensor Networks

Wireless sensor networks are typically battery powered devices, therefore minimizing the energy usage is one of the main issues in WSN [6, 5]. In reality, the network lifetime depends on energy consumption at each of the sensor nodes. The four main factors influencing the energy consumption of an individual sensor node are:

1. Radio transceiver.
2. Microprocessor.
3. Sensors and actuators.
4. Memories and other electronic circuits.

Therefore, to extend lifetime of the WSN, efficient power management protocols must be considered in hardware, software and algorithm design. As the transceiver consumes maximum amount of energy, including that in idle, transmit and receive mode [152, 145, 35], therefore it is useful to enable the radio to operate in low duty cycle. Low duty cycle is defined as the fraction of time where the node is awake in the whole operation.

There are four important factors that contribute to the energy consumption in WSN communication:

Idle listening: Occurs when the radio is on and listening on the channel but no packets are received. It has been observed that the idle listening consumes significant energy in the wireless sensor networks [182]. It might consume up to 50% of the energy required for receiving [182, 153].

Collision: It is the second source of energy consumption and it happens when collision occurs (two packets are transmitted at the same time and they interfere with each other) in this case, node usually retransmits the packet, that contributes to the energy wastage as well [74].

Control packet overhead: Signaling information helps to maintains the overall network operations. For example, signaling Acknowledgement (ACK)s, RTS/CTS packets. Moreover control packets used for time synchronization and maintaining the health of the network consume a significant amount of energy [143, 34, 87].

Overhearing: As the wireless channel is a shared medium, nodes might listen and receive neighbors communications although these communications may not be designated to these particular nodes. Consequently, the nodes should be switched off during the idle periods to save energy [154].

As a result of these factors, most of the energy management solutions are typically addressed in MAC layer ([182, 150, 164, 46, 135, 11]), few of them in network layer [38, 47, 103].

The MAC layer has two important roles:

Firstly, it controls the Radio transceiver states (listening, receiving, and sleeping), hence it allows energy saving of the nodes. Secondly, it is in charge of regulating channel access to a shared wireless medium as wireless media is broadcast in nature. Therefore, MAC protocols are responsible for resolving potential contention so that no nodes interfere with each others transmission. Moreover,

In the next section, we discuss several low power MAC protocols which address the four factors viz. idle listening, collision, control packet overhead and overhearing discussed above.

2.2 Low-Power Media Access Control Approaches

In this section, we present the related work in the area of energy-efficient MAC protocols for wireless sensor network. The primary goal of low-power MAC protocols is to put the radio into sleep mode as much as possible for energy conservation. Several wireless sensor network MAC approaches have been considered, which can be broadly classified into three main categories: random-based protocols, framed-based protocols and schedule-based protocols [104].

2.2.1 Random-Based Protocols

In the random-based MAC approaches, nodes may start a transmission at any random time and must contend for the channel. Basically, nodes define shared active and sleep periods. In the active periods nodes contend for the channel to communicate using carrier sense multiple access with collision avoidance protocol and in the sleep periods nodes can save energy by switching off the radio. An example of random based protocol is the popular *B-MAC* protocol [136]. In B-MAC (see Figure 2.1), each node periodically wakes up and briefly checks for the activity on the channel. If the channel is idle, the receiver goes back to sleep. Otherwise, the receiver stays on and continues to listen until the packet is received.

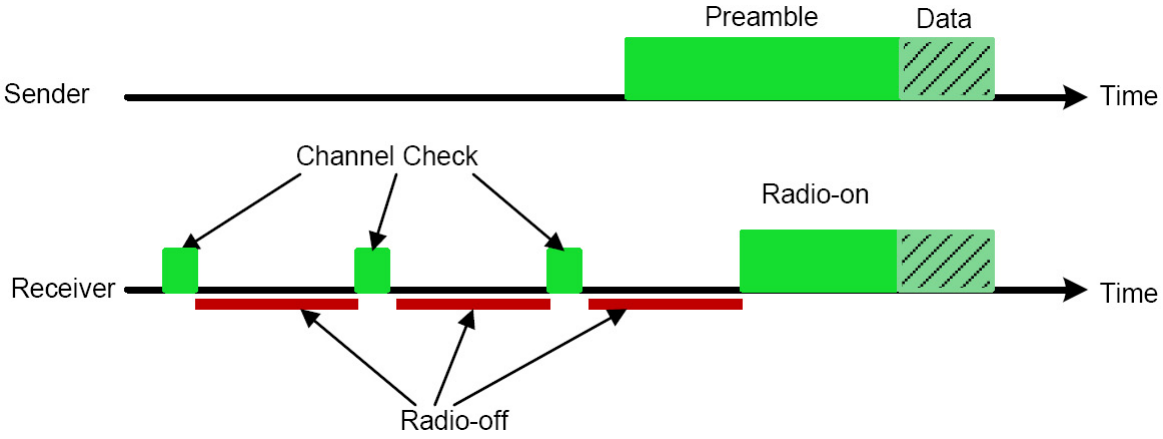


Figure 2.1: B-MAC communication example [184].

To rendezvous with receivers, senders preceded the data packets with long preambles to catch the channel check period. The period of the preamble must be longer than the sleeping period of the receiver, which ensures intersection with the polling period. This technique is invoked Low-Power Listening (LPL).

One of the advantages of B-MAC is that it reduces the idle listening because the radio is switched off for the duration of transmission of other node. However, in B-MAC the sender node drains significant amount of energy due to long preambles and occupation of the transmission channel for longer periods of time.

WiseMAC [43, 77], is an another example of LPL protocols. It improves the B-MAC by reducing the size of the preamble for unicast transmissions. It assumes that some information about the receiver is given, otherwise long preamble as B-MAC should be used. The receiver piggyback its next checking time in the acknowledgment packet, allowing the sender node to send the next packet with a short preamble. Each node maintains a timing information table about its neighbors which is updated after each successful transmission. One of limitations of this protocol are: first, when a node sends broadcast packets, it has to use long preamble, even if it knows the checking interval times of all its neighbors. The reason is that nodes asynchronously poll the channel, and only long preambles ensure that all nodes can capture the packet. Moreover, due to clock drift, a node can only send the second packet with a short preamble if it closely follows the first one, otherwise it uses long preamble for low traffic rate.

Authors in [11] and [148] introduce another protocol to reduce the preamble length. In this protocol, a long preamble packet is replaced by a train of strobe packets called micro-frames. Each strobe packet contains some information about the data packet. This includes information related to the destination address, source address and a digest field to indicate the number of strobe packets to be transmitted before the data packet. This protocol allows the receiver node(s) to enter sleep mode once a strobe packet is received. It then extracts and learns about the next data packet transmission from the sequence number provided by the strobe packets about the next data packet transmission, hence avoiding the node to listen to subsequent strobe packets.

X-MAC [20] uses the same idea to enhance B-MAC by providing shorter preambles. Instead of sending one long preamble, a node broadcasts a train of short strobe packets and listens between each strobe packet. Each strobe packet includes the target node address. After receiving a strobe packet, a node checks the address information of the strobe. If it is the node's address, it sends a short acknowledgment packet (called early ACK) and prepares to receive a full data packet. Otherwise the node goes back to sleep. Once the sender receives this early acknowledgment, it then transmits the packet immediately (see Figure 2.2).

The train strobe protocols such as X-MAC achieve good energy savings compared to the B-MAC under very low traffic rates [105], but their strobe packets still occupying the wireless medium till the packet is eventually transmitted. However, it is not an attractive solution in case of multi-flow traffic

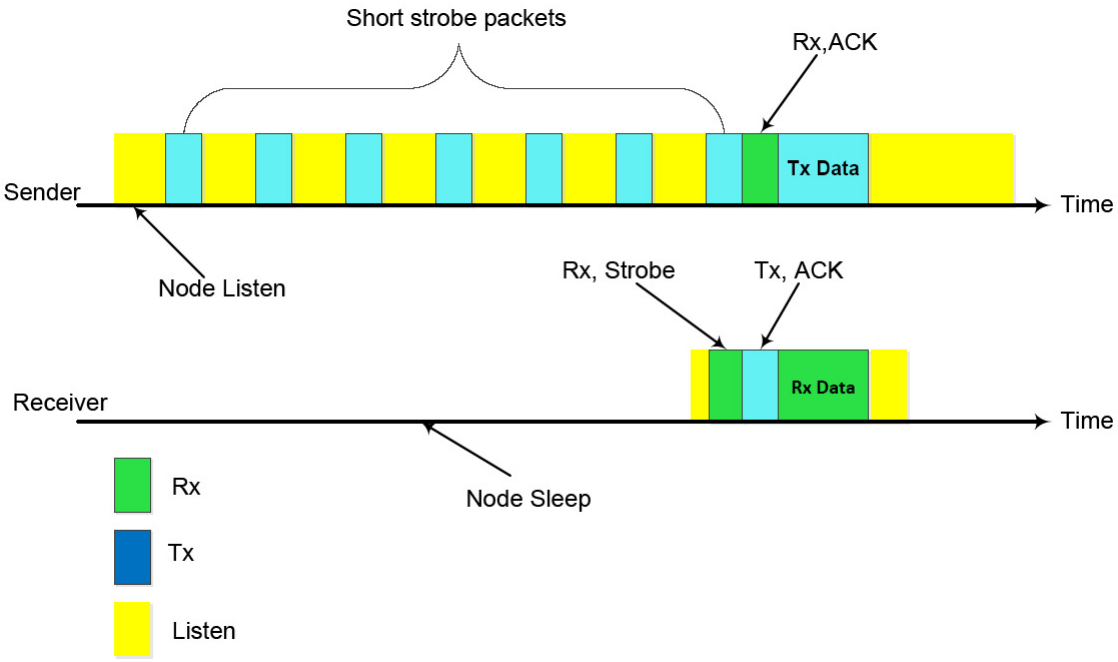


Figure 2.2: X-MAC communication example [20].

scenarios, due to the long contention periods. Also, the interval between the strobe packets can lead to collision when the new node starts to send new train of strobe packets which may overlap with other strobe packet periods.

There are also other schemes in the literature that refer to similar approach of the preamble sampling such as Receiver Initiated Cycled Receiver (RICER) [111, 110].

In the RICER the communication is shifted from the sender side to the receiver side. When the receiver node wants to receive a packet it transmits a wakeup beacon packet to announce that it is awaken. The receiver immediately switches to listening mode and waiting for a packet from the sender (for a predefined time). If the receiver gets a packet, it forwards the packet immediately otherwise, it enters the sleep mode. On the other hand, if the sender wants to transmit a packet it stays awake and monitoring the channel waiting for a beacon packet from the receiver. Upon the reception of the beacon packet, the sender sends its data packet and waits for an acknowledgment. The idea of the RICER is more or less similar to the preamble sampling. However instead of transmitting long preamble packet, the sender keeps receiving. This is more energy efficient compared to transmitting long preamble. The receive transmits periodic wakeup beacon to announce its ready to receive packet. The receiver also keeps listening for some time after each wakeup beacon transmis-

sion. Therefore, the overhead expected to be large especially in lightly loaded networks as the receive has to do that periodically.

Generally in the pure LPL MAC protocols [136], there is no need to coordinate the cycles, and therefore there is no need of clock synchronization. The major drawbacks of the LPL are: first long preambles significantly increases the burden on transmission. Second, while LPL can be optimized for known static periodic traffic, its performance may significantly degrade at varying periodic traffic load.

2.2.2 Framed-Based Protocols

In framed-based protocols such as *S-MAC* [182, 183] and *T-MAC* [164], nodes are required to be synchronized and time is organized into equal frame size. As shown in Figure 2.3, each frame is divided into two time intervals. The first time interval is meant for nodes to exchange synchronization information. In the second interval the nodes may receive or send based on the Request To Send (RTS)/Clear To Send (CTS), otherwise sleep mode is entered. This approach is borrowed from the 802.11 Power Saving Mode (PS) [176, 129].

Figure 2.3: Example of framed-based structure

S-MAC uses a mechanism called virtual cluster to enable nodes to synchronize on a common frame structure. Specifically, nodes transmit broadcast synchronization packets at the beginning of each frame periodically. The node waits a random time before it transmits the synchronization packet and if it hears another synchronization packet then it cancels its transmission. It then adjusts its clock to the received synchronization packet, after exchanging the synchronization packet within the first interval of the frame. To account for joining new nodes, a node must listen periodically (about 10 second every 2 minutes). In the second interval of the frame a node may agree on the data packet by exchanging request-to-send and clear-to-send packets. The frame size of the S-MAC is in the order of 500ms to 1 second and depends on the application requirements. In other words, the application should explicitly specify a fixed duty cycle before starting the S-MAC. This is the main disadvantages of common active interval approach. In order to compute an appropriate active interval, one has to understand the relationship between idle listening and collisions. On one hand, short active interval may decrease the idle listening however, it increases the probability of collision due to the

increase of contention. On the other hand, long active interval may reduce the contention and thus, the collision rates but at the cost of energy (increase idle listening). Another potential drawback is the scalability. If the size of network increases, S-MAC must maintain an increasing number of neighbors' schedulers or incur additional overhead by repeated rounds of resynchronization.

T-MAC [164], is an extension version of S-MAC. It includes an adaptive duty cycle. In T-MAC, nodes are required to be synchronized and time is organized into equal and fixed frame size (615ms). Each frame is divided into two time intervals. In the first time interval nodes can exchange synchronization information. In the second interval a node may receive or send based on the RTS /CTS. T-MAC uses time-out mechanism to dynamically determine the end of the active period. To span a small contention period and an RTS/CTS exchange the value of time-out set to 15ms. If the node does not detect any activities in the channel within the time-out value (about 15ms), then the node goes to sleep, otherwise it starts a new time-out window. T-MAC reduces the energy consumption at the cost of reduce throughput and additional latency. T-MAC also experiences similar issues as those discussed earlier in S-MAC protocol

2.2.3 Schedule-Based Protocols

Schedule-based access or time division multiple access (TDMA) protocols allocate an exclusive time slot for data transmissions between node pairs. In these protocols both slot assignment algorithms and tight clock synchronization algorithms are important.

For example, *LMAC* [165] uses a simple random slot assignment algorithm that ensures that nodes at two-hop distance do not use the same slot number. It assumes a global time synchronization. Synchronization is performed with every header that is sent. The drawback of LMAC's is that nodes must always listen to the control sections of all slots in a frame, including the unused ones.

Similar to the LMAC, TRaffic-Adaptive Medium Access protocol (TRAMA) [140] uses distributed election scheme to assign time slots to each node on demand. Thus it takes the traffic load for each node into account when assigning the schedule table among neighbors. It assumes that all nodes are time synchronize and organize the time into two different periods; random access period and scheduled access period. The random access period is divided into equal slots called signaling slots, and the scheduled access period is divided into equal slots called transmission slots. These two periods are repeated over time. The signaling slots within the random access period are used to form two-hop topology information (using CSMA protocol). During the scheduled access period the transmission slots are used to send data packet directly without waiting for channel access, this reduces the idle listening and overhearing.

The main issues for this protocol are: fist, complexity and memory requirement for maintaining scheduling information. Second, the assumption that nodes are synchronized.

All these solutions including our previous publications [92] and [91] which introduced a novel scheme for extending the sleep times of wireless sensor nodes on-line and in decentralized way (discussed in Chapter 4) are restricted to work in a single channel solution. In the next section we review the multi-channel media access control protocols for WSNs.

2.3 Multi-Channel Media Access Control Protocols

In this section we present the state-of-the-art for channel hopping MAC protocols for WSNs. Multiple-channel implies the ability to change the frequency channel of a node on a regular basis under a single radio interface. This becomes more practical and useful as the current WSN's hardware provides the basic functionality required in a very efficient way. For example MICAz [75], Telos [137], Imote2 [3] and SHIMMER [22] which uses CC2420 radio chip [30], support channel hopping capability. Moreover, the switching time for all the IEEE 802.15.4-compliant [130] chips are less than $192\mu s$, thus making this approach efficient. We do not study multi-channel MAC protocols that are based on multi-Radio interface and therefore, we don't considered them in this thesis as they are not-economical and not-practical solutions for WSNs so far (at the time of this writing). An interested reader is pointed to [4, 126, 179, 125, 23] research works in which the hardware is assumed to handle different transmission and able to listen to multiple frequencies at the same time.

2.3.1 Challenges of Multi-Channel Solutions

In this section we discuss the challenges of the multi-channel communication under single radio transceiver for WSNs. Unlike the traditional single channel solutions for transmitting and receiving packets, multi-channel solutions exploit the available channels and switch between them to improve the communication reliability [127, 178, 13]. However, multi-channel communication needs to address the following major challenges:

Precise Common clock reference: In order for a pair of nodes to hop to the same frequency channel to communicate, they need to have a common clock reference. This can be obtained using a time synchronization protocol however, since hardware clocks are generally imprecise, time synchronization is crucial, especially in multi-hop systems [44, 45]. There

are two main factors that influence the local time accuracy. The first is clock drift, which indicates the rate at which a clocks actual frequency deviates from its nominal frequency; and the second is clock offset, which is the difference from ideal time. A popular approach to maintain a common time reference across a set of nodes is to periodically exchange synchronization packets, however an interesting question to answer is how often to transmit such packets?. Determining the optimal periodicity of the synchronization packet is also challenging and requires taking into account the trade-off between clock accuracy and energy cost. High frequent packet exchange result in more accurate synchronization, but also in more network traffic and therefore more energy consumption. Low frequent exchange does the opposite, it reduces the energy consumption at the cost of clock accuracy.

Multi-channel quality monitoring: Another key challenge to address in multi-channel solutions is the scanning and monitoring of channel quality. As discussed in Section 1.1, channel hopping mitigates the multi-path prorogation and narrow-band interference caused by other closed radio sources such as 802.11 which share the same radio frequency band as 802.15.4 (particularity when 802.14.5 node and 802.11 client transmit simultaneously on any overlapping frequency). Also, Bluetooth and cordless devices operate in the same frequency band as 802.15.4 thus causing interference [146]. Consequently, in order to enhance the communication reliability, periodic scanning and ranking of the available channels is required. This is not only costly in terms of energy and time but also memory, as each node should maintain statistics for each channel and each link it communicates in.

Multi-channel discovery: Another important challenge in multi-channel protocols is network discovery in which a newly joint node has no prior knowledge about the network. Particularly, when and in which channel to operate, as the sender must know the channel in which the receiver is listening to, so it selects the particular channel and start its activities. As a result, there is a trade-off between energy efficiency and speed of discovery. The required time and energy for a node to listen and switch between different channels in order to join the network is considerably high compared to the single channel solution. This is even become more crucial when some of the nodes are not static (mobile) [40, 88].

2.3.2 State of the Art in Multi-Channel Protocols

Generally, in order for a pair of nodes to switch to a specific channel to communicate, they need to be tightly synchronized [170]. Thus, the current proposed

multi-channel MAC protocols are limited to a time division multiple access (TDMA) and should maintain a precise time information among their neighbors.

Multi frequency media access control (MMAC) [151] uses the same technique proposed in IEEE 802.11 PSM [163]. Particularly, the time is divided into beacon intervals which are repeated over time. Each beacon is subdivided into two parts: channel negotiations part and data exchange part. In the channel negotiation part the MMAC uses the same notion of the Ad hoc Traffic Indication Message (ATIM) window of the IEEE 802.11. During the ATIM window all nodes switch and listen to the default channel. If a node has a packet to transmit then it has to negotiate with its next hop neighbor about the channel to use during the ATIM window.

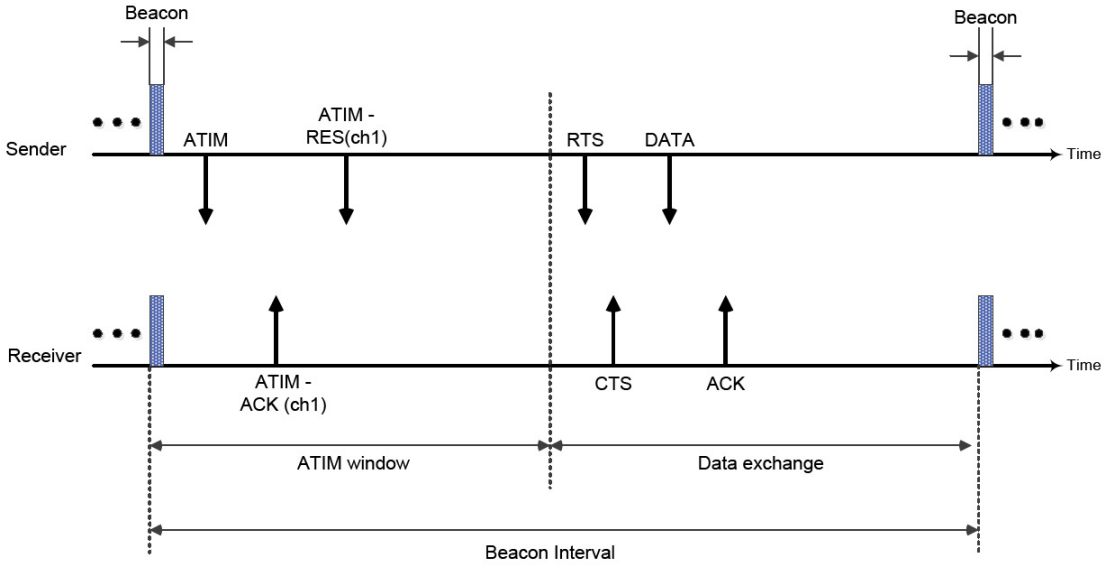


Figure 2.4: Example of MMAC structure [151].

As shown in Figure 2.4, the sender transmits an ATIM packet to the receiver, the receiver then selects a channel from its list and sends this information using an ATIM-ACK packet to the sender. Upon receiving the ATIM-ACK packet, the sender transmits back an ATIM-RES packet to confirm that the specified channel in the ATIM-ACK is selected. If the sender can not select the specified channel in the ATIM-ACK packet then it does not respond with an ATIM-RES packet and waits for the next negotiation phase with different channel suggestion. The sender transmits its data packets in the beginning of the data exchange part. The main disadvantage of the MMAC is the long fixed negotiation phase. Precisely, nodes are not allowed to transmit a data packet during the ATIM window even if they have already selected the channel but

they have to wait till the start of the data exchange phase.

Similar to TDMA protocols in a single channel solutions, Y-MAC [99] is a TDMA based multi-channel MAC protocol. It divides time into superframes and slots. Nodes are synchronized and slots are assigned to each sender and receiver nodes. The resynchronization period is 8 second. Y-MAC was evaluated against LPL which uses single channel solution. In its evaluation 5 channels were used and the periodic traffic was set to 10 second. The main drawbacks of the Y-MAC protocol are that it has the same flexibility and scalability issues as that like TDMA system (i.e., fixed scheduling slots assignment), as it assumes a static traffic scenarios.

Multi frequency media access control for wireless sensor networks (MMSN) [189] uses synchronization protocol to assign time slots and channels at the starting phase of the networks. Each node is scheduled to use two different frequencies, for transmitting (must be the same as the destination frequency) and receiving. Each time slot has two periods: broadcast contention part and transmission period. In the contention period node competes for the same default channel for broadcast packets and can transmit its own uni-cast packet in the assigned channel. Since node receives and transmits on two different channels in the same time slot, MMSN requires a very fast switching time between transmission and reception channels.

Multi-Channel-MAC [81] is an improved version of the single channel L-MAC [166]. It uses a TDMA protocol and the whole network is synchronized. The slot assignment algorithm is run in a distributed manner, thus not relying in a centralized manager. Slots are selected randomly, therefore it is possible for multiple nodes to select the same slot which would be result in a collision. This protocol is design specifically for small network size and does not perform well in large network. Also because the slot assignment is computed only once, it does not adapt to traffic change and topology.

Another protocol that uses TDMA-based medium access with multi-channel mechanism, is the IEEE 802.15.4E TSCH (Time Synchronized Channel Hoping) protocol which focuses on enhancing the well know IEEE802.15.4 protocol. It uses the same physical layer of the IEEE802.15.4 but with modified MAC layer. The IEEE802.15.4 and the TSCH protocols will be explained in Section 2.5 and Section 2.6, respectively.

A state-of-the-art solution for a TDMA-based system is the WirelessHART standard [27]. WirelessHART combines frequency hopping with a TDMA scheme utilizing a centralized a-priori slot allocation mechanism. In the next sections we discuss WirelessHART standard which will be used later as a benchmark against our autonomous solutions.

2.4 Overview of WirelessHART

WirelessHART [63, 66, 64, 71] (abbreviated as WHART in the following) is one of the first wireless communication standards specifically designed for process automation applications. The standard has been finalized in 2007, and at the beginning of 2010 it has been adopted as an International Electrotechnical Commission (IEC) standard.

On the physical layer, WHART adopts radios that are compliant to the IEEE 802.15.4 standard [79]. On top of the physical layer, WHART employs a TDMA-based MAC protocol and additionally performs slow frequency hopping (hopping on a per-packet basis). The frequency hopping-pattern is determined from a well-known pseudo-random sequence. The TDMA slot allocation is centrally controlled and slots are assigned at network configuration time.

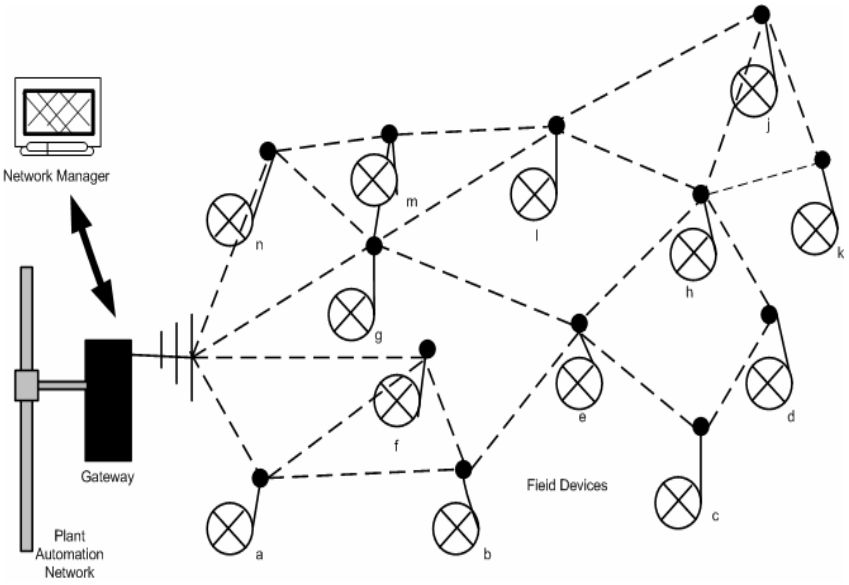


Figure 2.5: WirelessHART basic network components [65].

2.4.1 The WirelessHART TDMA Scheme

In this section, we briefly discuss the WHART TDMA scheme. The timing hierarchy of WHART has three levels. On the lowest level we have individual **time slots**. Within one time slot one data packet and the accompanying immediate acknowledgement packet are exchanged. A time slot in WHART has a fixed length of 10 ms. A contiguous group of time slots of fixed length form a **superframe**. On top of that, a contiguous group of superframes form

a **network cycle**. Within each cycle each field device receives at least one time slot for data transmission. However, certain devices may have more time slots than other devices because they have more data to report or they have additional forwarding duties. An individual field device receives a schedule from the network manager informing him about those time slots where it transmits and those slots where it receives. Furthermore, a field device must maintain time synchronization to agree on slot boundaries with neighbored devices.

2.4.2 Network Architecture

The network architecture of a WHART network features three different types of components (compare Figure 2.5).

Field Device

The WHART **field devices** are used for collecting measurement data from the field and for forwarding this data to a gateway node. They typically integrate wireless communication, sensing, and computational facilities. A field device might be either a genuine WHART device or it might be a legacy (wired) HART device equipped with a HART-specific wired-to-wireless adapter.

Gateway

A **gateway** forms the boundary between a WHART segment and other (often wired) parts of an automation network and is not energy-constrained. The gateway is the point where all sensor data provided by WHART field devices is collected and prepared for further processing. It enables communication between host applications and field devices. There is only one gateway per network and all the WHART devices are known to the gateway.

Network Manager

The **network manager** is a centralized unit which is integrated together with the gateway. It has global information about the network topology, link qualities and the traffic flows. Based on that, it computes a routing and a TDMA schedule and disseminates it to the remaining participants. Slots in the TDMA schedule are allocated hop by hop based, all other stations are allowed to sleep during a slot.

The network manager not only allocates slots, but places those slots also on different frequencies. The network manager has further responsibilities, including the monitoring and health reporting of the WHART network and adapting

the network to ongoing changes. The network manager is not assumed to be energy-constrained.

2.4.3 TDMA Slot Allocation Algorithm

One of the main tasks in designing a TDMA protocol is the allocation of time slots to sender-receiver pairs. Determining a throughput-optimal schedule for TDMA slot assignment in multi-hop networks is an NP-complete problem even in linear networks [2]. The WHART standard defines just the constraints for slot assignment algorithm leaving the selection of slot allocation method open. The defined constraints are:

1. Management slots have priority over data slots.
2. Each device gets three slots every 15 minutes for health reports.
3. Each device gets at least one slot every minute for management frames (advertisement, join request/response, command request/response).
4. Each device gets a slot for keep-alive frames every 30 seconds.
5. Slots for stations having the fastest transmission periods must be allocated first.
6. At least one backup slot should be allocated to each data slot to handle a retry.

It can be clearly seen that WHART standard uses separate slots to be scheduled for the management frames. However, piggybacking periodic keep alive messages with periodic data frames would be more energy efficient.

2.4.4 Time Synchronization

WHART defines two types of time slots: namely dedicated time slots and shared time slots. Here we consider only dedicated time slots, which are allocated to one specific sender-receiver pair. The internal structure of a dedicated slot is illustrated in Figure 2.6.

We first discuss the operation of a receiver node. It enters the receive state at the beginning of its time slot. The receiver measures the exact time when it has detected the start of the frame sent by the transmitter. The nominal time for this to happen is `TxOffset` seconds after the start of the slot. Since the time slot duration is fixed to 10 ms, a node can compute the start of the next time slot from the frame arrival time according to:

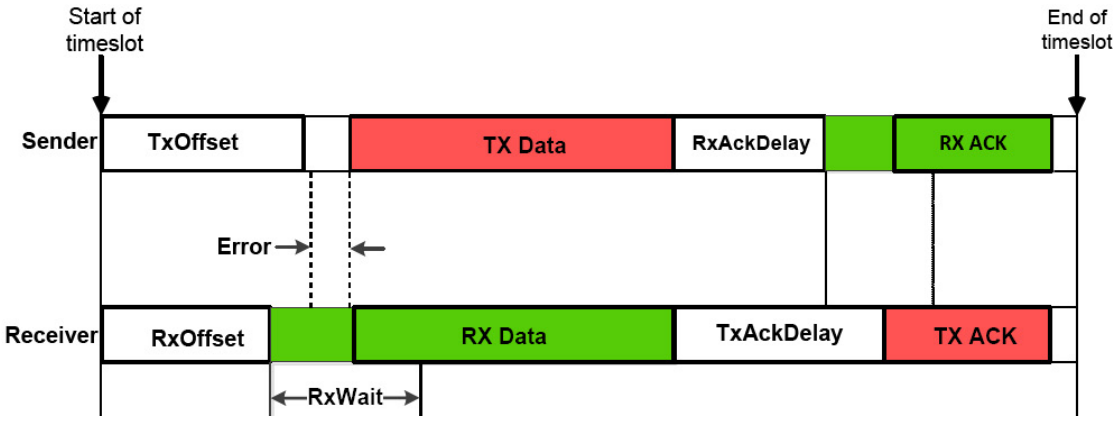


Figure 2.6: Dedicated slot timing [67].

$$T_{NextSlot} = FrameArrivalTime + 10ms - TxOffset \tag{2.1}$$

The receiver computes the timing offset between the actual time when it detected the start of the frame and the nominal time. When the receiver has successfully received the frame, it turns its transceiver into transmit mode, generates and sends an acknowledgement frame, and returns to receive mode. This acknowledgement frame includes the measured offset and allows the transmitter to adjust its local time accordingly.

The transmitter also starts a time slot when its transceiver being in receive mode. After time `TxOffset` it turns its transceiver into transmit mode, sends the data frame, switches back into receive mode and waits for the acknowledgement. The measured offset contained in the acknowledgement is used by the transmitter to re-calculate its local view on the start times of time slots and to adjust its local clock. This approach to clock readjustment is the elementary building block for WHARTs overall time synchronization scheme.

For proper operation of the TDMA algorithm neighboring nodes need to agree on boundaries of time slots and therefore a clock synchronization algorithm is needed. WHART uses the above described device-to-device adjustment method for this purpose, also known as Time Synchronized Mesh Protocol (TSMP) [134]. To achieve network-wide synchronization, a synchronization tree is built with the gateway as its root. Essentially, devices at depth d in the tree synchronize their clocks to devices at depth $d-1$. The gateway has depth 0, its immediate neighbors have depth 1 and so on.

2.4.5 WirelessHART Components

The state diagram for the WirelessHART TDMA protocol is shown in Figure 2.7. The TDMA machine specifies the overall operation, mainly link schedules, and maintaining the time synchronization.

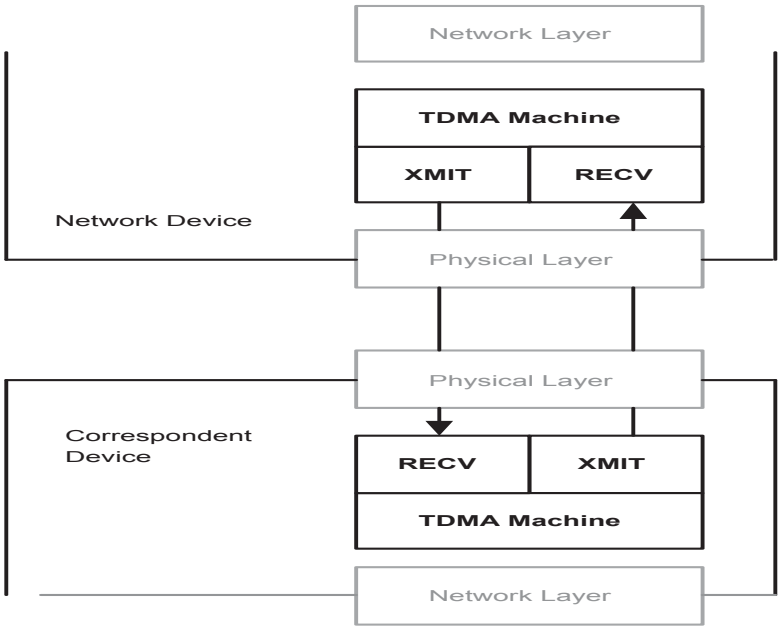


Figure 2.7: WirelessHART TDMA MAC components [67].

The transmitter engine (XMIT) is responsible for sending packets to the corresponding node. The receiver engine (RECV) performs listening and receives an incoming packet. These two aspects are discussed in the following sections.

XMIT engine

When a transmit slot starts, the device enters *Talk* state by invoking the *XMIT* engine and transmits the packet. If the packet transmission fails then the device tries to retransmit in the pre-assigned retransmission slots for the device.

If the transmission is successful and if its not a broadcast packet, then Time Division Multiple Access (TDMA) machine enters the *wait-for-ACK* state by initializing the *RxDelay* timer to *RxAckDelay* and the receive window (RxWindow) to *AckWait* and invoking the *RECV* engine. When a packet is to be transmitted, the *XMIT* engine is invoked to perform the actual packet transmission.

RECV engine

The receive path is managed by the *RECV* engine. This engine is invoked to acquire a data or ACK packet that is being sent by one of the neighbors. When the *RECV* engine is invoked, the transceiver is configured by selecting the correct channel. In addition, the *RxDelay* timer is started and the *Wait-For-Rx-Start* state is entered. During the *RxDelay* the transceiver is allowed to settle and synchronize to the correct channel. The *Wait-for-Rx-Start* state is left when the *RxDelay* timer expires.

The *RxDelay* timer is set by the TDMA machine to allow the receiver to become active at the beginning of the receive window. The duration of the receive window is controlled by the *RxWait* timer which is started after the *RxDelay* timer expires. The node remains in the *Listen for Packet* state until either 1) the start of a packet is detected or 2) the Rx timer expires and an *RxTimeout* occurs. If there is no addressing error, the received packet is validated, otherwise the *RECV* engine terminated indicating "No response".

2.4.6 WirelessHART state machine

In Figure 2.8 we show a schematic of the state machine that governs the operation of field devices in the TDMA scheme. The state machine starts when a device joins the network. Each device should be configured with a superframe and link tables beforehand. After joining the network, the device is in an *Idle* state.

One of the core functions inside the WirelessHART TDMA is slotTimeout function. This function is invoked periodically at the beginning of each time slot. In this function, the WirelessHART TDMA agent first decides whether the slot is assigned to particular node by checking the communication tables. If so then TDMA machine serves this event that indicates transmit or receive. If the receiving slot times out the device enters *Talk* state for transmitting the packet or otherwise it goes in the *Listen* state.

2.4.7 Routing in WirelessHART

According to the WHART specification (see [64]) there are two methods of routing packets in a WHART network, graph routing and source routing.

Figure 2.9 illustrates the source routing in which a source node sets a complete path of a set of forwarders toward the destination. For example to send a packet from Source1 to the gateway (GW), Source1 includes an ordered list ($Source1 -> F1 -> F4 -> GW$) of all forwarders devices through which the packet must be forwarded to the final destination. Source routing is less reliable as each pair of nodes has only one path available.

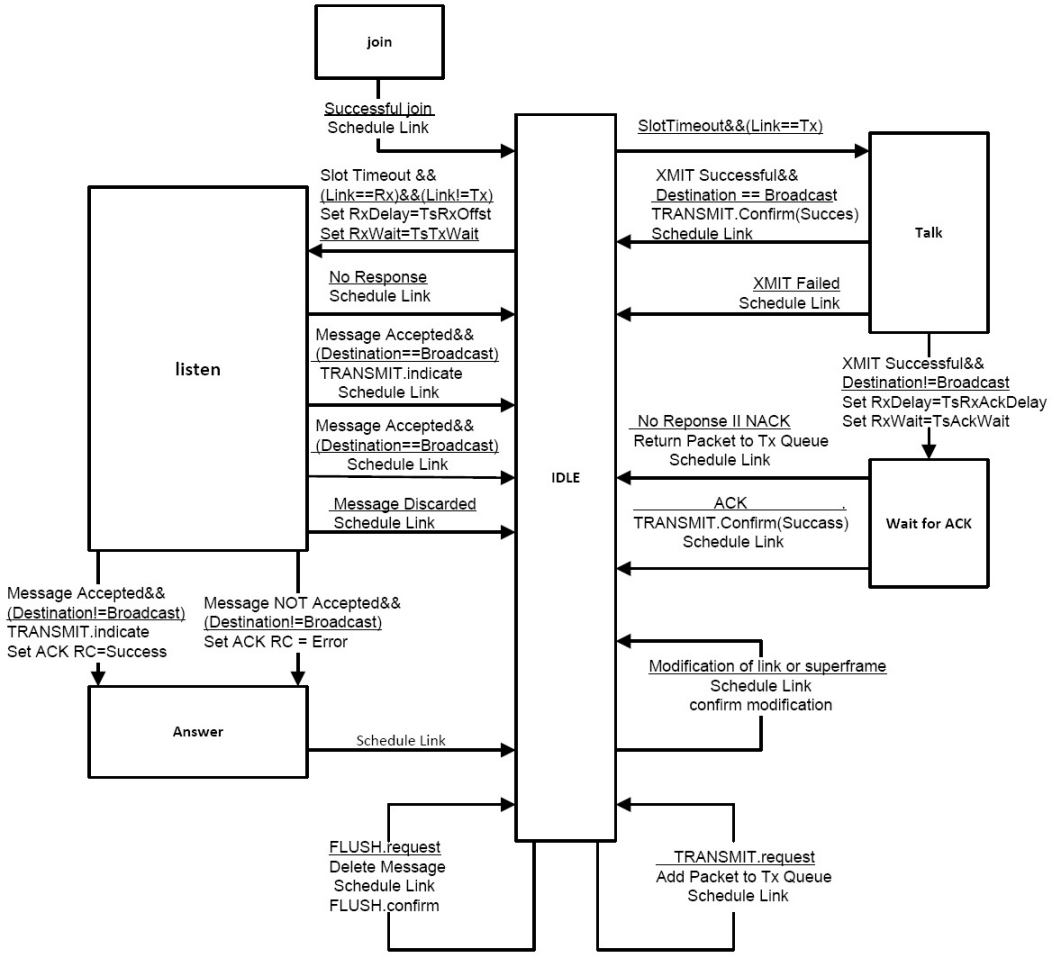


Figure 2.8: WirelessHART TDMA state machine [67].

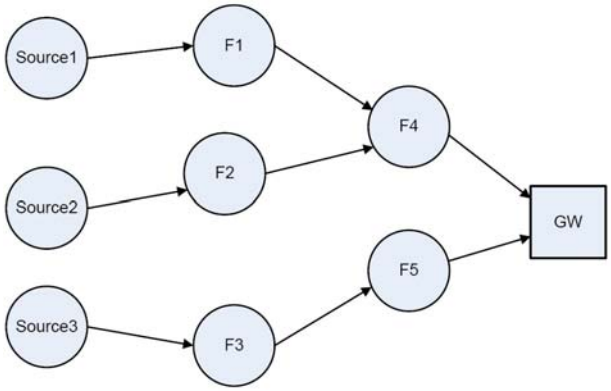


Figure 2.9: WHART routing graphs.

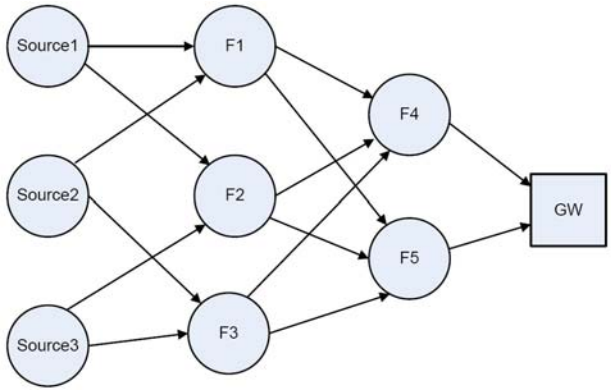


Figure 2.10: WHART routing graphs.

In graph routing (see Figure 2.10), the network manager pre-computes (and subsequently uploads to all nodes) a number of network graphs, so that any node in such a graph has at least two neighbors through which it can send frames to reach a given destination. When sending a frame, a node selects a suitable graph depending on the destination and includes the graph-id into the network-layer frame header. Both graph and source routing require the network manager to explicitly pre-configure routing tables for each potential destination by exploiting the link information provided by the nodes.

2.5 IEEE 802.15.4 Standard

The IEEE802.15.4 standard for Low-Rate Wireless Personal Area Networks (LR-WPANs) was first ratified in 2013. It provides both physical layer (PHY)

and media access control (MAC) specifications. The updated revision of the standard was released in 2006 (IEEE802.15.4-2006 [130]) and is the de-facto standard for low-power and short-range radio communications.

As shown in Table 2.1 the IEEE 802.15.4 may operate in three frequency bands. Among the three band frequencies listed in Table 2.1, the 2.4 GHz unlicensed Industrial, Scientific, Medical (ISM) frequency band is the most popular for wireless sensor networks applications. It offers higher radio data rate (compared to 868-969.6 MHz and 902 to 928 MHz), which reduces the frame transmission time and thus the energy per transmitted and received bit is lower.

Table 2.1: Frequency bands and data rates of IEEE802.15.4

Frequency bands and data rates of IEEE802.15.4			
Frequency Bands	Number of Channels	Data Rate	Channel spacing
868 to 969.6 MHz	1 (Channel 0)	20 kbps	–
902 to 928 MHz	10 (Channel 1 to 10)	40 kbps	2 MHz
2.4 GHz	16 (Channel 11 to 26)	250 kbps	5 MHz

In the 2.4 GHz band the Direct Sequency Spread Spectrum (DSSS) is adopted and can achieve up to 250kbit/s using the Offset Quadrature Phase-Shift Keying (O-QPSK) modulation. Each 4 data bits are mapped to a 32 bit sequence, denoted as a chip sequence. The data rate of the chip is 2 megachips per second which is equivalent to 250Kbits/s.

2.5.1 IEEE802.15.4 Devices and Topologies

There are two types of network nodes defined in the 802.15.4 standard namely Full-Function Devices (FFD) and Reduced-Function Devices (RFD). The FFD node(s) implements full protocol functionalities and may operate in three different modes: as a Personal Area Network (PAN) coordinator, router or as a device (compare Figure 2.11). The RFD can operate only as a device and implements a reduced set of the full protocol functionalities. As shown in Figure 2.11, the FFD may connect to any IEEE802.15.4 devices within its radio range. The FFD coordinator is the central entity of the network and responsible for the whole network configuration. The router device can relay packets on behave of other nodes and may connect to any FFD or RFD devices. The RFD is the last element of the network, it may transmit or receive packets but is not allowed to perform any routing activities. It should be connect to FFD device and can not support child devices. The main functionality of RFD device is to perform simple tasks such as sensing and data transfer for itself.

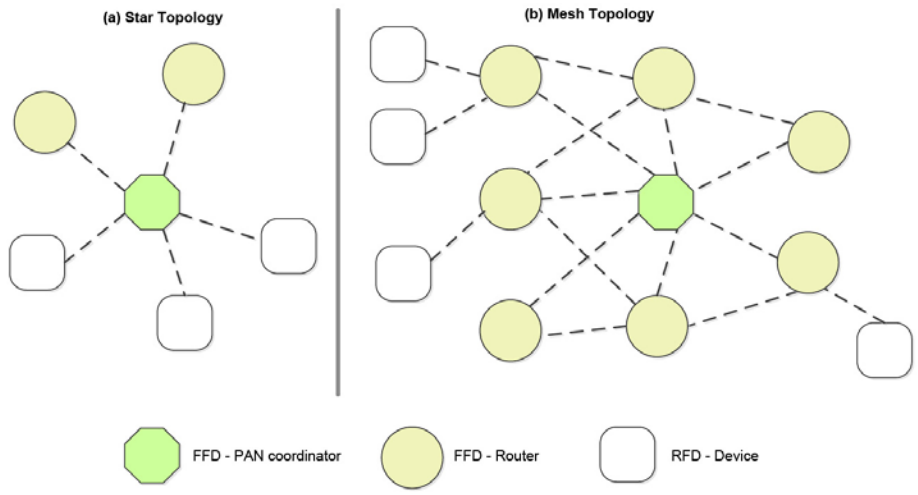


Figure 2.11: An example of IEEE802.15.4 topologies

As shown in Figure 2.11, the IEEE802.15.4 can support two main topologies, star topology and peer-to-peer topology. In the star toplogy (Figure 2.11 (a)) the RFD device can exchange information with the PAN coordinator only. In the peer-to-peer device any FFD nodes can talk to any other node within its radio rang and multi-hop netwok can be constructed. Moreover, peer-to-peer is very flexible and can allow to construct any other topology such as tree or mesh (see Figure 2.11 (b)).

2.5.2 IEEE802.15.4 Access Methods

The IEEE802.15.4 provides two main media access methods, CSMA and scheduled access. In both access method the coordinator can select among two modes: beacon-enabled and non-beacon enabled. The beacon-enabled mode is widely used in star topology in which a coordinator is connecting and managing the whole network. In the beacon-enabled mode, time is splitted into superframe which is repeated continuously. An example of a superframe structure is shown in Figure 2.12. Each superfreame is splitted into two period: an active period and inactive period. The active period is divided into 16 slots and the first slots is reserved for the beacon packet. At the beginning of each active period (slot 1), the coordinator is broadcast beacons information about its identity, synchronization, superframe structure and etc., without using any mechanism of accessing the media (no waiting or random access method is performed in this operation). The node may use the other slots to transmits its data using ether contention access period (CAP) or contention free period (CFP). During the CAP period the node allowed to transmit using a random

access method (will be explained in the non-beacon mode). In the CFP, the coordinator can reserve some Guaranteed Time slot (GTS) (up to seven slot in each superfame) to particular device(s) in which a node transmits a packet immediately without using any random access methods. All the nodes in the network enter sleep mode during the inactive period. Please note that the length of the superframe and its active and inactive periods are a configurable parameters and are not specified in the standard (depends in the application requirements).

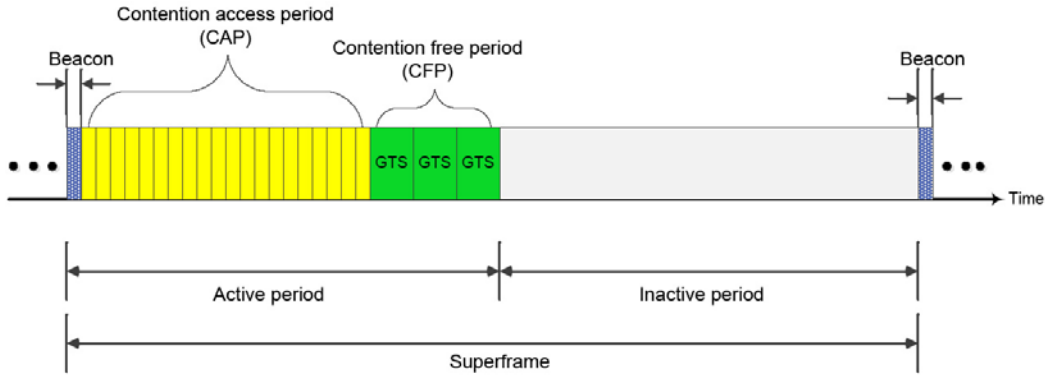


Figure 2.12: An example of IEEE802.15.4 superframe structure [130]

In the other hand, non-beacon mode is mostly used in peer-to-peer networks in which no central coordinator is required. There is no explicit time synchronization nor a superframe structure. All the operation of the network done using a random access approach and therefore no GTS slots are used. Since there is no time synchronization in the network, each node adapts a carrier sensing multiple access with collision avoidance (CSMA-CA) protocol to access the medium. When a node needs to transmit a packet, it first picks a random back-off value, once the back-off time is elapsed, it then performs a clear channel assessment (CCA) to check the channel activities. If the channel is idle, the node imminently transmits its packet, otherwise the node repeat the same procedure again but with different random back-off delay.

2.6 IEEE802.15.4e TSCH Protocol

Recently, another multi-channel TDMA-based solution is proposed. The Time Slotted Channel Hopping (TSCH) mode of the IEEE 802.15.4e MAC protocol [80] which is an amendment to the IEEE 802.15.4-2006 MAC protocol. The main goal of the IEEE802.15.4e is to support for industrial applications requirements such as communication reliability, low energy and security. The

main idea of the TSCH mode of the IEEE802.15.4e TSCH protocol is the use of the well known mechanism of channel hopping which enable the node to switch between the available 16 channels of the IEEE802.15.4 radio. Similar to the TSMP [134], WirelessHART [49] and ISA SP100.11a [84] standards the nodes in the TSCH network are synchronized and follow a TDMA-based schedule. In TSCH mode, the time is organized into superframe which continuously repeated over time. Each superframe is split into fixed time slot in which a node can be scheduled for specific activity (Tx,Rx,Listen,or Sleep). The size of the time slot is long enough to transmit or receive one single packet and its corresponding acknowledgment. One key challenge of the TSCH networks is the slot assignment algorithm in which a time slots and channels to be defined for each participant node.

The IEEE 802.15.4e standard assumes that the slot assignment algorithm is exist and defined a prior. Based on this assumption the IEEE 802.15.4e defined mechanisms in which a TSCH nodes can communicate and therefore does not specify how to build and maintain a communication scheduler. Thus, the TSCH mode brings several research challenges: first the need of slot assignment algorithms that take into account the application requirements such as traffic demands, energy consumption, and delay into account while building the communication scheduler.

Another questions that should be address as well are: the node discovery (how to discover new node), scanning and ranking of the channels, adapt the network to in going change such as traffic load and topology change, maintaining clock synchronization and network statistics. All these challenges and others are addressed in the currant active internet draft. In this draft a new sub-layer denoted as 6top is defined.

As shown in Figure 2.13, the 6top sub-layer interacts between the network and the MAC layers and responsible for the whole communications schedule. As discussed in [157] and [171], 6top provides interfaces (data and management) to network and MAC layers. It might includes scanning and motoring of channels qualities, slot assignment algorithms, and etc.

The main advantage of the TSCH is its interoperability feature. For example, TSMP, ISA100.11a, WirelessHART and IEEE802.15.4e using more or less the same mechanisms of time synchronization and channel hopping but they different protocols in that they do not inter-operate [108]; If a WHART node deployed next to an ISA100 node, they will never communicate with each other as they use different packet formats. However by employing the TSCH mechanisms, these standards will be able to inter-operate.

Moreover the 6TSCH might reuse some of the existing protocols which already defined in the IETF, specifically, the following three groups: Routing Over Low-power and Lossy networks (ROLL), Constrained RESTful Environments (CoRE) and Low power Wireless Personal Area Networks (6LoWPAN)

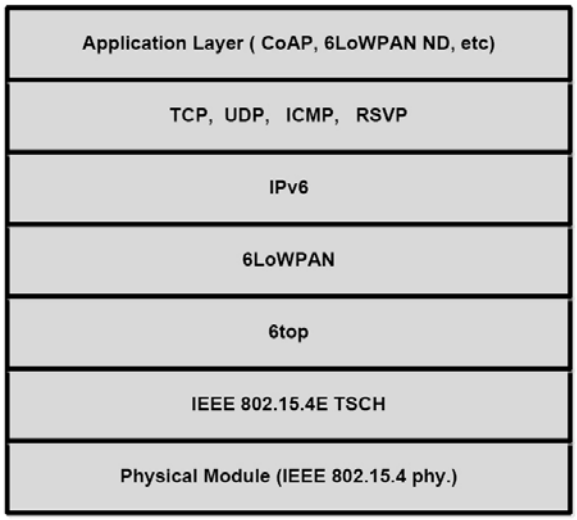


Figure 2.13: Example of TSCH protocol stack [158].

to provide Internet connectivity for wireless sensor networks devices. The ROLL working group is formed to provide a standardized routing protocol for WSNs [76, 174]. The goal of the CoRE working group is to standardize an application layer protocol for WSNs [147]. Finally, the aim of the 6LoWPAN working group is to enable end-to-end IPv6 connectivity for WSNs, particularly, it deals with the issue of integration of IPv6 datagrams with the IEEE802.15.4 frame format [102]. This include header compression and fragmentation mechanisms to reduce the 1280 bytes of IPv6 packet to the 127 bytes of the IEEE802.15.4 packet size.

Another channel hopping approach was recently proposed by Al Nahas et al., called MiCMac [124]. MiCMAC is an extension version of the low-power listening that utilizes channel hopping without maintaining and explicit time synchronization. In this protocol, the sender needs to make sure that the receiver node is awake in the same channel before sending data; it thus transmit periodic strobes for each channel. The sender also listens (for ACK from the receiver) between each strobe in order to make sure that the receiver node selects the same channel before sending the data packet. However, in MiCMAC the sender and receiver nodes are expected to drain significant amount of energy due to periodic strobes and occupation of the transmission channels for hopping synchronization.

CHAPTER 3

Scope of the Thesis

We organize this chapter around the following steps: We first give a high level description of the research challenges, hypothesis and the system under consideration. Then we present the methodology approaches common to subsequent chapters for both our proposed solution, autonomous framework (Chapters 4, 5 and 7) and the benchmark protocol, WirelessHART (Chapters 6 and 7). We also present the general performance evaluation approach, metrics and model validations for both autonomous and WirelessHART systems.

3.1 Problem Statement and Design Objectives

We consider a multi-hop sensor network supposed to carry periodic traffic, coming from different sources and at different rates. A very important design goal is to run the network as energy-efficient as possible while supporting low delay and high communication reliability. Figure 3.1 illustrates an example of a multi-hop network which includes the three main players in our system model. Source nodes send their data periodically to a single sink through a number of forwarder nodes [41]. In most multi-hop wireless sensor network scenarios [112, 14], forwarder nodes are responsible for two main tasks: (a) obviously forwarding the data to the next hop reliably and (b) minimizing their energy usage mainly by switching to an energy-conserving sleep state as often as possible to prolong their (and the networks) lifetimes. There are many definitions of the network lifetime in WSN literature. For example, the authors in [168, 53] defined the network lifetime as, the time until the first sensor node dies. In [118], the authors defined the network lifetime as the time the area of interest is covered by at least N number of nodes. The time each target is covered by at least single node [25]. The percentage of node that have a path to the sink node [24]. The authors in [86, 117] combine two factors to define the lifetime of WSN specifically, "lifetime is defined as the

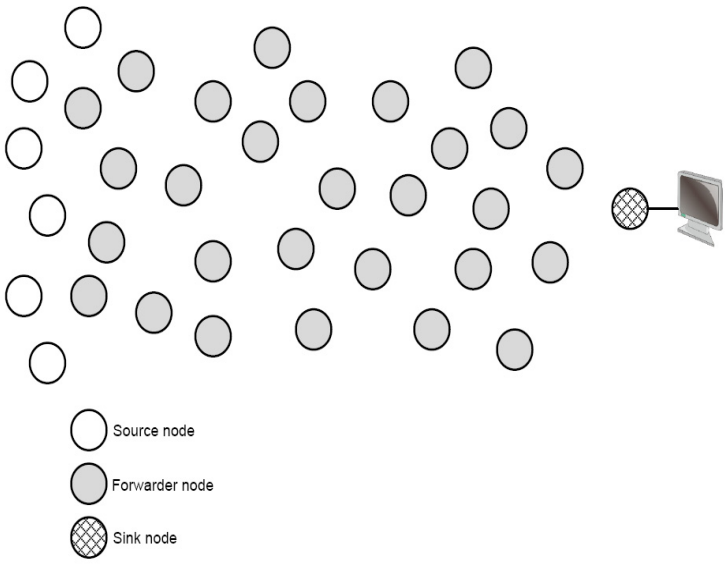


Figure 3.1: An example of multi-hop network scenario

number of successful data gathering cycles that are possible until connectivity and/or coverage are lost.". We believe that, whatever the definition of network lifetime, it is strongly related to the nodes' energy consumption.

Despite the fact that source nodes transmit their data periodically, the time differences between packet arrival times seen by a forwarder are not ideally regular but have a random component, see [29, 92, 96] and Chapter 4. As discussed before, this might be caused by the randomized operations of MAC protocols, time-varying cross-traffic (resulting in queueing effects), or Operating System (OS) imperfections (i.e. interrupts handling). We defined the jitter as the deviation from the ideal time of the traffic periodicity. Intuitively, one might expect that the amount of jitter is a function of the number of hops a packet traverses. To accommodate this, a forwarder would have to wake-up earlier and for longer time than a forwarder closer to the source.

Moreover, source nodes may increase or decrease their traffic period upon the occurrence of new event of interest or upon re-configuration. For instance a source node may start with very light traffic load (sampling temperature every 1 minute) and when the source node detects an event of interest then it begins to report its readings at a higher rate (for example: sensing and reporting a light sensor each 1sec).

Furthermore, wireless sensor nodes may decide to change their traffic rate not only to cope with a change of an event of interest but also to prevent partitioning of the whole network due to battery depletion of forwarder nodes important for connectivity. For example, a node might decide to change its

traffic rate to a lower one temporarily, when the battery drops below a certain level [26, 156, 1].

Therefore, decreasing the sampling rate in such situation not only increases the network lifetime but also ensures enough time for maintaining the network (for instance replacing batteries), and maintain the fine granularity of the information gathered from such source nodes. Thus, the periodicity of data sensing and reporting may be different over time and it depends on the application and the capabilities of the source nodes. Hence, allowing the forwarder node to adapt to the traffic change in an agile manner while consuming the least amount of energy during this phase is another important design goal.

Another important design objective is to support communication reliability which is defined as the percentage of the successfully received packets relative to the total transmitted packets. One important approach to improve reliability is to exploit frequency diversity by periodically changing the communication channel. As previously discussed, channel hopping is known to substantially improve communication reliability in WSNs and it has been adopted in recent standards for industrial wireless sensor networks, for example Wireless-HART and ISA100.11a [99, 66, 189, 83, 28]. To achieve similar levels of reliability and to also ensure a fair comparison of our autonomous framework to WirelessHART a frequency-hopping approach is adopted.

The following points summarize the main requirements and weaknesses of centralized TDMA protocols in multi-hop WSNs:

Tight time synchronization: TDMA-based protocols require a tight time synchronization in order to be able to switch between different channels at the right times and to avoid overlaps of time slots [170]. Such tight time synchronization requires specialized protocol mechanisms for time synchronization, which add to the overall overhead. For example, many protocols (including WirelessHART) rely on periodically exchanging packets containing timestamps [57, 177]. In low-traffic scenarios these synchronization packets might even require more energy than the actual data packets [10]. For instance and according to the WirelessHART standard, nodes need to re-synchronize every 30s, even if there is no need to send data packets in the near future [64]. Thus, re-synchronization packets not only increase energy consumption but also the packet delay as well [114].

Signalling overhead: In centralized TDMA protocols scheduling decisions are made at a single point, the network manager. It requires both global knowledge of topology and traffic demand of each node, which the network manager needs to collect from the network nodes, costing time and energy. In the other direction, the coordinator needs to disseminate the computed schedules back to the nodes. Moreover, each node

needs to periodically send control packets (each 30 second according the WirelessHART standard) not only for forming the network but also for maintaining the overall operation of the network, this could include: link status reports, health report updates and keep-alive packets. This control overhead is costly in terms of energy and communication resources [143, 34, 87].

Centralized network manager: Any change in traffic demands requested by a sensor node needs to be signaled to the central network manager, which computes a new schedule (possibly affecting other nodes as well) and disseminates them. This can incur a substantial delay and overhead before changes become effective. Furthermore, a central network manager also represents a single point of failure [76].

Implementation Complexity: In general, centralized TDMA protocols for multi-hop wireless networks require a high implementation complexity as they rely on both tight time synchronization and slot time assignment algorithms [98, 34, 57]. Generally speaking, code size is one direct indication of the complexity of a protocol. Reducing protocol complexity reduces the code size, the amount of dynamic memory needed and the probability of inconsistent network state [131]. This is important since wireless sensor nodes usually operate with very limited memory (RAM and ROM) size. The author of [59] shows that implementing a reduced version of WirelessHART using Tmote Sky [122] nodes requires about 11,876 bytes which is approximately 8 times more than a classical CSMA/CA protocol for WSN needs. Another study [131] reports that implementing the PracMac [107] protocol (which is a small version of a scheduled multi-channel protocol) requires almost 5 times more RAM and more than two times more code memory than the CSMA-based B-MAC protocol. Both the B-MAC and PracMAC protocols were implemented and compared in TinyOS, the total code size of B-Mac is 3,212 bytes and the total size of PracMac is 10,305 bytes.

Scalability and Adaptability: There is a wide class of applications [173, 109, 72, 181] requiring large-scale deployments (hundreds or thousands of nodes) of sensor networks. Therefore, scalability is an important factor to guarantee that the network performance does not change significantly as the number of nodes increases. At the same time the whole network should adapt to any change in the network (for example, when node join or leave) in an efficient manner. However, since centralized protocols such as [39, 19, 142, 46] require communication schedules to be computed and distributed in advance, it is relatively costly in terms of energy and delay to adapt to any changes in network typology or load situations

[34, 143]. In addition, adaptability of the network in case of change in traffic loads usually requires long response time as the node has first to request additional time slots and then wait for a schedule update from the centralized authority; this renders centralizes TDMA solutions less attractive in large-scale WSNs [12, 186, 185].

3.1.1 Hypothesis and Assumptions

Our hypothesis is that we can achieve our aims (energy-efficiency, low delay and communication reliability) by exploiting the characteristics of periodic traffic and by scheduling the node activities in a decentralized way much faster and much cheaper than communicating and computing a new schedule using centralized approaches. We believe that the full centralized TDMA operation (including time synchronization and the construction and maintenance of communication schedules) represents too much overhead for periodic traffic in multi-hop WSNs, especially for lower to moderate overall traffic load. Instead, each node tries to estimate and learn its incoming traffic flows locally. Basically, by estimating the period of an incoming traffic flow a forwarder node can locally and autonomously schedule its duty cycle such that it wakes up before the expected (from its measurements) arrival of the next packet, handles it and returns to sleep. The key assumption of our solution is that the forwarders do not know the traffic characteristics beforehand, but they have to estimate it and maintain their estimate over time. Another important assumption is that we do not have any kind of explicit time synchronization protocols. To keep the protocol complexity low we also assume a Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA)-based MAC protocol and the hidden terminal problem is excluded (all sender nodes can hear each other). Moreover, all the nodes use the same hardware.

We also make the following assumptions about the type of application and traffic model for both autonomous framework and the benchmark protocol.

Application Scenario: As mentioned in the introduction we consider applications in which periodic reporting from sensors to a gateway / sink node in a multi-hop WSN is pre-dominant. The source nodes sense some physical quantity (for instance light, temperature, illumination sensors or specific valve data) and their goal is to transmit this data via some forwarder nodes to the gateway node reliably and in an energy-efficient manner.

Traffic type: The traffic is periodic with varying traffic generation rates. This type of traffic is essential for many applications such as environmental monitoring, industrial sensing and diagnostics in which the source nodes

may increase or decrease their traffic periodicity due to change of the nature of the monitoring environment.

Radio: We assume that the physical layer is given by the physical layer offered by the IEEE 802.15.4 standard operating in the 2.4 GHz ISM band. This standard offers a number of different orthogonal frequency channels with a center frequency separation of 5 MHz for adjacent channels (see table 3.2).

Routing: We assume for both systems that routing is carried out by algorithms and means outside the scope of this thesis. For both, autonomous framework and WirelessHART we assume that each node knows the next hop towards the sink node, from adopting tree-like routing protocols like the CTP protocol from TinyOS [48]. The overhead of routing is excluded from the whole analysis and evaluation of both systems.

Please note that more assumptions will be listed for each system (autonomous framework and WirelessHART) in their respective sections.

3.2 Research Challenges

In order to address these challenges, we have developed a distributed and self-learning framework integrating asynchronous channel hopping, estimation of periods and adaptation to several flows, local dynamic multiple sleep states scheduling and on-the-fly traffic adaptation mechanism.

To develop and integrate these approaches, some significant challenges have to be addressed.

Challenge 1: Jitter: Period estimation and the scheduling of wakeup times will have to deal with jitter in the packet inter-arrival times. If a packet arrives before the forwarder wakes up or after it has returned to sleep, it is lost. This opens up a trade-off between loss rates and the sleeping activities of the forwarder: when the forwarder wakes up "early", the packet loss rate will be low but the forwarder spends more energy, and vice versa. This tradeoff needs to be assessed and properly controlled.

Challenge 2: Asynchronous channel hopping: Enabling nodes to switch between different channels without maintaining explicit time synchronization is difficult; since collisions are possible across different neighbors while hopping on different frequencies in a asynchronous manner. To the best of our knowledge asynchronous channel hopping has not been addressed in the WSN literature so far (at the time of this writing).

Challenge 4: Adaptability to changing traffic conditions: Source nodes may have different traffic requirements which might change over time. This change could be due to the change in sensing phenomena or due to low battery levels. Thus, the traffic generation rates of the sources can be vastly different over time which implies the need for a fast adaptive traffic policy. Therefore, designing an agile adaptation mechanism avoiding long delays (as they can occur in centralized systems due to the need to consult the network manager to re-schedule parts or whole of the network) and having tolerable overhead is another challenging task.

Challenge 3: Multi-flows overlapping: Certain forwarders might be placed on the routes for several distinct sources and must adapt both their sleep-/wakeup windows and also the frequency channel, especially in situations where packets of different source flows "collide" at a forwarder, i.e. are to be received at about the same time. To solve this issue we propose to separate each flow by associated flow IDs and then find potential overlap beforehand by comparing the flows characteristics. The details of this approach will be explained in Section 5.4

3.3 Performance Evaluation Methodology

Selecting the right methodology for evaluating the above schemes is not trivial. On one hand, theoretical channel models usually do not capture complex phenomena such as multi-path fading, or the impact of a dynamic environment. On the other hand, real-life experiments do not easily provide the ability to evaluate different schemes or algorithms under the exact same conditions, as the RF environment is time-varying. We decided to combine these two methods and to use connectivity traces gathered from a real-world deployment as an input to our simulations.

3.3.1 TWIST Testbed

We carry out our evaluation using the TKN WIreless Sensor network Testbed (TWIST) testbed (Telecommunication Network Group (TKN) WSNs Testbed) [62]. It has approximately 102 Tmote sky nodes spread over three floors of our FT building at the TU Berlin campus (see Figure 3.2). Each mote is integrated with the popular IEEE 802.15.4-compliant ChipCon CC2420 radio transceiver [30] operating in the 2.4 GHz ISM band and has a data rate of 256 kbps. The transceiver supports 16 channels (from 11 to 26) in the 2.4 GHz band, with a center frequency separation of 5 MHz for adjacent channels. There are some obstacles in the TWIST testbed area that could impede RF communication and cause multi-path reflections. In addition, the building is occupied with

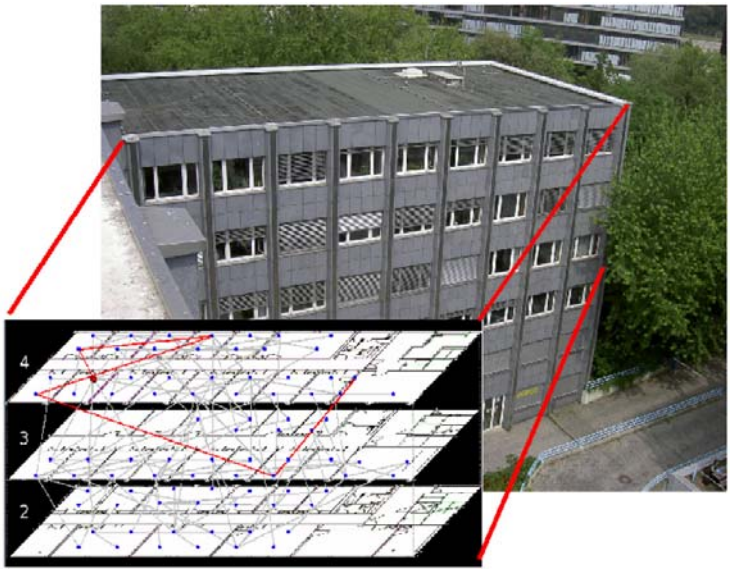


Figure 3.2: Example of TWIST Testbed [100].

some WiFi access points which may introduce external interference to the TWIST network.

We use Tiny Operating System (TinyOS) version 2.0 [73, 51] and its default protocol stack, which uses a Carrier Sense Multiple Access (CSMA)-type MAC protocol(B-Mac [136]). Every node waits a random short amount of time before transmission and it goes into back-off and picks another random time if the channel is still busy. The CC2420 radio stack can select one of two back-off periods: initialBackoff or congestionBackoff [121, 120]: initialBackoff is the shortest back-off period and is used on the first attempt to transmit a packet. The congestionBackoff is a longer back-off period used when the channel is found to be busy. We also use the well known Collection Tree Protocol (CTP) routing protocol [55] to construct routes between the sources and the sink.

3.3.2 Connectivity Traces

We evaluate both autonomous framework and WirelessHART systems using connectivity traces gathered in a real-world deployment. The connectivity traces have been collected by the DUST networks group [36]. The measurements were conducted in a printing factory in Berkeley, California, and data was collected over 26 days. The building has a rectangular footprint, measuring 250 feet x 225 feet. There are many obstacles in the work area that could impede RF communication and cause multi-path reflections. Overall, 45 sensor nodes were deployed with a relatively uniform distribution. Each node

is equipped with a ChipCon CC2420 radio chip [30]. For this experiment, the data consisted of periodic reports of the quality of the communication path, where a path represents all transmission between a pair of nodes. The nodes had up to eight neighbours and communicated on all of the 16 channels supported by IEEE802.15.4. The trace contains all path-channel reports. For more details about the setup of the experiment please refer to [36].

3.3.3 Simulation

In order to realize a simulation model to study the performance of the autonomous framework and the WirelessHART benchmark protocol over a wireless multi-hop network, we have chosen the OMNeT++ [167] simulation environment together with the Castalia framework [128]. OMNet++ is an open-source discrete-event simulator, Castalia is an OMNet++ based framework designed specifically for wireless sensor networks.

Unless otherwise specified we set the radio parameters based on the IEEE 802.15.4-compliant ChipCon CC2420 radio chip [30]. The CC2420 operates in the 2.4 GHz ISM band and supports eight transmission power settings in the range between -25 dBm and 0 dBm.

For channel errors (unless otherwise specified) we use a channel model provided by Castalia (please refer to the Castalia users manual [128]). In particular we use the log-normal shadowing model which has been presented as a reasonable model for the average path loss in [113, 18]. Generally speaking, the path loss PL is a function of the distance d from the transmitter as shown in the following equation:

$$PL(d) = PL(d_0) + 10 \cdot \eta \cdot \log\left(\frac{d}{d_0}\right) + X_\sigma, \tag{3.1}$$

where $PL(d)$ is the path loss at distance d, $PL(d_0)$ is the known path loss at a reference distance d_0 (which we assume to be 1 m), η is the path loss exponent, and X_σ is a Gaussian zero-mean random variable with standard deviation $\sigma > 0$. For the channel fading model we use a block Rayleigh fading model in which the Rayleigh-distributed fading gain changes for each slot or time period. The values of the channel parameters are listed in Table 3.3. From the path loss, the current fading gain, and knowledge of the transmit power it then becomes possible to determine the Signal to Noise Ratio (SNR) at the receiver and, by knowledge of the modulation scheme, subsequently the instantaneous Bit Error Rate (BER).

The parameters of the path loss model (Equation 3.1) are set according to [113]: $PL(d_0) = 55$ dB, $\eta = 2.4$, $\sigma = 4.0$. Please note that links between two nodes can be asymmetric, as each direction of such a link uses its own

realization of the random shadowing coefficient X_σ. Moreover, we have defined -94dBm as the sensitivity threshold of the CC2420 Radio, which is the same value as specified in the data sheet. These calculations are carried out within Castalia.

For the trace-based simulation we combine the traces introduced above using the following method. For each link and each channel we change every 15 (simulated) minutes the packet delivery ratio by reading the next value for the packet delivery ratio for this link and channel from the trace files.

For the implementation of both our autonomous framework and WirelessHART we have modified the radio and channel models of the simulator to support the 16 channels of the IEEE 802.15.4 standard. We have also modeled the cost of channel switching. Channel switching is modeled in both receiving and transmission slots within the TxOffset and RxOffset time intervals, respectively (see Section 2.4.4). Based on the characteristics of the CC2420 we have assumed that channel switching takes $192\mu s$ and consumes the same amount of power as the transceiver's listen state. The transceiver has four main operational states: transmit, receive, listen and sleep with different power consumption parameters, see Table 3.1.

In addition to the CC2420 transceiver we have also taken the power consumption of the microcontroller into account. More precisely, we use a MSP430 microcontroller from Texas Instruments, Inc. (TI). This controller alternates between two different states: an active state, in which it performs computations, and a sleep state in which major parts of its circuitry are switched down. In Table 3.1 the main power consumption parameters of a CC2420 transceiver and MSP430 microcontroller are summarized assuming a 3.3V supply voltage. These parameters are used in the physical layer model of our simulator to obtain energy-consumption results. Please note that the microcontroller is active at the same times as the transceiver. Please note that more details of the simulation setup for autonomous framework and the WirelessHART will be described in their respective chapter.

3.3.4 Network Topology and Routing

In order to perform our evaluation, we use randomly generated topologies. More precisely, we have generated 100 random topologies and for each setting of simulation parameters we correspondingly perform 100 replications. For each random topology we have placed 150 nodes in an area of size $120 \times 120\text{m}^2$, using a uniform distribution for node positions. As shown in Figure 3.3 the gateway is placed in the upper right corner of the field. Out of the 150 nodes we randomly pick ten nodes as source nodes. Each of these sources periodically generates frames of 133 bytes total size (including PHY and MAC parts). Since it is already implemented in both TinyOS and Castalia simulator we use the

Table 3.1: CC2420 Power Consumption Parameters and of the MSP 430 Microcontroller with 3.3 V supply voltage

Main power consumption parameters of CC2420 radio			
Notation	Parameters	I(mA)	Power(mW)
P_{Tx}	Transmit power (0dBm)	17.4	57.42
P_{Rx}	Receive power	18.8	62.04
P_L	Listen power	18.8	62.04
P_{S-m1}	Sleep-mode-1 power	0.426	1.406
P_{S-m2}	Sleep-mode-2 power	0.02	0.07
P_{s-m3}	Sleep-mode-3 power	$2.0 \cdot 10^{-5}$	$6.6 \cdot 10^{-5}$
CPU_A	CPU active power	1.8	6
CPU_S	CPU sleep power	0.045	0.148

Table 3.2: Physical channel of the IEEE 802.15.4

Index	Channel (802.15.4)	Frequency (MHz)
0	11	2405
1	12	2410
2	13	2415
3	14	2420
4	15	2425
5	16	2430
6	17	2435
7	18	2440
8	19	2445
9	20	2450
10	21	2455
11	22	2460
12	23	2465
13	24	2470
14	25	2475
15	26	2480

Table 3.3: General parameters.

Main Radio and MAC parameters	
Parameter	Value
Radio layer	CC2420
Data rate	250kbps
Frame size	133
CC2420 sensitivity	-95dBm
Noise floor	-100dBm
$PL(d_0)$	55dB
η	2.4
X_σ	4.0

collection tree protocol CTP for constructing the routing among the nodes. Please note that the majority of network typologies are tree, however we used linear typologies in some evaluation scenarios.

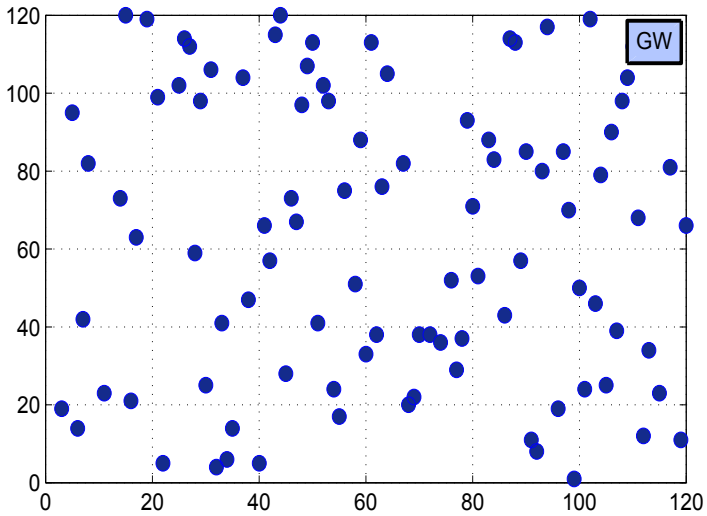


Figure 3.3: Example of a random scenario.

3.3.5 Major System Metrics

In this section, we list the main performance metrics for both the autonomous framework and the benchmark WirelessHART system. Specific metrics are explained for each system in their respective chapters. The simulation time is fixed to 168 hours (one week) and the three main performance measure are the total energy spent by the radio transceiver of a node over this period. The second metric is the end-to-end packet delivery ratio (PDR), i.e. the fraction of all packets sent by the sources that reach the destination. On one or two occasions we also use the end-to-end packet loss rate, which is just the complement of the packet delivery ratio, as a performance measure. The third important performance measure is the average per-hop packet delay. To derive the end-to-end packet delay, we sum the contribution of each per-hop packet delay.

The simulation records the amount of time spent in various states (transmit, receive, listen, sleep and turnover) and calculates from this the total energy consumption of a node over a span of 168 simulated hours. We also take into consideration the energy consumed by the node's microcontroller. We split the microcontroller energy consumption in two main states, active state and sleep state. The microcontroller is active at the same time as the radio. At the end of each run, the simulation computes the total energy consumed for all nodes in the network using the amount of energy consumed by the radio and microcontroller in each state. More recently, authors in [37] proposed an emulation-based tool for computing energy consumption of the wireless sensor nodes. It uses a non-ideal battery model. This method has been available only recently and it remains to be seen how the WSNs community adopts this interesting tool.

CHAPTER 4

Autonomous Framework: Overview

In this chapter we introduce and evaluate our self-adaptive and self-learning approach under single channel scenario to support energy efficiency under periodic traffics. There is no explicit time synchronization, but instead each forwarder learns the traffic period and jitter distribution from observing the traffic in distributed manner. Based on this information a forwarder determines suitable times for sleeping and for waking up to receive the next packet. We then proceed with an evaluation of our approach. Please note that in this chapter we keep the discussion somewhat limited to a single flow scenario.

4.1 Distributed Wakeup/Sleep Scheduling Approach

We consider a multi-hop wireless sensor network in some source nodes generate periodic traffic, which needs to be carried to a sink node. The network also comprises forwarder nodes which help with forwarding the traffic from the sources to the sink. Each source wakes up periodically, samples the environment through its local sensors, transmits a packet carrying the sensor data to the next forwarder on its path to the sink, and goes back to sleep. Each source node can have its own period and there is no common time reference in the network, nor is there any other synchronization mechanism present. We only require that sources have unique identifiers which they include in their packets.

A forwarder has the goal to stay most of its time in a sleep state. Since the source traffic is periodic, a forwarder can expect to see traffic that is approximately periodic, but with non-negligible jitter. As was already discussed before

the jitter can for example result from the operation of the underlying MAC protocol (e.g. when random backoff times are used like in the IEEE 802.15.4 CSMA MAC) or from cross-traffic and queueing effects in upstream nodes. Hence, the interarrival time distribution seen by a forwarder is not impulse-like, but instead has some variability around its average (which is the source period).

The main design goal of our dynamic sleep/wake-up time scheduling solution is to enable forwarder nodes to acquire knowledge of the traffic period and its jitter in a distributed way. We want to use this knowledge to let a forwarder node wake up and sleep "just at the right time". The "right time" to wake up is at the nominal arrival time of the next periodic packet minus some safety margin to avoid packet loss because of waking up too late. On the other hand, the "right time" to go back to sleep mode is at the nominal arrival time of the next periodic packet plus some safety margin to avoid packet loss because of waking-up too late. The size of this safety margin in relation to the duty cycle has a direct influence on the forwarders energy consumption as well as its packet losses: the larger it is the more energy it spends and less packet losses incurred and vice versa. The safety margin in general depends on the amount of jitter seen by a node and the percentage of sleeping-induced losses that can be tolerated.

4.1.1 Estimators

Assume that there are N forwarder nodes. The forwarder node i, where $1 \leq i \leq N$ receives packets at times $(t_i n)_{n\geq 1} = (t_{i1}, t_{i2}, t_{i3}, \ldots)$. These packets carry link-local sequence numbers $(s_i n)_{=} (s_{i1}, s_{i2}, s_{i3}, \ldots)$ which for simplicity we assume monotonically increasing – to ease presentation we avoid here the issue of overruns, but it has been considered in our experimental implementation. The first building block of our scheduling approach is the design of appropriate estimators. The following estimators are only updated after each packet arrival (ordered pair), if there is a gap then we just drop it form the calculation. (i.e. when the packet with sequence number s_{in} has arrived at time t_{in}, the estimators uses the values $t_{i1}, t_{i2}, \ldots, t_{in}$ and $s_{i1}, s_{i2}, \ldots, s_{in}$):

- A period estimator $\widehat{p_i}(n)$ returns an estimate of the period. The general form of the period estimator is:

$$\widehat{p_i}(n) = P(t_{i1}, \ldots, t_{in}, s_{i1}, \ldots, s_{in})$$

- A quantile estimator returns the estimate α-quantile of the jitter distribution. The general form of the quantile estimator is

$$\widehat{q_i}(n;\alpha) = Q\left((t_{in_2} - t_{in_1}) - \widehat{p_i}(n), (t_{in_3} - t_{in_2}) - \widehat{p_i}(n), \ldots (t_{in_k} - t_{in_{k-1}}) - \widehat{p}(n); \alpha\right)$$

where the subsequence $(t_{n_k})_{k\geq 1}$ consists of those timestamps for which $t_{n_{2k}}$ and $t_{n_{2k-1}}$ belong to successive packets (i.e. for which $s_{n_{2k}} - s_{n_{2k-1}} = 1$ holds).

It depends on the assumptions on the jitter distribution how many quantiles must be estimated simultaneously:

- If the jitter distribution is assumed to be symmetric, only one quantile must be estimated. Given a prescribed allowed loss rate of $L_{\max}$, the quantile estimation would be applied with $\alpha = L_{\max}/2$ to account for the fact that losses can be incurred either because the forwarder awakes too late or goes back to sleep too early.
- In the more general case of a non-symmetric distribution, two different quantiles would have to be estimated: $\alpha_1 = L_{\max}/2$ for the lower part, and $\alpha_2 = 1 - L_{\max}/2$ for the upper part of the jitter distribution.

- A loss-rate estimator computes the local loss rate between the forwarder and its successor. It mainly operates on the sequence numbers, but since the sequence number space is in general finite and ambiguities might occur, the packet arrival timestamps are also taken into account, i.e. the general form of the loss-rate estimator is

$$\widehat{l}(n) = L(t_1, \ldots, t_n, s_1, \ldots, s_n)$$

4.1.2 Node states

The second major building block is the introduction of two separate node states: the *acquisition state* and the *operational state*. When the forwarder knows the interarrival time distribution, it can obtain the period as the average of this distribution. Furthermore, knowing the lower $\alpha/2$ and upper $\alpha/2$ quantiles (for $0 < \alpha < 1$), the forwarder can schedule its sleeping activities such that it wakes up at the lower $\alpha/2$ quantile, waits for a packet to forward, forwards it (if any), and goes back to sleep at the upper $\alpha/2$ quantile. Following this approach guarantees that no more than a fraction of α packets are lost from the forwarders sleeping activities.

Clearly, in the absence of any synchronization and signaling, the forwarder does not know the interarrival time distribution a priori. Therefore, a forwarder alternates between two different states: the *learning state* and the *operational state*.

- In the **learning state** the forwarder node does not sleep but unconditionally tries to capture all packets in order to obtain reliable first estimates of the period and the relevant quantiles.

This state is entered after the node has been switched on, or too many losses have been incurred during the operational state. The latter can occur for example when the cross-traffic situation and therefore the actual jitter variance changes. The end of the acquisition state is determined by a *stopping rule*, which in general can take the achieved accuracy of the period- and variance-estimator into account or can simply stop after recording a prescribed number of packets. Once these estimates are reliable enough, the forwarder enters the other state, called the **operational state**.

- In the **operational state** the sleep-/wakeup scheduling is applied. The forwarder follows the sleep/wakeup cycle, where it wakes up at the lower $\alpha/2$ quantile and returns to sleep at the upper $\alpha/2$ quantile. Furthermore, the forwarder observes the packet loss rate in the operational state and continues to update the estimates of the period and the quantiles (we refer to this as **statistics update**). If the packet loss rate grows too large, the forwarder returns to the learning state in order to re-estimate period and quantiles. This allows forwarders to adapt to changes in topology or load scenario. We also introduce other scheme in which forwarder nodes enter the learning phase upon a change in the traffic requirements immediately (without the need to observe the packet loss rate). This will be detailed in Chapter 7.

  The duration of the learning state and the precise consecutive packet loss rate threshold triggering the transition back from the operational into the learning state are design parameters of the scheme.

  - The forwarder maintains three main predicted times for each flow (see Figure 4.1):
    - $t_w(n)$ refers to the wakeup time of the n-th activity phase (i.e. it denotes the start of the activity phase)
    - $t_s(n)$ refers to the sleep time (denoting the maximum end time of the n-th activity phase), and
    - $t_a(n)$ is the nominal packet arrival time for the n-th activity phase.

For the n-th activity phase, the forwarder schedules wakeup for the time $t_w(n)$. The forwarder remains awake until either a packet is received and forwarded, or until sleep time $t_s(n)$ is triggered. At the end of the activity phase the forwarder updates its estimates of the period $\widehat{p_i}(n)$, jitter $\widehat{q}(k)$ and loss rate $\widehat{l}(n)$. From the updated jitter the quantiles $\widehat{q}(k)$ of the jitter distribution are updated. Based on this, the times for the

$n+1$-st activity period are calculated as follows :

$$\begin{aligned} t_a(n+1) &= t_a(n) + \widehat{p}(n) \\ t_w(n+1) &= t_a(n+1) - \widehat{q}(n) \\ t_s(n+1) &= t_a(n+1) + \widehat{q}(n) \end{aligned}$$

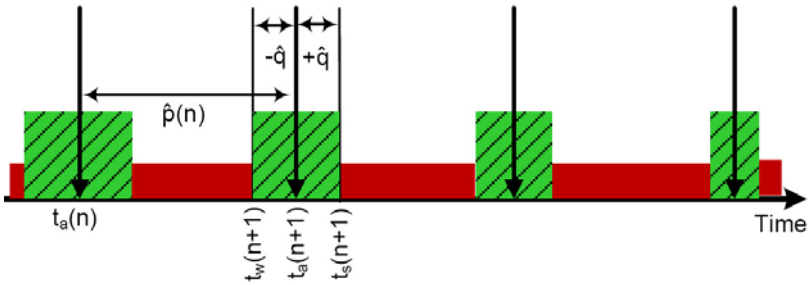


Figure 4.1: The figure shows the three predicted timers $t_a(n), t_w(n),$ and $t_s(n)$ for the wakeup and sleep scheduling algorithm

As the second major action at the end of the activity phase, the forwarder decides about a possible state transition back into the learning state. We refer to the rule used for this as the *transition rule*.

Please note that in this approach the forwarder continuously updates its period and jitter variance estimates in the operational state. We later on show results that confirm the necessity of these continuous updates. For later reference, we refer to the policy with updates as the *adaptive policy* and to the policy without updates as the *non-adaptive policy*.

4.2 Single Flow Experimental Jitter Measurements

In this section we study and analyze the jitter distribution to get clear picture of its characteristics under a real-world experiments.

For the measurements each sensor samples the temperature sensor periodically. Unless otherwise specified, the generation period was varied, ranging from 1 to 30 to 60 seconds. Furthermore, for each of the 16 channels a separate set of measurements has been carried out. During one experiment, each source transmits 5000 packets to the sink via a set of forwarders on its respective path. MAC-layer acknowledgments are enabled and a maximum of two MAC-layer retransmissions are carried out. Each forwarder records the timestamps of the received packets, source and destination addresses, flow identity (ID) , and

other parameters. The packet size is set to 80 bytes (not including packet overheads). The number of sources is set to 10 and we allow for each source and forwarder to have up to 5 neighbours and communicate on all of the available channels (16 channels). The minimum, average and maximum number of hops from any source to the sink node was, 3, 5.6 and 8, respectively.

4.3 Setup Under Consideration

In this section we describe our experimental setup and performance metrics.
4.2.

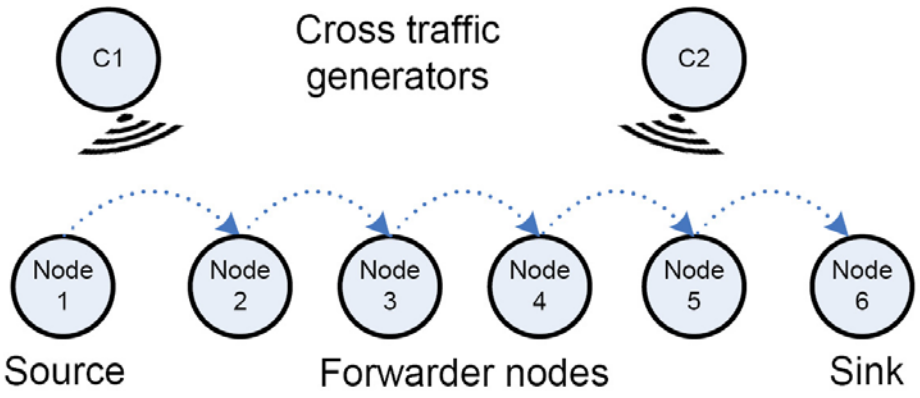


Figure 4.2: Experimental setup

4.3.1 Experiment setup

We consider our system model as a chain of one source, four forwarders and a sink node placed in our lab. The source node (node1) generates ad sends periodic traffic to the sink node through four forwarder nodes as shown in Figure 4.2. The sink node is a laptop used to record the packets for offline evaluation. Packets are unicast along the chain. The sensor nodes are located close to each other to prevent packet losses from channel errors, allowing us to focus on sleeping-induced losses. There is sufficient space in the packet so that each node can append its local reception timestamp to the packet. Our evaluations use these local timestamps.

In our experiments we used Tmote sky nodes and TinyOS version 2.0 operating system [73, 51]. We have set the radio channel of IEEE 802.15.4 compliant CC2420 transceiver [30] to a channel that from preliminary measurements has been found to be relatively interference-free.

For the purpose of our experiment we disabled the MAC layer acknowledgement and no retransmission was performed during the experiment. All

the other parameters of both MAC and Radio (such as Clear Channel Assessment (CCA), Back-off time interval,..etc.) stacks are not tuned but we use the default values provided by TinyOS. The nodes did not sleep during the experiment but unconditionally try to capture all packets. The impact of the sleep/wakeup scheduling policy was evaluated offline, based on the timestamps obtained from each node. Further, we introduce a cross traffic to our experiment by deploying two broadcast nodes C1 and C2 (see Figure4.2). These two node broadcast packets periodically. The generation packet period was varied ranging from 1 to 30 seconds. Specifically the cross traffic generator very the inter arrival period of the generated cross traffic randomly by 1 second step .

4.3.2 Performance metric

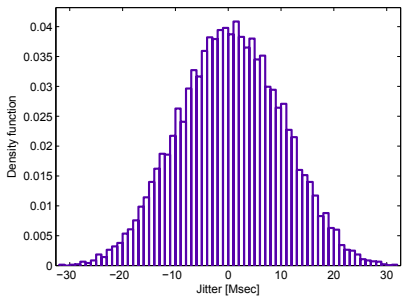
The prime objective of this evaluation is to study the relationship between the energy consumption of a node and the packet loss rate that results from either waking up too late or going back to sleep too early. We have decided to use the duty cycle of a forwarder node as a measure for its energy consumption. In our opinion this is a reasonable approximation, since for the CC2420 transceiver that we have used for this study, the power consumption in the transmit, receive and idle (waiting to receive something) states are very similar [182, 145, 35]. The second important performance measure under consideration is the loss rate observed by an individual node when operated under a sleep/wakeup schedule.

4.4 Discussion and Result for Jitter Measurements

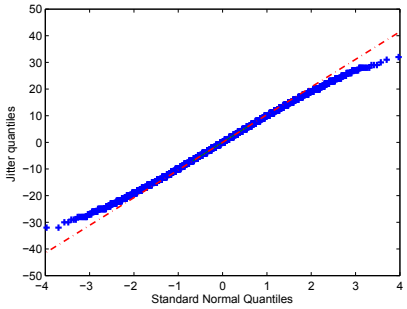
We conducted several experimental runs, measuring the jitter varying for example the number of hops in the multi-hop network and the packet generation period. As a matter of fact, in general the jitter distribution depends on a multitude of factors: the MAC and link-layer protocol, operating system imperfections (i.e. interrupts handling), the local load situation, and the position of a particular forwarder in the forwarder chain. We do not investigate this relationship in general, but look at one particular measurement setup and set of protocols.

For these specific choices we observe that the jitter distribution in multi-hop network as shown in Figures 4.3(a), 4.3(c), and 4.3(e) can be well modeled by a normal distribution as shown in the quantile-quantile plots in Figures 4.3(b), 4.3(d), and 4.3(f).

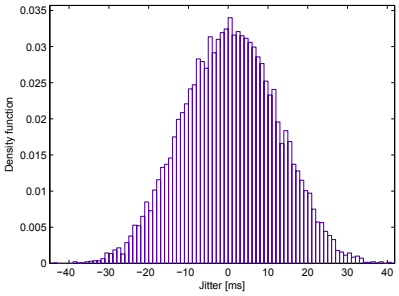
In addition, Figure 4.4 summarizes the jitter histogram and it is probability density function for multi-hop network with their parameter values.



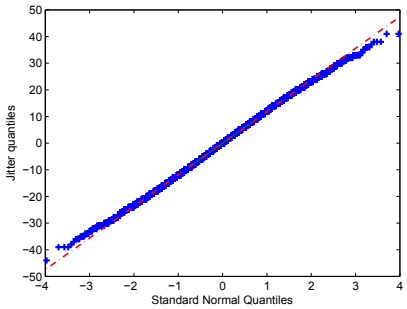
(a) Jitter histogram for 2nd hop



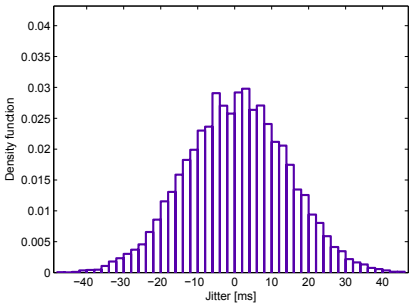
(b) QQ-plot for 2nd hop



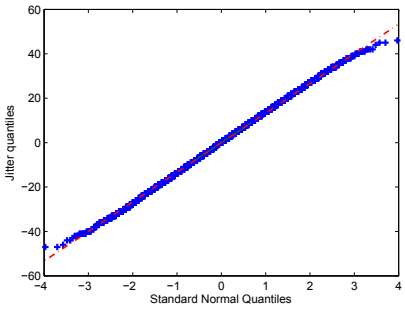
(c) Jitter histogram for 3rd hop



(d) QQ-plot for 3rd hop



(e) Jitter histogram for 4th hop



(f) QQ-plot 4th for hop

Figure 4.3: Jitter histogram and qq-plot

The results obtained from the above experiments are really motivating and it might be exploited to model and propose some solutions for several problems in wireless sensor networks. Moreover, this result allows to reduce the problem of quantile estimation (which is much harder and more memory-intensive than the estimation of averages [106, Sec. 9.5]) to the problem of estimating the variance of a normal distribution.

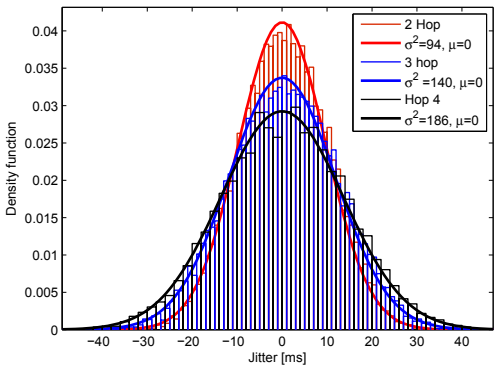


Figure 4.4: A Jitter histogram for multi-hop network

4.5 Estimation

Based on the previous gathered results of the jitter distributions we choose to use recursive estimators in which the current estimate depends on the previous estimate and the current measurement.

4.5.1 Traffic period estimator

Given k measurements of packet inter-arriving time X_i the sample mean is

$$\widehat{X}_k = \frac{1}{k}\sum_{i=1}^{k} X_i \tag{4.1}$$

Suppose that $\widehat{X}_k$ has been computed based on measurements X_i for i=1,....,k. Now one more measurement X_{k+1} is made. The new sample mean is computed as :

$$\widehat{X}_{k+1} = \frac{1}{k+1}\sum_{i=1}^{k+1} X_i \tag{4.2}$$

$\widehat{X}_{k+1}$ can be computed in terms of $\widehat{X}_k$ and X_{k+1} by proceeding as follows.

$$\widehat{X}_{k+1} = \frac{k}{k+1}\left(\frac{1}{k}\sum_{i=1}^{k} X_i\right) + \frac{1}{k+1}X_{k+1}$$

$$= \frac{k}{k+1}\widehat{X}_k + \frac{1}{k+1}X_{k+1}$$

So that the estimator for the mean is:

$$\widehat{X}_{k+1} = \widehat{X}_k + \frac{1}{k+1}\left(X_{k+1} - \widehat{X}_k\right) \tag{4.3}$$

The random variable (packet inter-arrival time) is denoted by $x(k)$ and its estimated value of the mean is denoted by $\widehat{x}(k)$.

4.5.2 Traffic jitter estimator

A recursive method can also be used to compute an estimate for the variance as following:

$$\sigma^2_{k+1} = \sigma^2_k + \frac{1}{k+1}\left[\frac{k}{k+1}(X_{k+1} - \widehat{X}_k)^2 - \sigma^2_k\right] \tag{4.4}$$

4.6 Performance Evaluation

We performed several experimental runs to evaluate our proposed schemes. The same experiment configuration and setup as explained in section 4.3 is utilized. We use the following parameters to evaluate the proposed schemes:

1. The first two parameters are mean and variance. These two parameters are combined to determine an appropriate sleep and wakeup time windows depending on a given performance requirements.

2. Stopping rule, which decides whether to continue or stop the acquisition state based on a prescribed number of packets.

3. Maximum allowable packets loss for each individual node, which tradeoffs between energy usage and packet loss rate.

4.7 Simulation Setup

In order to evaluate our approach, we conduct a trace-based simulation study based on real data collected from sensor nodes measurements. The evaluation consists of using real traces as an input to the simulation. These traces were taken from each forwarder node and contain the packets arrival times as well as the sequence number of the packets. Our approaches and algorithms are implemented and evaluated through Matlab version 7.1.

Figure 4.5 and Figure 4.6 illustrate a simplified simulation model for both non-adaptive and adaptive scheduling, respectively. For both, the acquisition

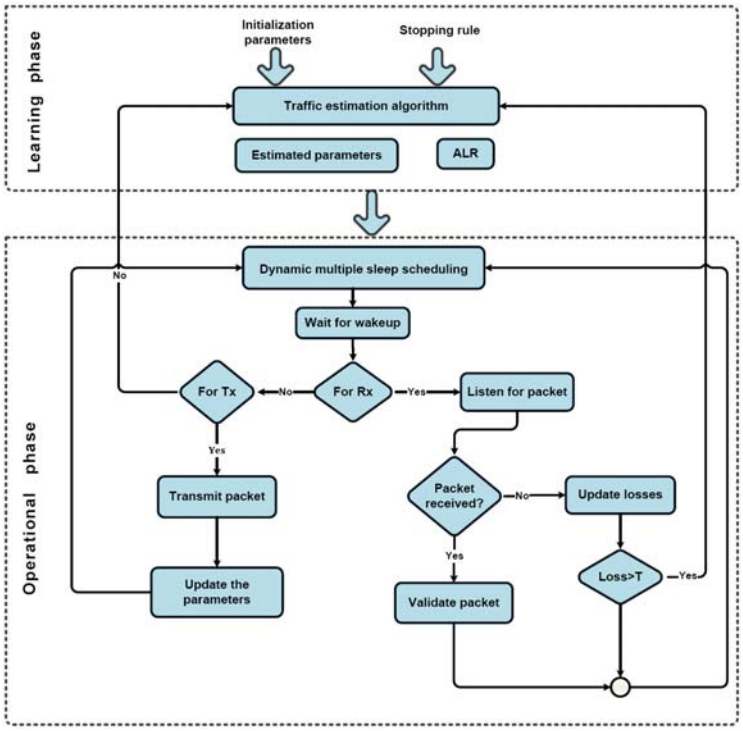


Figure 4.5: Single channel simulation model:non-adaptive policy

state starts by initializing the mean and the variance to zeros. The initialization of model parameters are applied to each hop independently and each hop estimates the values of the parameters model. The end of this state is determined by a stopping rule (must be explicitly stated) as described in section 4.1.2. After the acquisition state is converged and before entering the operational state a value of allowable loss rate should be specified based on the application requirements. In the operational state the sleep and wakeup times scheduling is applied by selecting an appropriate windows for sleep and wakeup times as discussed in section 4.1.2. During this phase each node locally monitors its packet loss and reacts based on the specified policy. In the first scenario non-adaptive policy as shown in Figure 4.5, the estimator algorithms run only once and no adaptation of the parameters model are performed, regardless of the packet loss rate. This policy provides a basis for evaluating the advantage of the other policy that does adapt. In the other hand if the policy is set to "adaptive" as shown Figure 4.6 then, the second scenario is applied, in which each forwarder continuously updates the estimate values and adapts its sleep and wakeup windows accordingly, without reentering the acquisition state again.

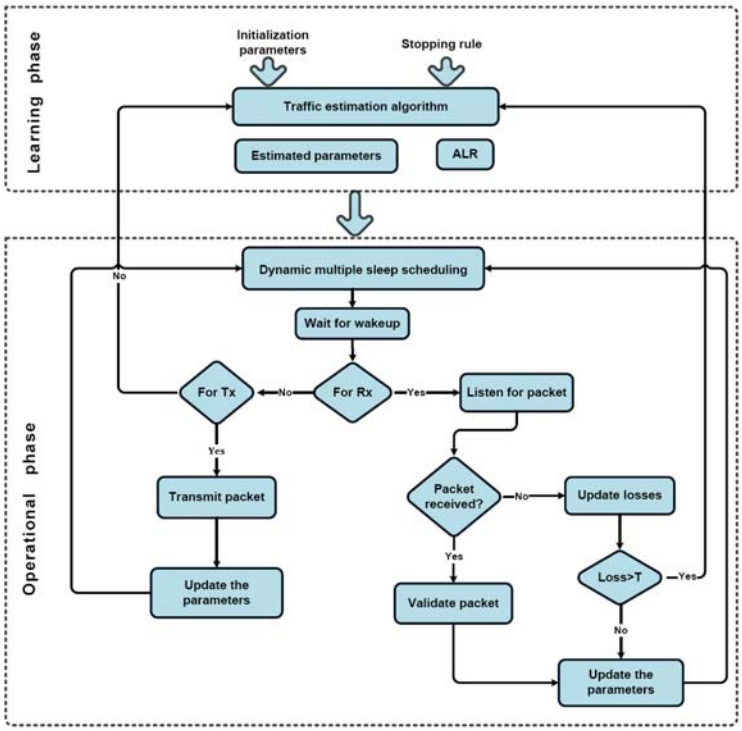


Figure 4.6: Single channel simulation model: adaptive policy

4.8 Results

In this section we present the simulation results of the proposed algorithms. We present our simulation result into the following scenarios: scheduling based on real parameters values, Non-adaptive scheduling, and adaptive scheduling.

4.8.1 Scheduling based on real parameter values

We start by examining the effect of using the real values of model parameters. The observed sample size of 10000 packets for each hop level were taken independently from a real measurement with expected value of 1024ms for each hop and variance values of 31ms, 46ms, 62ms for hop 2 , hop 3 and hop 4, respectively as shown in table 4.1.

Table 4.1 shows the result of wakeup times based on requested allowable loss rate of 2% and 5%, respectively using the real values. The result achieved in this scenario could be used to evaluate the reliability of the following scenarios.

Hop	**Sample**	**Real mean**$[ms]$	**Real variance**$[ms]$
2	9000	1024	31
3	9000	1024	46
4	9000	1024	62

Table 4.1: Full statistics without consideration of sleeping activities

4.8.2 Non-adaptive scheduling

Table 4.2 and Table 4.3 summarize the results of the non-adaptive scheduling scenario for 2% and 5% allowable loss rates, respectively. The first column lists the number of hop being examined, the second column shows the stopping rule based on prescribed number of packets and the last two columns in the acquisition state show the estimate of mean and the variance parameters. In the operational state the $L_{\max}$ column specifics the allowable loss rate, the subsequence columns lists the number of packets being transmitted for each hop level, the frequency of packet losses, and the packets loss rate, respectively.

Acquistion state				**Operational state**				**Fraction of**	
Hop	S-rule	$\widehat{\mu}[ms]$	$\widehat{\sigma}[ms]$	$L_{\max}$	Sample	Loss	% loss	Wakeup	Sleep
2	10	1024.2	24.524	2%	9000	310	3.44	0.0225	0.9775
2	20	1024.1	30.428	2%	9000	176	1.96	0.025063	0.9749
2	30	1024	29.045	2%	9000	180	2.00	0.024487	0.9755
2	50	1024.1	30.522	2%	9000	177	1.97	0.025102	0.9749
2	70	1024.1	30.832	2%	9000	139	1.54	0.025229	0.9748
2	100	1024.1	31.476	2%	9000	138	1.53	0.025491	0.9745
3	10	1024.3	29.501	2%	8690	396	4.56	0.024678	0.9713
3	20	1024	44.582	2%	8824	110	1.25	0.030337	0.9697
3	30	1023.9	44.562	2%	8820	110	1.25	0.03033	0.9697
3	50	1024	44.586	2%	8823	110	1.25	0.030339	0.9697
3	70	1024.1	45.725	2%	8861	120	1.35	0.030724	0.9693
3	100	1024.1	46.733	2%	8862	120	1.35	0.031061	0.9689
4	10	1024.1	45.023	2%	8294	176	2.12	0.030487	0.9692
4	20	1024	46.154	2%	8714	318	3.65	0.030867	0.9691
4	30	1023.9	50.597	2%	8710	203	2.33	0.032319	0.9677
4	50	1024	55.467	2%	8713	121	1.39	0.033839	0.9662
4	70	1024	54.951	2%	8741	122	1.40	0.033681	0.9663
4	100	1024	55.921	2%	8742	130	1.49	0.033977	0.966

Table 4.2: Non-Adaptive scheduling for 2% allowable loss rate

From the below tables we can observe the following: (a) from one hop to the next hop the fraction of wakeup times do not vary too much. (b) The first hop

has the highest fraction of sleeping times and (c) within every hop level you can see the trade off between acquisition state and the energy consumption. It is also clear that the fraction of sleep times is increasing tell 30 packets in the stopping rule then it goes slightly down and it behaves almost the same for all the hops.

Acquistion state				**Operational state**				**Fraction of**	
Hop	Stopping rule	$\widehat{\mu}[ms]$	$\widehat{\sigma}[ms]$	$L_{\max}$	Sample	Loss	% loss	Wakeup	Sleep
2	10	1024.1	24.524	5%	9000	638	7.09	0.019344	0.9807
2	20	1024.1	30.428	5%	9000	400	4.44	0.021547	0.9785
2	30	1024	29.045	5%	9000	503	5.59	0.021052	0.9789
2	50	1024.1	30.522	5%	9000	401	4.46	0.021581	0.9784
2	70	1024.1	30.832	5%	9000	406	4.51	0.02169	0.9783
2	100	1024.1	31.476	5%	9000	307	3.41	0.021915	0.9781
3	10	1024	29.501	5%	8362	596	7.13	0.021217	0.9788
3	20	1024	44.582	5%	8600	218	2.53	0.026082	0.9739
3	30	1023.9	44.562	5%	8497	198	2.33	0.026076	0.9739
3	50	1024	44.586	5%	8599	218	2.54	0.026083	0.9739
3	70	1024.1	45.725	5%	8594	218	2.54	0.026414	0.9736
3	100	1024.1	46.733	5%	8693	243	2.80	0.026704	0.9733
4	10	1023.9	45.023	5%	7766	261	3.36	0.026211	0.9738
4	20	1024	46.154	5%	8382	456	5.44	0.026538	0.9735
4	30	1023.9	50.597	5%	8299	302	3.64	0.027786	0.9722
4	50	1024	55.467	5%	8381	318	3.79	0.029092	0.9709
4	70	1024	54.951	5%	8376	319	3.81	0.028957	0.971
4	100	1024	55.921	5%	8450	333	3.94	0.029211	0.9708

Table 4.3: Non-Adaptive scheduling for 5% allowable loss rate

4.8.3 Adaptive scheduling

Table 4.4 and Table 4.5 list the results obtained by continually updating the parameters estimate for 2% and 5% allowable loss rate, respectively. The first column lists the hop level being examined, the second column shows the stopping rule based on prescribed number of packets and the last two columns in the acquisition state show the estimate of mean and the variance parameters. In the operational state the $L_{\max}$ column specifics the allowable loss rate, the subsequence columns lists the number of packets being transmitted for each hop, the frequency of packet losses, and the packets loss rate, respectively.

We cal also observe from the adaptive scheduling results that target packet loss rate has been improved compared to the non-adaptive policy and has never

Acquistion state				**Operational state**				**Fraction of**	
Hop	S-rule	$\widehat{\mu}[ms]$	$\widehat{\sigma}[ms]$	$L_{\max}$	Sample	Loss	% loss	Wakeup	Sleep
2	10	1024.1	24.524	2%	9000	141	1.57	0.025291	0.9747
2	20	1024.1	30.428	2%	9000	138	1.53	0.025291	0.9747
2	30	1024	29.045	2%	9000	141	1.57	0.025291	0.9747
2	50	1024.1	30.522	2%	9000	139	1.54	0.025291	0.9747
2	70	1024.1	30.832	2%	9000	141	1.57	0.025291	0.9747
2	100	1024.1	31.476	2%	9000	140	1.56	0.025291	0.9747
3	10	1024	29.501	2%	8859	98	1.11	0.030875	0.9691
3	20	1024	44.582	2%	8862	96	1.08	0.030875	0.9691
3	30	1023.9	44.562	2%	8859	96	1.08	0.030875	0.9691
3	50	1024	44.586	2%	8861	96	1.08	0.030875	0.9691
3	70	1024.1	45.725	2%	8859	96	1.08	0.030875	0.9691
3	100	1024.1	46.733	2%	8860	96	1.08	0.030875	0.9691
4	10	1023.9	45.023	2%	8761	93	1.06	0.035749	0.9643
4	20	1024	46.154	2%	8766	92	1.05	0.035749	0.9643
4	50	1023.9	50.597	2%	8763	90	1.03	0.035749	0.9643
4	30	1024	55.467	2%	8765	88	1.00	0.035749	0.9643
4	70	1024	54.951	2%	8763	89	1.02	0.035749	0.9643
4	100	1024	55.921	2%	8764	90	1.03	0.035749	0.9643

Table 4.4: Adaptive scheduling for 2% allowable loss rate

exceed the requested allowable loss rate however, this at the cost of wakeup times.

4.9 Multi-Channel Traffic Jitter Measurement

What we have described so far does not consider frequency hopping, which we explain in detail in Chapter 5. For the time being, it suffices to mention that a forwarder in the learning state initially starts listening on a randomly chosen channel. Having received the first packet on this channel, the forwarder knows on which channel the next packet from the same source will be transmitted, as the source switches channel for each new packet and the channel hopping sequence is well known to all nodes.

In the following we report the results of jitter measurements carried out with the standard protocol stack coming with the TinyOS 2.0 operating system [52]. We provide evidence that for this protocol stack the jitter distribution at various hop distances is well modeled by a normal distribution. With this

Acquistion state				**Operational state**				**Fraction of**	
Hop	Stopping rule	$\widehat{\mu}[ms]$	$\widehat{\sigma}[ms]$	$L_{\max}$	Sample	Loss	% loss	Wakeup	Sleep
2	10	1024.1	24.524	5%	9000	328	3.64	0.021743	0.9783
2	20	1024.1	30.428	5%	9000	317	3.52	0.021743	0.9783
2	30	1024	29.045	5%	9000	322	3.58	0.021743	0.9783
2	50	1024.1	30.522	5%	9000	318	3.53	0.021743	0.9783
2	70	1024.1	30.832	5%	9000	323	3.59	0.021743	0.9783
2	100	1024.1	31.476	5%	9000	319	3.54	0.021743	0.9783
3	10	1024	29.501	5%	8672	237	2.73	0.026544	0.9735
3	20	1024	44.582	5%	8683	234	2.69	0.026544	0.9735
3	30	1023.9	44.562	5%	8678	234	2.70	0.026544	0.9735
3	50	1024	44.586	5%	8682	234	2.70	0.026544	0.9735
3	70	1024.1	45.725	5%	8677	234	2.70	0.026544	0.9735
3	100	1024.1	46.733	5%	8681	234	2.70	0.026544	0.9735
4	10	1023.9	45.023	5%	8435	218	2.58	0.030734	0.9693
4	20	1024	46.154	5%	8449	218	2.58	0.030734	0.9693
4	50	1023.9	50.597	5%	8444	217	2.57	0.030734	0.9693
4	30	1024	55.467	5%	8448	214	2.53	0.030734	0.9693
4	70	1024	54.951	5%	8443	215	2.55	0.030734	0.9693
4	100	1024	55.921	5%	8447	216	2.56	0.030734	0.9693

Table 4.5: Adaptive scheduling for 5% allowable loss rate

insight the estimation process is greatly simplified: a forwarder only needs to measure the average and standard deviation of the packet interarrival times in order to obtain the full distribution. From this distribution then the desired upper and lower $\alpha/2$ quantiles are easily derived.

4.9.1 Experimental Setup

We carry out our experiments on the TWIST testbed (TKN Wireless Sensor Networks Testbed) [62]. A detailed description of the TWIST testbed is given in Section 3.3.1.

We use the TinyOS version 2.0 operating system [73, 51] and its default protocol stack, which uses a CSMA-type MAC protocol. We also use the well known CTP routing protocol [55] to construct routes between the sources and the sink. For the measurements each sensor samples the temperature sensor periodically. The generation period was varied, ranging from 1 to 30 to 60 seconds. Furthermore, for each of the 16 channels a separate set of measurements has been carried out. During one experiment, each source transmits 5000 packets to the sink via a set of forwarders on its respective path. MAC-layer

acknowledgments are enabled and a maximum of two MAC-layer retransmissions are carried out. Each forwarder records the timestamps of the received packets, source and destination addresses, flow ID, packet sequence, Received Signal Strength Indicator (RSSI), Link Quality Indicator (LQI), and the frequency channel.

The packet size is set to 80 bytes (not including packet overheads). The number of sources is set to 10 and we allow for each source and forwarder to have up to 5 neighbours and communicate on all of the available channels (16 channels). The minimum, average and maximum number of hops from any source to the sink node was, 3, 5.6 and 8, respectively.

4.9.2 Discussion and Result for Multi-channel Jitter Measurements

We conducted several experimental runs, measuring the jitter under various scenarios, including different channels, random topology, multiple flows, cross traffics, varying the number of hops and the packet generation period.

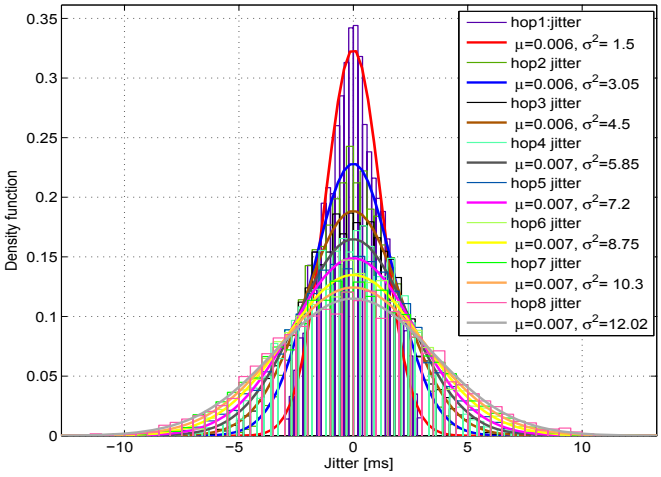


Figure 4.7: Multi-hop jitter for channel 11.

For all the scenarios covered in our measurements we find that the per-flow jitter distribution is well modelled by a normal distribution. For brevity we only show the histograms and the quantile-quantile plots for channel 11 as shown in Figures 4.7 and 4.8, respectively. Please note that this finding was also confirmed in our previous paper [92], which was limited to a single source scenario with a common single channel. However, here we have extended our measurements to multiple channels solution and different periods. Please note

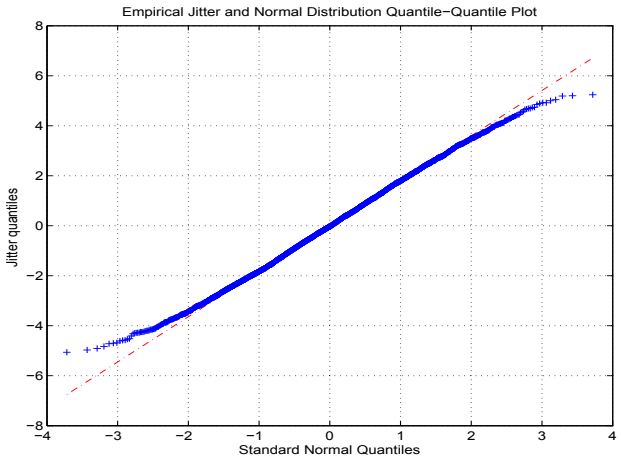


Figure 4.8: Quantile of empirical jitter against quantile of normal distribution for channels 11.

that similar trends are observed also for scenarios with 30 and 60 seconds traffic generation period and for all other channels.

4.10 Summary

In this chapter we test our hypothesis in small setup scenario and proposed a new scheme of extending the sleep times of wireless sensor nodes online and in decentralized way to save energy usage and consequently prolonging the network lifetimes. This approach is suitable for a number of applications, i.e. data collection and monitoring applications in which nodes have to send their data periodically to sink node thought a number of intermediate nodes. In our approach, the forwarder nodes compute a local estimate of both the traffic period and the jitter. The extra work of period estimation that forwarders do could save the source node from explicitly signaling the period to forwarders, and furthermore it saves the whole network from maintaining a common time base through a time synchronization protocol that lets all forwarders interpret the signaled period in a consistent way. The novelty of our approach is the adaptation of sleep periods to the estimated traffic period and jitter subject to prescribed packet loss requirements. Moreover, we adjoin mechanisms to the proposed scheme that allow to update the period and jitter estimates and to react to packet losses. We also studied the jitter distribution by conducting several experimental runs, measuring the jitter varying for example the number of hops in the multi-hop network and the packet generation period under cross traffic scenario. Then we hit upon a theoretical distribution that fit the random

variables of the jitter and exploited the characteristic of the jitter distribution to design and implement estimation algorithms. We used theoretical analysis, real raw measurements and simulation experiments to evaluate our proposed algorithms. Our proposed solution showed that by adapting the sleep cycles into the observation loss rate would indeed result in a significant energy saving and thus achieving a long network lifetimes. Furthermore, we demonstrated the results of the trade-off between energy consumption and packet loss rate. Moreover, we conducted a real world experiment to measure the jitter across different channels using the TWIST testbed.

In the next chapter we extend our autonomous framework to multi-flows and introduce a novel mechanism to support high reliability and low packet delay .

CHAPTER 5

Multi-Channel Autonomous Framework Design and Evaluation

In this chapter we extend our autonomous framework introduced in the previous chapter to support not only energy efficiency but also communication reliability under general scenarios. For this we present a scheme which supports multiple periodic traffic flows and frequency hopping without requiring an expensive protocol infrastructure providing synchronization features (time synchronization, hopping synchronization) by relying entirely on the periodicity of the traffic itself for synchronization purposes. Our approach is extremely light in terms of signaling, only ACK packets need to carry few bits of information. We believe that our autonomous framework is an attractive alternative to WirelessHART and similar systems in lightly loaded networks with periodic traffic. We also present design and performance evaluation of our autonomous framework.

5.1 Asynchronous Channel Hopping

Our asynchronous channel hopping approach is not synchronized to any external time reference. Instead, channel hopping is synchronized to the period with which a source node sends its data packets. Our approach distinguishes itself from existing channel hopping protocols, such as WirelessHART [39] and ISA100.11a [83], by scheduling the whole network activities in a distributed manner and without maintaining an explicit time synchronization protocol, thus reducing the signaling load and saving overall system complexity. We first explain our approach for a single flow and then extend to the case with

multiple flows in a network.

There are two main approaches to channel hopping: (synchronized) blind channel hopping and adaptive channel hopping. Blind channel hopping (as used in WirelessHART) might use all 16 channels independent of their current quality and hops on a per-time-slot basis (which in WirelessHART amounts to a per-packet basis).

In contrast, adaptive hopping aims to use a subset of the best channels (white-listing). Adaptive hopping is more complex to implement, as it first requires a mechanism to frequently scan and rank all the channels for their quality for each link. Second, each node has to keep statistics of channel qualities for each link. Third, each pair of nodes needs to achieve consistent rankings of the individual channels, otherwise they will may end up communicating using different channels. This requires additional signaling. For these reasons we base our asynchronous blind channel hopping (ABCH) on the blind hopping approach.

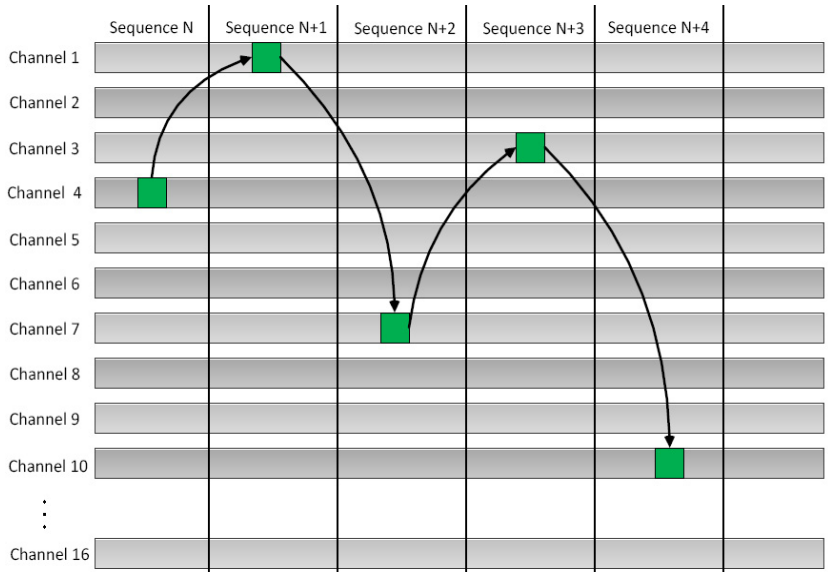


Figure 5.1: Asynchronous blind channel hopping

ABCH exploits the characteristics of a single periodic traffic flow and estimates the next channel to be used based on the sequence number of the packet. Each source node starts hopping blindly on a per-packet basis, using all available channels (see Figure 5.1). The source includes sequence numbers into its packets, and the next channel to use depends on the sequence number as follows:

$$NextChannel = (SQ + chOffset) \mod chNum \tag{5.1}$$

where SQ is the next sequence number, $chOffset$ is the channel offset and $chNum$ is the number of channels being used. In the learning phase, each forwarder starts by listening for a packet on some random channel. Upon receiving the first packet on this channel, the forwarder retrieves the sequence number and determines the next channel according to Equation 5.1. As an example, if $chOffset = 1$, $chNum = 16$ and a forwarder received packet with $SQ = 8$, then the current channel index is 9 and the next one is 10 (Note that here the channels are numbered from 0 to 15 instead of 11 to 16, but translation is straightforward). Please note that the result of this equation is the index of the channel which just a lookup table for the corresponding IEEE 802.15.4 channels. So the offset index number does not necessarily corresponds to the same physical channel number.

Please also note that each forwarder applies the ABCH mechanism after receiving the first packet – specifically, a forwarder also uses the determined channels for its own transmissions of the packet. In the next section we examine the synchronization of two neighbors in the presence of transmission errors.

5.1.1 Handling Transmission Errors

Figure 5.2 shows the interaction between a pair of nodes for exchanging packets. We assume that the nodes have learned the flow period and are ready to communicate. Figure 5.2) illustrates three sequences, the first sequence shows a simple error-free transmission. In this sequence a sender transmits packet p_1 on channel 11 and waits for an ACK for a predefined time-out on the same channel. Upon reception of the packet the receiver sends an ACK back to the sender indicating the next expected sequence number to be received and performs a statistics update. If the transmitter receives the ACK, it also performs a statistics update and removes that packet from its buffer, otherwise a copy of the transmitted packet is kept in the buffer.

The second sequence illustrates the interaction in case of a data packet loss. When the receiver wakes up on channel 12 to receive a packet, it waits for its wakeup window and remains awake until either a packet is received or until the upper $\alpha/2$ quantile has passed, as explained in Section 4.1. In this example the receiver does not get a packet and assumes that the packet is lost and updates its statistics. However, it computes the next channel frequency as if it received packet p_2 (the lost packet). This is important as we will explain in the third sequence (ACK loss). Similar actions are taken at the sender side. Once the ACK time-out is triggered, the sender assumes that packet p_2 or its associated ACK is lost. However, it computes the next channel as if it received a successful ACK for packet p_2 and then updates its statistics. In the next wakeup-window it transmits packet p_3 on channel 13. Upon receiving p_3, the receiver node sends back an ACK indicating the next expected packet

to receive. In this case the receiver returns the sequence number of packet p_2 and it stays awake[1] to receive packet p_2 on the same channel (channel 13). The sender then retransmits packet p_2 on channel 13. Please note that, the recovery process of the lost packet p_2 immediately follows the previous successful transmission of p_3, leveraging the good conditions on the current channel.

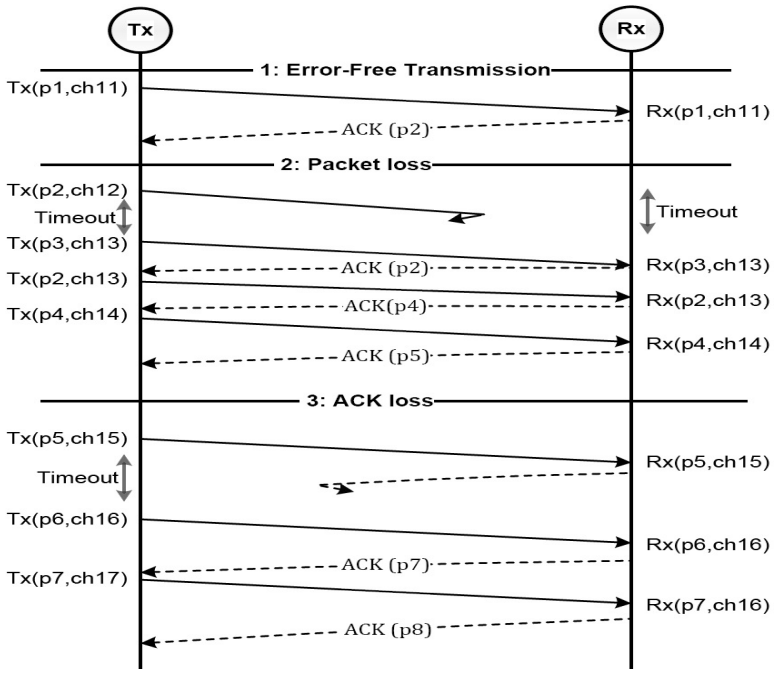


Figure 5.2: Autonomous channel hopping transition diagram

The third sequence shows the packet exchanges in case of ACK packet loss. When the sender transmits p_5 on channel 15 and its timer expires before the packet is acknowledged, it assumes that either the data packet or the ACK packet is lost. In either case, it computes the next channel as if it got a successful ACK for p_5. It also updates its statistics and goes to sleep. In the next wakeup window, the sender transmits p_6 on channel 16 and waits for an ACK. If it receives a successfully ACK then it knows that the previous packet (p_5) was correctly received, because the receiver indicates in its ACK that the next expected sequence number is that of p_7, otherwise the ACK would have included the sequence number of the lost packet.

[1]Please note, that in this circumstance, the forwarder increase its wakeup-window temporary to accounts for the retransmission. Upon receiving the missing packet, the forwarder may go to sleep (depends on how much time left for the next activity).

5.1.2 Estimation of Multi-Flow Traffic

Direct estimation of quantiles is non-trivial and relatively memory-intensive [106, Sec. 9.5] as compared to the estimation of simple averages. In order to handle this issue we apply parametric approach, the class of distribution functions for the jitter distribution is known a priori (for example from measurements) and the task reduces to the problem of estimating the actual parameters of the distribution and the subsequent computation of adjusting proper window for sleeping activities. It turns out that even for the parametric case considered in this thesis nothing more than a variance estimate for the jitter is required (Based on our measurements, see Section 4.4);

Since in our measurements, we want to compute the current estimate from the previous estimate and the current measurement, we use recursive estimators for the traffic period and the jitter variance. The mean period for any independent flow f can be computed as:

$$\widehat{X}_{f,k+1} = \widehat{X}_{f,k} + \frac{1}{k+1}\left(X_{f,k+1} - \widehat{X}_{f,k}\right) \tag{5.2}$$

where $\widehat{X}_k$ refers to the estimated mean packet inter-arrival time after the k-th packet and X_k is the k-th actual interarrival time – interarrival times are only obtained for two successively received packets. For any independent flow f, a recursive method can be found to compute for the jitter variance as:

$$\widehat{\sigma}^2_{f,k+1} = \sigma^2_{f,k} + \frac{1}{k+1}\left[\frac{k}{k+1}(X_{f,k+1} - \widehat{X}_{f,k})^2 - \sigma^2_{f,k}\right] \tag{5.3}$$

where σ^2_k denotes the estimated jitter variance after observing the k-th interarrival time per flow f.

5.2 Local Dynamic Sleep State Scheduling

In this section we analyze how an improved usage of the transceiver sleep states can substantially reduce the overall energy-consumption, thereby increasing the autonomous system energy-efficiency.

Modern radios have built-in support for several sleeping states of operation with each state consuming a different amount of power. The radio also requires some time to switch into and out of different sleep states. For example, the CC2420 has three sleeping states: the idle-sleep-state, the power-down-state, and voltage-regulator-off-state, hereafter referred to as sleep-mode-1, sleep-mode-2, and sleep-mode-3, respectively. These sleep modes and their possible transitions are illustrated in Figure 5.3. In sleep-mode-1, both the voltage regulator and the crystal oscillator are enabled. The energy-saving in sleep-

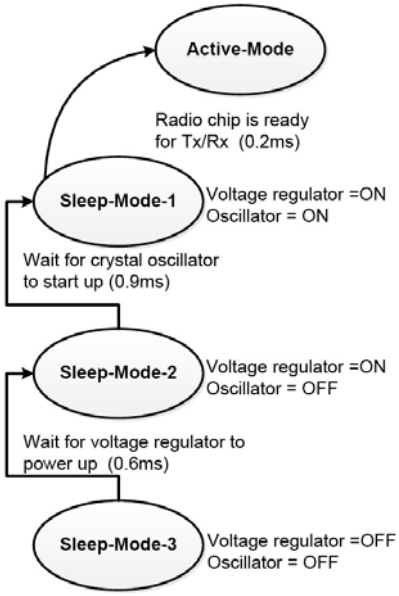


Figure 5.3: Sleep transition states for CC2420 Radio.

mode-1 state is obtained by disabling the radio frequency synthesizer which controls the channel selection and up/down RF conversion. Sleep-mode-1 has the fastest transition time of around 0.192 ms and consumes 1.4 mW of power, which is the highest among the sleep modes. In sleep-mode-2, the voltage regulator is enabled and the crystal oscillator is disabled. This mode consumes 0.07 mW of power. In sleep-mode-3, both the voltage regulator and the crystal oscillator are disabled. This mode has the slowest transition time and lowest power-consumption ($6.6 \cdot 10^{-5}$ mW). In general this mode switches off the radio chip completely, including radio RAM. As a result any packet waiting in the receiving or transmitting buffer is lost. Despite the fact that most (if not all) energy-efficient WSNs MAC protocols use the popular CC2420 Radio chip or similar Radio chip that support multiple sleep modes, they only use one single-fixed sleep mode. Moreover, they usually control and use the lightest sleep mode (sleep-mode-1) which calls in CC2420 data-sheet "idle" mode. For example B-MAC [136] and X-MAC [20] uses sleep-mode-1. According to X-MAC paper: (when X-MAC "sleeps" the radio, in fact it puts the radio into idle mode, as sleep mode turns off the oscillator and requires a longer time to transition back to receive mode [20]. To the best of our knowledge there is no a MAC protocol in WSNs that utilizes the multiple radio sleep states and thus provide a dynamic assignment of multiple sleep modes. According to our previous study we showed that about 40% of the total energy-consumption is due to the sleeping activity thus, we believe that our approach elegantly

copes with the issues raised above, and it will become commonplace for the energy-saving mechanisms in WSNs MAC protocols under variant periodic traffic rates.

In the next section we propose a generic simple approach that may runs on top of any MAC layer protocols and in which each node can apply individually, based on its current traffic rate.

5.2.1 Dynamic Multiple Sleep States Scheduling (DM3S)

In what follows, we propose a practical and effective dynamic multiple sleep states scheduling scheme, abbreviated as DM3S. It exploits the multiple sleep states of the CC2420 radio and utilizes them based on the estimation of the next packet arrival. This approach is independent of the underlying link scheduling algorithm, but a node uses its given schedule to determine the right sleep states. For ease of presentation, we called the scheduled activity window (Tx or Rx interval time) as a time slot. Generally speaking, a nodes activities are constrained to certain slots whereas in all other slots they can sleep. We call the slots that a node might be involved in its **active slots**. There will generally be some active slots in which a node will have to wake up unconditionally, for example those slots in which the node is scheduled to receive, or those transmit slots where a packet is transmitted the first time. On the other hand, retransmission slots are only used when a transmission in a previous transmit slot has failed (i.e. the sender has not received an acknowledgement). A key observation is that at the end of a transmit slot the sender will know if it has to utilize a retransmission slot or not. More generally, based on its schedule and the transmission outcomes in the current active slot, at the end of the current slot a node can determine how much time will elapse before its next active slot starts.

The second key ingredient is borrowed from a technique used in dynamic power management to control the device's operational states, see [78, 15]. Specifically, since the number of transceiver states and their switching time is known a-priori, it is possible to construct a function $\phi(\cdot)$, which takes a non-negative time duration τ as a parameter and which returns a sleeping schedule that:

(i) ensures that after τ seconds the node transceiver is ready to transmit or receive, (ii) sends the transceiver through a "monotone" sequence of sleep states (the deepest state at the beginning and the lightest state at the end), and that (iii) ensures that the chosen sequence of states (and the times being spent in each visited state) has the smallest energy-consumption over the time horizon of τ seconds. For the CC2420 transceiver this function $\phi(\cdot)$ is straightforward to construct. Specifically, we need to determine three threshold values: (i) a duration τ_1 that is minimally needed to make sleep-state-1 more energy-

efficient than to stay awake; (ii) a duration $\tau_2 > \tau_1$ that is minimally needed to make an initial choice of sleep-state-2, followed by a transition through sleep-state-1 and subsequent wakeup more energy-efficient than to start initially with sleep-state-1; and (iii) a duration $\tau_3 > \tau_2$ that is minimally needed to make an initial choice of sleep-state-3, followed by a transition through sleep-state-2, sleep-state-1 and subsequent wakeup more energy-efficient than to initially start with sleep-state-2.

When at the end of an active slot it takes a time τ before the next active slot starts, it is a simple matter of comparing τ to the three thresholds τ_1, τ_2 and τ_3 to figure out which sleep state (if any) should be entered next.

5.3 On-the-fly Traffic Adaptation Mechanism

In many industrial applications, source nodes may increase or decrease their traffic periodicity upon the occurrence of new even of interest. For instance a source node may start with very light traffic load (sampling temperature every 1min) and when the source node detects an event of interest then it begins to sample its traffic at a higher sampling rate (for example: sample light sensor each 1sec).

Moreover, sensor, nodes may decide to change its traffic period to a lower rate, when the battery drops below a certain level. This might prevent an abrupt terminations of the whole network due to battery depletion. Therefore, deceasing the sampling rate in such situation not only increase the network lifetime but also allow enough time for maintaining the network (for instance replacing batteries), and maintain the fine granularity of the information gathered from such source nodes. Thus, the generation sampling rates of the sources can be vastly different over time. In such situation the forwarder nodes should respond and adapt to the new traffic requirements as fast as possible while consuming the minimum amount of energy during this phase.

In our autonomous framework we design and implement an efficient approach that allows a forwarder reacts fast to the traffic change. The key idea is that, source node exploit the currant traffic situation to notify its neighbor about the new traffic requirement (piggybacking on existing traffic) in decentralized manner. When a source node wants to change its traffic data rates, it just set the adaptive bit indicator to one. Otherwise the bit is set to zero. Upon the receiving of the data packet, the forwarder checks this bit and determines whether it has to enter new learning phase or not. Please note that the evaluation of this policy will be presented in Chapter 7.

5.4 Interference due to Multi-flow Overlapping

We finally consider the operation of our scheme in the multi-flow case, focusing on a forwarder through which two or more flows of possibly different period and from different sources pass. For such a forwarder it might happen that two upstream nodes want to send packets at nearly the same time but possibly on different frequencies – we refer to this as a collision (see Figure 5.4).

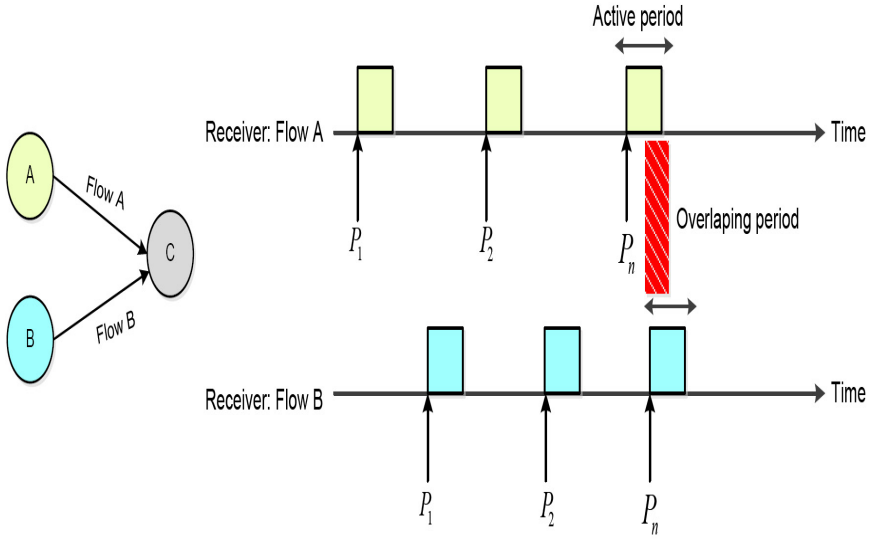


Figure 5.4: An example of overlapping time period

To deal with this, we propose to exploit the traffic estimation values to detect and resolve a potential collision beforehand. Specifically, after receiving a packet a forwarder checks all flows going through it whether there is a potential collision for the next packet. Figure 5.4 shows an example of an overlap period; when node C receives a packet from node A-Flow1 or node B-Flow 2, it first checks the next activity periods by comparing $t_a(n+1)$ of both flows (please refer to Section 4.1.1 for more details about the activity periods). If so, it notifies the upstream node (by setting a special flag in the ACK packet) to randomly back-off longer in the next transmission cycle (in this example it notifies node B). Also the forwarder readjusts its corresponding wakeup window temporarily. We assume that there is no overlap in the first cycle of the transmissions. This is a realistic assumption since each node usually starts with a random offset.

Pleas note that because each flow is associated with random channel hopping sequence (each flow start with different channel number; based on it is

Table 5.1: Main CC2420 and autonomous framework parameters

Parameter	Value
Data frame size	128
Acknowledgment frame size	12
Channel switching time	192 μs
Length of learning phase	5 packets
Allowable packet loss rate α	2
Loss threshold	3 packets

source flow ID) and the starting time of each transmission is set randomly, interference among neighbor multi-flow is rarely happened. A similar observation was made in [160, 82, 180]. This is one of the main advantages of channel hopping solution which increases the resilience to interference (compared to the single channel solution) by transmitting consecutive packets using different channels [169, 56],

5.5 Methodology and Setting

In order to evaluate the autonomous framework's mechanisms (asynchronous channel hopping, estimation and adaptation of multi-flows traffics and the local dynamic multiple sleep states scheduling). We use the same methodology discussed in Section 3.3 which combines both connectivity traces and simulation. Both are explained in details in Section 3.3.2 and Section 3.3.3, respectively. Table 3.1 summarizes the main power consumption parameters of a CC2420 transceiver and of a MSP430 micro-controller, assuming a 3.3V supply voltage.

The main parameters of the autonomous framework are listen Table 5.1.

5.5.1 Network Topology and Traffic

We have generated 150 random topologies and for each setting of simulation parameters we correspondingly perform 150 replications. For each random topology we have placed 45 nodes in an area of size 225×225 feet, using a uniform distribution for node positions. The sink is placed in the upper right corner of the nodes. Out of the 45 nodes we randomly pick five nodes as source nodes. Each of these sources periodically generates packets with a payload of 80 bytes (not including PHY and MAC overhead). Unless otherwise specified, all the sources transmit with the same period, however, the starting phase is set randomly. The generation period was varied, ranging from 1 to 30 to 60

seconds. During each simulation run, each source transmits packets based on its periodicity and then forwards these packets to the sink via some forwarders. MAC-layer acknowledgments are enabled and the size of the ACK packet is 12 bytes. If the packet is lost then the sender tries to transmit the packet for a maximum of two retries.

5.5.2 Major Performance Measure

The simulation time is fixed to 168 hours (one week) and the two main performance measure are the total energy spent by the radio transceiver of a node over this period, and the end-to-end Packet Delivery Ratio (PDR), i.e. the fraction of all packets sent by the sources that reach the destination. At one or two occasions we also use the end-to-end packet loss rate, which is just the complement of the packet delivery ratio, as a performance measure.

The simulation records the amount of time spent in various states (transmit, receive, listen, sleep and turnover) and calculates from this the total energy consumption of a node over a span of 168 simulated hours. We also take into consideration the energy consumed by the node's micro-controller. We split the micro-controller energy consumption in two main states, active state and sleep state. The micro-controller is active at the same time as the radio. At the end of each run, the simulation computes the total energy consumed for all nodes in the network using the amount of energy consumed by the radio and micro-controller in each state.

5.6 Results

In order to study the performance of our autonomous framework, we compare the asynchronous blind channel hopping with 16 channels against a system in which only one channel is used. We first investigate the packet delivery ratio and the total energy-consumption. Then we study the impact of the multi-flow overlap on energy-consumption and packet delivery ratio. We also investigate the impact of the length of the learning phase on the performance of the autonomous framework.

5.6.1 Packet Delivery Ratio

Figure 5.5, shows the average packet delivery ratio when a using single channel vs. using all 16 channels in case of 1sec data rate. The results are averaged over all runs. This graph confirms that our framework is able to reap the benefits of channel hopping, the single channel scenario has a lower packet reception rate that varies across the channels. This is due to the fact that

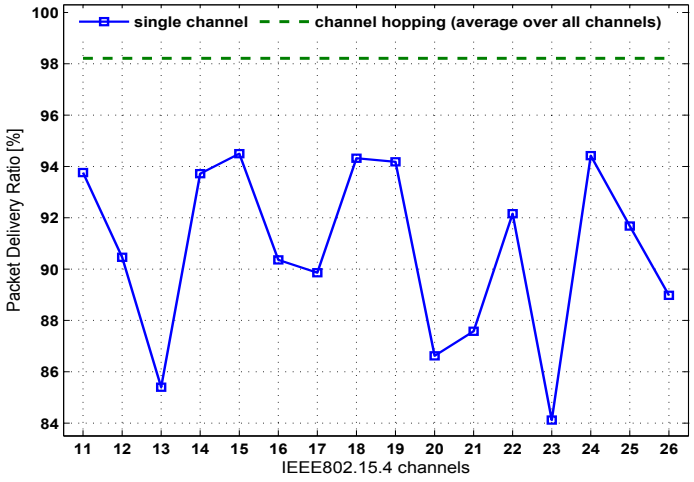


Figure 5.5: Average PRR: Single channel vs blind channel hopping

there is usually no single channel which is persistently reliable most of the time. On the other hand, the ABCH mechanism increases the reception rate because if the current channel is bad the next retransmission will be done on a different channel, thus increasing the probability of successful transmission. Similar trends are observed also for scenarios with 30s and 60 seconds traffic generation period (see our technical report [95]).

5.6.2 Energy Consumption

Figure 5.6 shows the average per-node energy-consumption for both the ABCH mechanism and the single channel solutions (for all channels), where the average is only taken among the nodes being on the path of any source flow. We can observe from the figure that the energy consumption of the single channel solution is much higher than with all 16 channels available. This is due to the higher number of retransmissions carried out on lossy channels.

Please note that, in order to have a fair comparison study between our asynchronous channel hopping and the single channel solution, we didn't apply the dynamic sleep state scheduling, but we rather used the same sleep-mode as in the single channel solution in this evaluation. This also allows us to gain a better understanding of the benefit of the channel hopping in team of energy consumption.

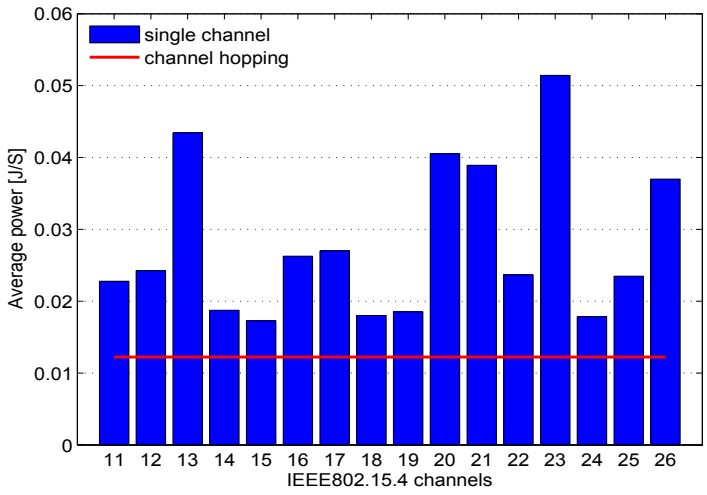


Figure 5.6: Average energy: Single channel vs blind channel hopping

5.6.3 Impact of the Multi-flow Overlap

We study the performance of the multi-flow overlapping mechanism in terms of both energy-consumption and end-to-end packet loss rate under multi-flow traffic. We use the same setting as explained in Section 3.3.3, but without channel errors. This ensures that packet loss are due to flow collisions at forwarders and not due the channel errors. We have varied the number of paths sharing one forwarder from one to five. Specifically, within a single run, each source picks a random period ranging from 1 sec to 60 sec. The long simulated time of one week / 168 hours guarantees the occurrence of collisions.

In Figure 5.7 we show the impact of the number of flows on the packet loss rate with and without applying the overlapping mechanism. The confidence intervals are very tight, the 95% confidence intervals for the packet loss rate is within $\pm 0.06\%$ and $\pm 0.12\%$ with and without applying the overlapping mechanism, respectively. For the energy consumption the 95% confidence intervals are within ± 0.003 joules. The figure shows that without applying the overlapping mechanism the packet loss rate increases steeply as the number of flows increases. However, when applying our overlapping mechanism the packet loss rate increases much slower. In Figure 5.8 we show the relationship between number of flows and the energy consumption for the same simulations. This figure shows that the energy consumption increases with the number of flows, presumably due to retransmissions after collisions. Furthermore, it can be seen that the overlapping mechanism has a modest additional cost over the case without the overlapping mechanism, coming from additional times that

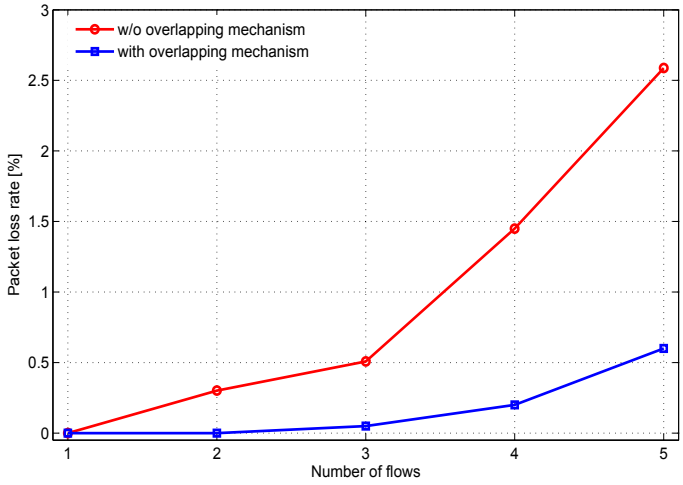


Figure 5.7: Multiple flows vs. packet loss rate

the forwarder has to be awake.

5.6.4 Length of Learning Phase

Our autonomous framework depends on obtaining good estimates of the period and the relevant quantiles (which for the assumed normal distribution boils down to finding the average and variance of the interarrival time). The quality of these estimates can be expected to depend on the length of the learning phase. To get more insight into this, we vary the length of the learning phase (expressed as number of packets to be observed) and observe both the energy consumption and packet loss rate in an otherwise error-free channel. Figures 5.9 and 5.10 show the impact of the length of the learning phase on both measures. For this result, the 95% confidence intervals are within $\pm 0.011\%$ for the loss rate, and $\pm 0.002 joul$ for the energy consumption. It is interesting to find that the packet loss rate or the energy consumption is more or less constant regardless of the length of the learning period. So the length of the learning phase does not really affect the performance. This is because the system continues to improve the estimators based on all Time Of Arrival (TOA) and reacts in a adaptive manner.

5.6.5 Length of Wakeup Window

In this section we evaluate the influence of the length of wakeup window on the performance of the system. As customary when dealing with normal distribu-

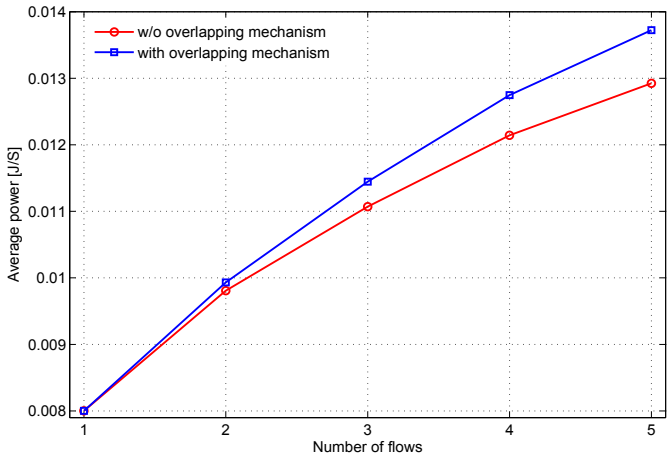


Figure 5.8: Average energy: multiple flows

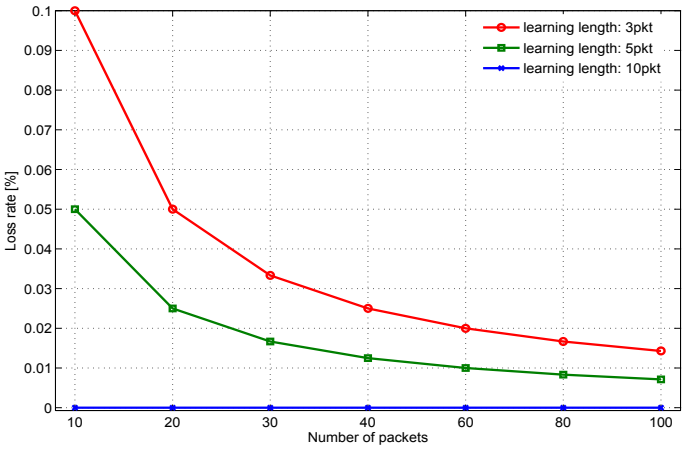


Figure 5.9: Length of learning phase vs packet loss rate

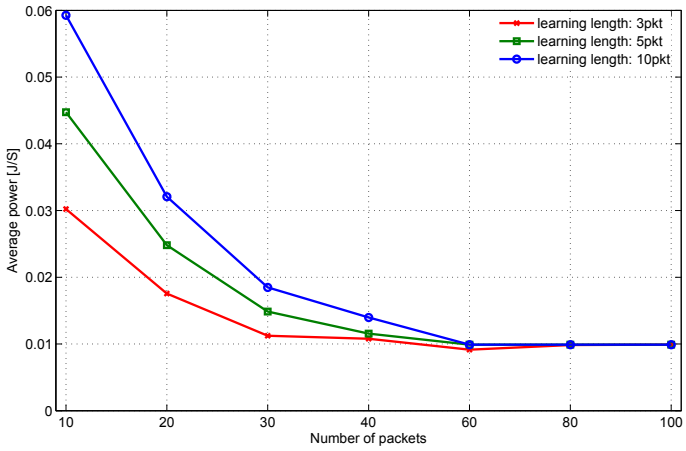


Figure 5.10: Length of learning phase vs average energy consumption

tions, we express the wakeup window as multiples of one standard deviation, σ. Figures 5.11 and 5.12 show the impact of the wakeup window length (as multiples of σ) on the loss rate and energy consumption, respectively. For these graphs, the 95% confidence intervals are within $\pm 0.17\%$ loss rate and ± 0.0035 Joules for the energy consumption. The packet loss rate behaves as one would expect: smaller values of σ lead to higher packet loss rates (remember that the default value of α is 2 in our framework). The behavior for the energy consumption is less straightforward: Figure 5.12 shows that the energy consumption for $sigma = 1$ is much higher than for larger values of σ. To explain this, we recall from Section 4.1 that a forwarder goes back from the operational state into the (much more energy-consuming) learning state after having observed too many packet losses. With $\sigma = 1$ the probability that this transition rule is triggered (after retransmissions failed) is substantially higher than for the larger values of σ. The differences in energy consumption for the larger values of σ are smaller, but for $\sigma = 3$ it is noticeably larger than for $\sigma = 2$.

5.7 Sensitivity Analysis

In the previous section we have analyzed the packet loss rate and energy consumption performance of our proposed autonomous framework. In this section we perform a sensitivity analysis in order to analyze how much the energy consumption (taken over 168 hours, see Section 5.5.2) is influenced by various factors, including important physical layer parameters and key parameters of our

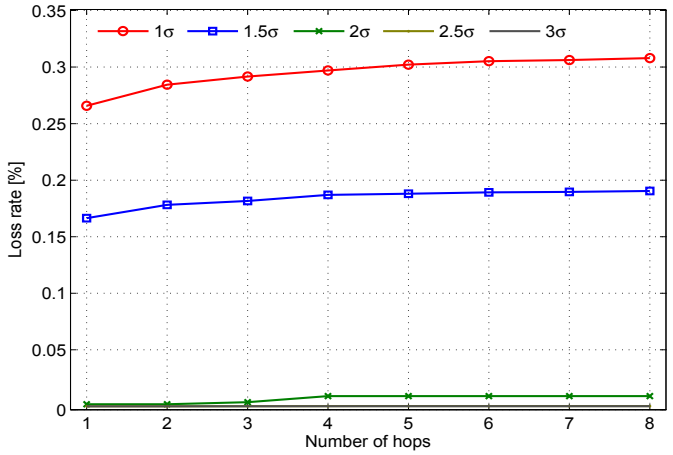


Figure 5.11: Length of wakeup window vs packet loss rate

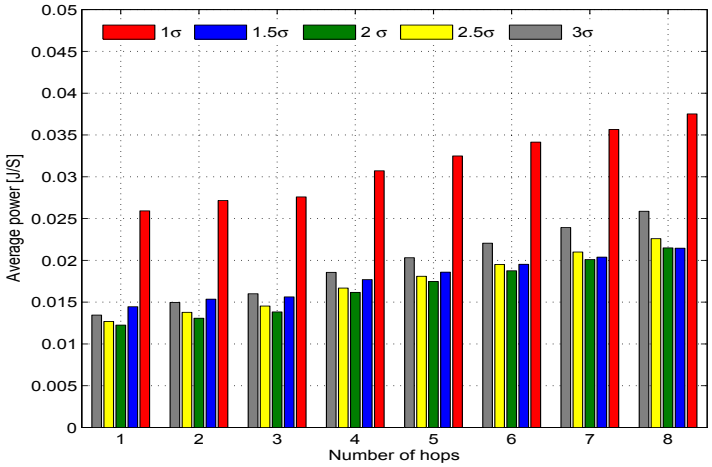


Figure 5.12: Length of wakeup window vs average energy consumption

autonomous framework. Identifying the factors contributing most to the overall energy consumption can provide useful guidance for further optimization of our framework.

We first give a brief introduction to the methodology used in this dissertation, then present the analysis results.

5.7.1 Response Surface Methodology

The Response Surface Methodology (RSM) is a statistical framework that is frequently applied to analyze systems in which a response of interest is influenced by several variables, called factors. The relation between the system response Y (performance measure), assumed to be a scalar, and the factors x_i (which need to be identified before carrying out the RSM analysis) is usually unknown. Figure 5.13 shows an example of the response surface graph which describe the relationship between the response Y and the two factor A and B. For each value of A and B there is a corresponding value of the response Y variable and we may view these values of the response Y as a surface lying above these two factors.

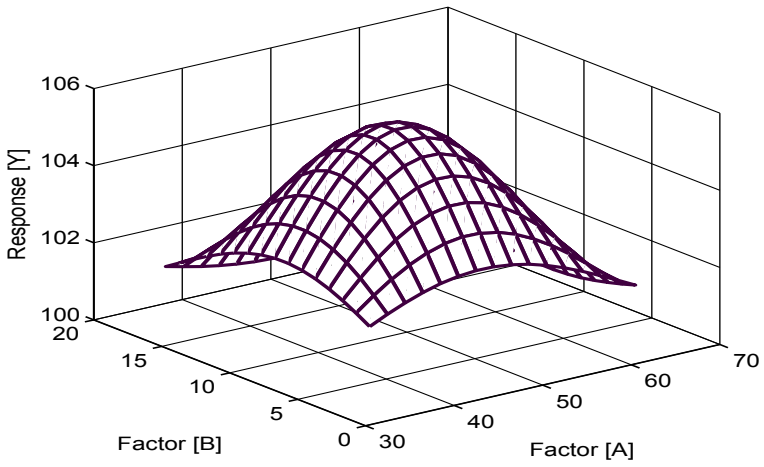


Figure 5.13: An example of response surface graph

In the course of an RSM analysis the response is approximated using a relatively simple functional form (e.g. quadratic equations) which's parameters are estimated from data, this form is also known as a regression model. Then the approximation is checked whether or not it is of sufficient quality. For more details about the RSM we point the reader to [144, 123]. Please note that RSM is not the only methodology that can be used for sensitivity analyses, see [187] for another approach.

Two popular choices for the regression model are first-order (or linear) and second-order (quadratic) models. Linear models can be represented as:

$$Y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \epsilon \tag{5.4}$$

and are appropriate when either the response is already an (almost) linear function of the factors, or when the response is a sufficiently smooth function of the factors and the factors are varied only in a small region. If none of these assumptions holds, it is appropriate to use higher-order polynomials as regression model, for example a second-order model:

$$\begin{gathered} Y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum_{i=1}^{k} \sum_{j<i} \beta_{ij} x_i x_j \\ \text{for } i = 1, 2, ..., k, \text{ and } j = 1, 2, ..., k \end{gathered} \tag{5.5}$$

In both Equations 5.4 and 5.5 the variable Y represents the response variable, k is the number of factors, the formal variables $x_1, x_2, \ldots, x_k$ are the actual factors, and the coefficients $\beta_i, \beta_{i,j}$ are the unknown regression factors. Sometimes, the coefficient β_0 is also referred to as the intercept term of the system. Many RSM studies have successfully used second-order models because of their ability to deal with response surfaces displaying interactions between factors and curvature. For these reasons we have decided to adopt the second-order model of Equation 5.5. Please note that this approach for construction of an energy-consumption model is substantially different from computing the energy consumption from first principles, like for example in the comprehensive model presented in [33].

Our response variable Y is the total energy consumption taken over 168 hours, the factors x_i are described in Section 5.7.2. Notably, they include factors of different dimensions, for example the transmission power (in mW) and the length of the wakeup window, whereas the response is given in units of Joules. This is in some contrast to Equations 5.4 and 5.5, which suggest that all factors and the response should have compatible units, and furthermore that all factors should enter with the same order of magnitude. To achieve this, formally the factors are dimension-less, whereas the response will be interpreted as being in Joule. Furthermore, we do not use real values for a factor x_i, instead we choose two "extreme" values (e.g. a minimum and maximum wakeup window length) and represent these as "$x_i = -1$" or "$x_i = 1$". With k factors one then performs 2^k experiments, ranging over all 2^k possible allocations of $\{1, -1\}$ to the k factors. Once the responses Y have been obtained for all 2^k experiments, the regression coefficients β_i and $\beta_{i,j}$ are determined,

for example using a least-squares algorithm. We have used the Matlab SUMO and statistics toolboxes for this. In our experiments we use seven factors. For each of the 2^7 factor combinations we have calculated the response as the average energy consumption taken over 100 replications. This averaging largely eliminates random fluctuations in our responses, which otherwise would have necessitated introduction of an error term into our regression model, Equation 5.5.

In order to test whether the fitted model is an adequate approximation to the true responses, we test the goodness of the fit. Several tests can be applied, for example: testing for normality of residuals, Analysis of Variance (ANOVA), lack of fit test, R^2 test, and others. We will show results for some of these tests in Section 6.2.2.

The most important information that can be extracted from the regression model is the relative importance of each factor (and all considered factor combinations), given as a percentage contribution in explaining the variation of responses across the parameter combinations taken into account. This information can be used to separate important factors from unimportant ones.

5.7.2 Factor Screening

In every RSM analysis the first step is to identify the factors, i.e. those control knobs of the system which potentially influence the response. For the average total energy consumption in our autonomous framework we have identified the following factors as potentially relevant:

- Factor A – Transmission power: the transmission power is the power consumed for transmitting data frames and acknowledgement frames.
- Factor B – Reception power: the receiving power is the power consumed while receiving data or control frames.
- Factor C – Listening power: the listening power is the radio power consumption when the radio is ready to receive frames but not actually receiving any.
- Factor D – Sleeping power: the sleeping power is the power consumption while the radio is in the low-power state.
- Factor E – Turnaround power: the turnaround power is the power consumed while switching the radio state between different modes.
- Factor F – Length of learning phase: number of packets that a forwarder processes before learning phase ends.

Table 5.2: The factors and the levels of each factor.

Term	Factor	Level 1(-1)	Level 2(+1)
A	Tx power	32.67mW	57.42mW
B	Rx power	31.68mW	62.04mW
C	Listen power	31.68mW	62.04mW
D	Sleep power	0.72mW	1.41mW
E	Turnaround power	31mW	62mW
F	Length of learning phase	5 pkts	20 pkts
G	Length of wakeup window	σ	$3 \cdot \sigma$

- Factor G – Length of wakeup window: given in multiples of the estimated standard deviation of the period, σ.

Clearly, it can be expected that for a realistic implementation on a specific hardware platform more factors might have to be added, for example the power consumed by the sensors, the microcontroller, etc. Table 5.2 lists the most relevant factors and the levels of each factor considered in our study.

5.7.3 Analysis of the Results

Table 5.3 shows the percentage which the individual factors and some of their pairwise combinations contribute to the variation of total energy-consumption over all 2^7 different factor combinations.

From the statistical analysis it can be seen that factor D (the sleep power) explains most of the variation in the energy consumption. This indicates on the one hand that our autonomous framework is doing a good job in keeping forwarders sleeping most of the time, and it indicates also that further reductions in energy consumption should be sought through using more energy-efficient hardware.

The second- and third-most important factors are factor C (listening power) and G (length of wakeup windows). Both factors C and G refer to the listening state, they determine its power consumption (C) and length (G). Clearly, the larger the length of the wakeup window, the more (potentially un-necessary) time is spent for listening. Since G is expressed in multiples of the measured standard deviation σ, it is worthwhile to consider ways to reduce σ, e.g. through changing the backoff time distribution of the underlying MAC protocol. It is important to observe, however, that σ also depends on the position of the forwarder in the chain, i.e. on how many hop counts it is away from the source. As we have shown in Section 4.9.2 the jitter depends on this position, and so

Table 5.3: The percentage of factors contribution.

Term	Sum of Squares	Percentage contribution
A	525.26	1.01
B	624.26	1.2
C	**3662.12**	**13.21**
D	**34790.93**	**66.12**
E	72.45	0.14
F	945.84	1.84
G	**3462.12**	**10.13**
AB	121.98	0.20
AC	22.24	0.032
AF	15.85	0.023
AG	29.42	0.05
CF	121.98	0.20
CG	1714.76	3.41
FG	964.96	1.92
ACF	156.48	0.31
AFG	16.04	0.025
Error	$1.202 \cdot 10^{-4}$	

Table 5.4: ANOVA for total energy consumption.

Source	df	Sum of squares	Mean Square	F-value	Prob > F value
Model	16	43568.53	2723.033125	3341.94	0.0001
A	1	525.26	525.26	39.98	0.0001
B	1	624.26	624.26	46.57	0.0001
C	1	3662.12	3662.12	508.64	0.0001
D	1	34790.93	34790.93	2547.82	0.0001
E	1	72.45	72.45	5.64	0.0415
F	1	945.84	945.84	71.54	0.0001
G	1	3462.12	3462.12	390.68	0.0001
AB	1	121.98	121.98	8.96	0.0148
AC	1	22.24	22.24	1.68	0.2
AF	1	16.04	16.04	1.37	0.23
AG	1	29.42	29.42	2.35	0.19
CF	1	121.98	121.98	8.96	0.0148
CG	1	1714.76	1714.76	131.84	0.0001
FG	1	964.96	964.96	74.68	0.0001
ACF	1	156.48	156.48	12.64	0.0092
AFG	1	15.85	15.85	1.61	0.24
Error	111	$5.67 \cdot 10^{-5}$	$1.27 \cdot 10^{-7}$		
$R^2 = 0.99$					

forwarders closer to the sink will experience larger values of σ. Given this, the forwarder position could be identified as an eight factor, but we will investigate its importance more closely in future work.

Clearly, factor G can also be reduced by reducing the wakeup window length so that the resulting packet loss rate still matches application requirements. A reduction in factor C (listening power) again would require advances in transceiver technology.

Please note that most of the quadratic factors $x_{i,i}$ and several of the mixed factors $x_{i,j}$ have negligible weights. The regression analysis using the least-squares estimation method on the values of the response obtained from the various combinations of the factors yields the following Equation (5.6) for the total energy-consumption:

$$\begin{aligned} T_E = &+150.32 + 4.05A + 4.42B + 12.87C + 39.68D \\ &+ 1.12E + 5.85F + 10.95G + 1.82AB - 0.84AC \\ &+ 0.67AF + 0.95AG + 1.86CF + 6.45CG + 5.23FG \end{aligned} \tag{5.6}$$

When a fitted model has been derived, it is necessary to apply further statistical tests to ensure that it provides an adequate approximation to the true system. We have conducted a range of statistical tests, using MATLABs facilities. Specifically, in Table 5.4 we show the results of an analysis of variance

(ANOVA). The ANOVA reveals the influence of each design factor on the total energy-consumption and their statistical significance through the F-test and associated probability ($Prob > F$). The F-value represents the ratio of the mean squares of the model over the mean squares of the residual. The F-value is compared to the reference distribution for F, and the result allows to determine the probability that a result is observed due to error. When the value in the last column of table 5.4 is below 0.05 (at a 95% significance level), we can conclude that the factor is statistically significant. In this case there is a very small probability, near 0.01%, that the differences in the factors model averages are due to statistical fluctuations. The results given in Table 5.4 demonstrate that all the elementary factors A to G and the compound factors AB, CF, CG and FG are highly significant, together they explain almost all the variation. Based on the ANOVA test we have simplified the regression model by excluding insignificant factors to become:

$$\begin{aligned} T_E = &+150.32 + 4.05A + 4.42B + 12.87C + 39.68D \\ &+ 1.12E + 5.85F + 10.95G + 1.82AB + 1.86CF \\ &+ 6.45CG + 5.23FG \end{aligned} \tag{5.7}$$

We also examine the fitted model to ensure that it provides an adequate approximation to the true system by conducting the following procedures:

Firstly, we compare the observed responses with responses predicted by the regression model (see Figure 5.14).

This figure shows that the regression model is fairly well fitted with the observed values.

Secondly, we test whether the residuals of total energy consumption are normally distributed. Figure 5.15 shows the normal probability plot of the residuals for total energy consumption. The residuals are falling on a straight line, which means that the errors are normally distributed.

Finally we checked for absence of correlation between residuals and the order in which runs are carried out. Figure 5.16 shows the residuals against run order.

To sum up, all these tests confirm that the regression model is a very good approximation of the real total energy consumption.

5.7.4 Impact of Traffic Density

In our final analysis we have looked at the impact of the intensity of data traffic on the overall energy consumption. To keep the presentation simple, we have analyzed how the coefficients β_i for the main factors A, B, C etc. (which in total have far more impact on the response than the combined factors AB, CF and so on) change when either the number of data sources or the data

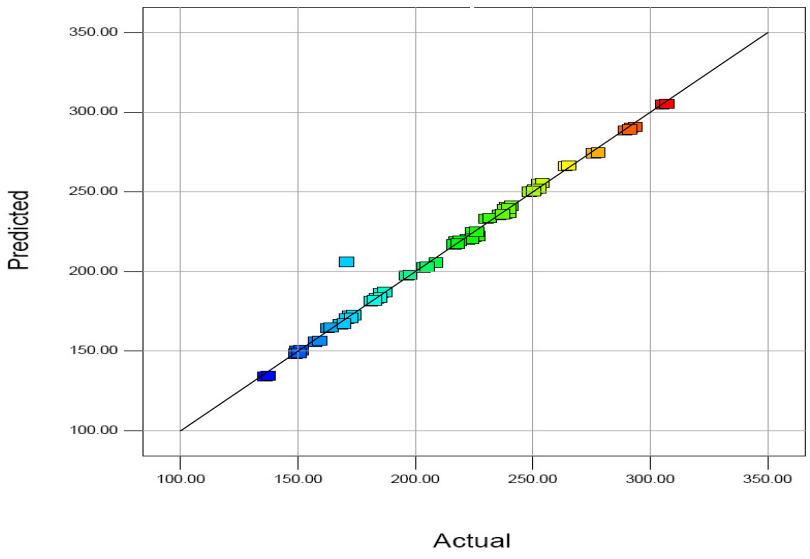


Figure 5.14: Test of the Predicted vs. actual values

generation rate changes.

Specifically, in Figures 5.17 and 5.18 we show the impact of the number of sources and traffic rates, respectively. Please note that for Figure 5.17 we vary the number of sources from 2 to 10 sources. It can be seen that the variation is minor, the most sensitive parameters are the regression coefficients β_C, β_D and β_G for factor C (listening power), D (sleeping power) and factor G (length of wakeup windows), respectively. This makes sense intuitively, as with increasing number of sources there is more cross-traffic and the measured standard deviation σ increases, making the wakeup windows longer and shortening the time in sleep state. Figure 5.18 shows the impact of the traffic rate (data generation period) on the regression coefficients β_i. One can see from this Figure (5.18) that the regression coefficients β_A to β_D vary somewhat between a rate of 1 sec on the one hand and the rates of 30 and 60 secs on the other hand, the remaining factors are more or less the same for all traffic generation rates.

5.8 Summary

In this chapter we presented a scheme which supports periodic traffic flows and frequency hopping without requiring an expensive protocol infrastructure providing synchronization features (time synchronization, hopping synchronization) by relying entirely on the periodicity of the traffic itself for synchronization purposes. Our approach is extremely light in terms of signaling, only ACK packets need to carry few bits of information. We believe that our ap-

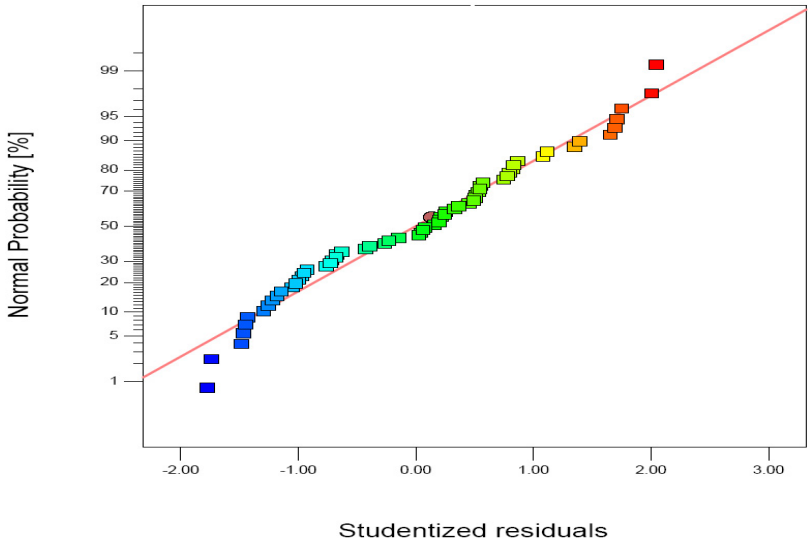


Figure 5.15: Test of the normal probability plot

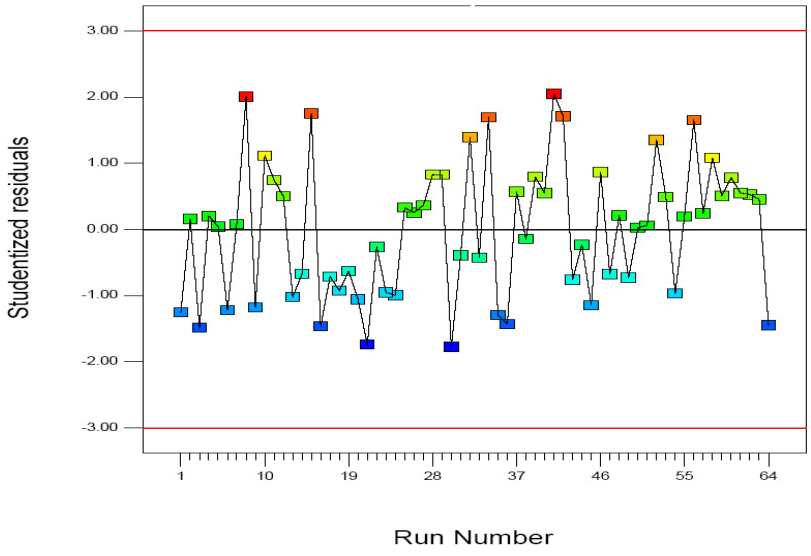


Figure 5.16: Test of the residuals vs. run number

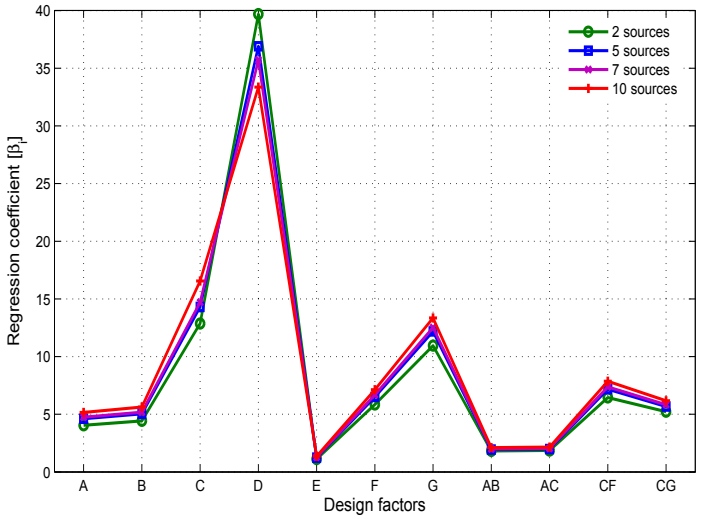


Figure 5.17: Impact of the number of sources on regression coefficients β_i

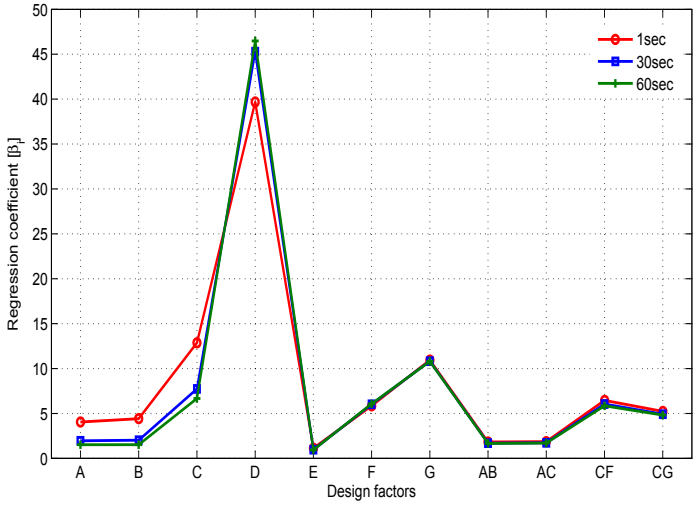


Figure 5.18: Impact of traffic density on regression coefficients β_i

proach is an attractive alternative to WirelessHART and similar systems in lightly loaded networks with periodic traffic.

The main contributions of this chapter are:

- We designed and evaluated a novel asynchronous channel hopping mechanism, which we have integrated with the distributed wakeup scheduling

scheme for multi-flows traffic.

- We also evaluated the impact of overlapping traffic and introduced an overlapping controller that exploits the traffic estimation values to detect and resolve a potential collision beforehand.
- We also proposed an energy management policy which exploits the available sleep modes and utilized them in an efficient manner.
- Furthermore, we provided a sensitivity analysis which shows how the energy consumption depends on certain parameters, including the power consumption of the transceiver in different modes of operation (Tx, Rx, Listening, Sleeping, and turnover), as long as the length of learning phase and length of wakeup window factors. In this analysis we use the response surface methodology and show the most influenced parameters to the energy consumption
- We have evaluated the proposed scheme in a range of scenarios using trace-based simulations, and we have shown that it indeed reaps the benefits of frequency hopping and can also improve energy consumption over systems working on just a single channel. Furthermore, our results show that the proposed scheme works at a very good level of reliability, and in addition has only little implementation complexity.

CHAPTER 6

Performance Evaluation of WirelessHART Protocol

In this chapter we analyze in detail the energy consumption characteristics of the benchmark protocol, WirelessHART. We analyze how much various factors contribute to the overall energy consumption over a longer period. These factors include the amount of management traffic and the power levels required for various transceiver activities (transmit, receive, listen, sleep). It turns out that in light traffic scenarios and with only a minimum-complexity level of exploitation of the transceivers sleeping capabilities the energy spent in the sleep state is quite substantial. We then proceed to analyze the energy consumption characteristics with a more complex usage of the popular transceiver, the ChipCon CC2420 sleeping capabilities in which each node individually selects its next sleep state according to its transmission / reception schedule. With this scheme the energy consumption in the sleep state can be reduced substantially.

6.1 Performance Evaluation Approach

In this section, we describe our approach for evaluating the energy consumption of the WHART TDMA protocol. Since we are mostly interested in the energy consumption of the field devices, we compute it only for the source and forwarder nodes. We do not take into account the gateway energy-consumption in our calculation as its assumed to be directly connected to a power source.

6.1.1 Simulation setup

In order to realize a simulation model to study the performance of WHART TDMA over a wireless multi-hop network, we have chosen the OMNeT++

Table 6.1: WirelessHART-specific parameters

Number of slots per superframe	100s
Slot-time length	10ms
Synchronization frame size	26 bytes
Re-synchronization rate	60s
Health report rate	15min

[167] simulation environment together with the Castalia framework [128]. OMNet++ is an open-source discrete-event simulator, Castalia is an OMNet++ based framework designed specifically for wireless sensor networks. We set the radio parameters based on the IEEE 802.15.4-compliant ChipCon CC2420 radio chip [30]. For channel errors we use a channel model provided by Castalia (please refer to the Castalia users manual [128]). Please refer to Section 3.3.3 for more details about the simulation models used in this evaluation.

For our implementation of the WHART TDMA scheme we modeled the cost of channel switching. Channel switching is modeled in both receiving and transmission slots within the TxOffset and RxOffset time intervals, respectively (see Section 2.4 for more details about these time intervals).

Based on the characteristics of the CC2420 we have assumed that channel switching takes $192\mu s$ and consumes the same amount of power as the transceiver's listen state. The transceiver has four main operational states: transmit, receive, listen and sleep. Given that for long-running applications most of the energy will be spent in the operational phase of a MAC protocol, we have chosen to ignore the initialization phase, in which for example the nodes report their neighbor lists and channel qualities, the network manager disseminates the schedules, and initial time synchronization is established. The TDMA schedule is precomputed based on the Fastest Periodic Flow First (FPFF) scheduling algorithm [90, 95] and given as input to the simulator.

In Table 3.1 the main power consumption parameters of a CC2420 transceiver and MSP430 micro-controller are summarized assuming a 3.3V supply voltage. These parameters are used in the physical layer model of our simulator to obtain energy-consumption results. The sleep power mentioned in the table refers to the lightest of four available sleep states. Please note that the micro-controller is active at the same times as the transceiver. Further simulation parameters related to the physical and MAC layer properties and the node deployment are given in Table 3.3 and Table 6.2, respectively.

In addition to the micro-controller we have also taken the power consumption of the micro controller into account. More precisely, we use a MSP430 micro controller from Texas Instruments. This controller alternates between

Table 6.2: WHART TDMA MAC parameters.

WHART MAC parameters		
Notaion	Parameters	value (ms)
TxOffset	The guard time at beginning of time slot at the sender side.	2.12
RxOffset	The guard time at beginning of time slot at the receiver side.	1.12
RxWait	The time to wait for start of frame	2.2
Maxframe	Maximum frame length	4.256
TxAckDelay	The time between end of frame and start of ACK at the receiver side.	0.8
RxAckDelay	The time between end of frame and start of listening for ACK at the sender side.	0.9
AckWait	The minimum time to wait for start of an ACK	0.4
Ack	ACK (26 bytes)	0.832
RxTx	TxRx turnaround time	0.192

two different states: an active state, in which it performs computations, and a sleep state in which major parts of its circuitry are switched down. The power consumption figures of both states are given in Table 3.1.

6.1.2 Network Topology and Traffic

In order to perform our evaluation, we use randomly generated topologies. More precisely, we have generated 100 random topologies and for each setting of simulation parameters we correspondingly perform 100 replications, the energy consumption results of which are then averaged. For each random topology we have placed 150 nodes in an area of size $120 \times 120\text{m}^2$, using a uniform distribution for node positions. The gateway is placed in the upper right corner of the field (see Figure 3.3). Out of the 150 nodes we randomly pick ten nodes as source nodes. Each of these sources periodically generates frames of 133 bytes total size (including PHY and MAC parts). The generation period was varied, taking values of 1, 30, or 60 seconds, respectively. During each simulation-run, each source transmits frames based on its periodicity and then forwards these frames to the GW via some forwarders. MAC-layer acknowledgments are enabled and the size of the ACK frame is 26 bytes. If the frame is lost due to channel errors the sender tries to transmit the frame for a maximum of two retries.

6.1.3 Major Performance Measure

The simulation time is fixed to 24 hours for each scenario and the main performance measure is the total energy spent by the radio transceiver and the

micro-controller of a node over this period. This time duration of 24 hours is on the one hand long enough to give meaningful results for the energy spent in sleep states, and on the other hand is not too taxing on simulation times.

For the transceiver, the simulation records the amount of time spent in various states (transmit, receive, listen, sleep and turnover) and calculates from this the total energy consumption of a node over a span of 24 simulated hours. We also model the energy spent in switching between states by multiplying the turnover time by the power consumed in the most power-consuming of the two involved states. We also take into consideration the energy consumed by the node's micro-controller. We split the node's micro-controller energy consumption into two main states, active state and sleep state. The micro-controller is active at the same time as the radio. At the end of each run, the simulation computes the total energy consumed for all field devices in the network using the amount of energy consumed by the radio and micro-controller in each state.

This 24-hour energy consumption is obtained for each source and router node separately. In the next sections we report mainly the average energy consumption, taken over all nodes.

6.2 Sensitivity Analysis

The major goal of our study is not only to obtain insights into the overall network energy-consumption of the different types of nodes (sources and forwarders), but we also want to obtain some insights on how the average total consumption (taken over 24 hours, see Section 6.1.3) breaks down into different factors. By identifying the factors contributing most to the overall energy-consumption, we can provide guidance on where to start with any effort geared towards saving energy.

It is reasonable to assume that, given the long time span of 24 hours over which energy consumption is computed, the energy spent in the sleep state will play a significant role. Therefore, we need to clarify our assumptions on how the four available sleep states of the CC2420 (compare Section 6.3) are used in this analysis. As stated previously, the only sleep state that works indiscriminately for every possible schedule is the lightest sleep state, which we have used here and is given in Table 3.1. This sleep mode is entered after each active slot and its wakeup time is quick enough so that wakeup can commence at the beginning of the nodes next active slot. An advantage of this method is that there are no run-time costs for determining the next sleep state. In the next section 6.3, we report a simple method which for a known schedule exploits the other available sleep states to obtain higher savings.

Our major tool for energy consumption analysis is the Response Surface

Methodology (RSM). RSM is a collection of mathematical and statistical techniques useful for modeling and analysis of problems in which a response of interest is influenced by several variables (factors). Usually the relation between the response variable Y (performance measure) and the input variables (or "factors") x_i is unknown. After identifying the factors and the response(s) under study, the next step in RSM is to find a suitable approximation for the response surface and check whether or not this model is adequate.

The two most popular choices for regression models are first-order and second-order models, presented in Equation 5.4 and Equation 5.5, respectively.

The first-order model is mostly appropriate when one is interested in approximating the true response surface over a very small region of the input variables, where the response surface is approximately constant (no curvature). If there is curvature in the system, then a polynomial of higher degree must be used, such as the second-order regression model.

Many studies in RSM show that second-order models are capable of solving real response surface problems that have interactions and curvature. This is due to the fact that second order regression model offers a wider variety of functional forms. For these reasons we have decided to adopt the second-order model of Equation 5.5. For more details about the RSM we point the reader to the [144, 123].

We have used the total energy consumption taken over 24 hours as the response variable Y. The factors x_i are described in the next Section 6.2.1, they include the transmission power, reception power, the rate of management packets and three others. Please note that in Equations 5.4 and 5.5 the response variable Y should have the same dimension as the input variables, and all factors x_i should enter with the same order of magnitude. In applications like ours, often the factors have incompatible units (like the transmit power expressed in mW or the management rate in Hz, see Section 6.2.1). To accommodate this, all variables are formally dimension-less, although the response variable will be interpreted as being in mW. Furthermore, the factors in the RSM method are not used directly, but one chooses two "extreme" values of a factor (e.g. the minimum and maximum transmit power) and encodes these as "-1" and "1", respectively. When there are k factors, one performs 2^k experiments over the set $\{-1, 1\}^k$ of all possible allocations to factors. Once the response values Y have been obtained for all experiments, the regression coefficients β_i and $\beta_{i,j}$ are determined for the encoded input factors x_i (i.e. assuming that these input factors assume the values -1 and 1, respectively). We have used the Matlab SUMO[1] and statistics toolboxes for this. Internally, these toolboxes use a least-squares algorithm for estimating these coefficients. In fact, for each of the 2^6 possible factor combinations we have performed

[1]See `www.sumo.intec.ugent.be/?q=sumo_toolbox`.

100 replications and used the average total energy consumption, taken over all these replications, as our response variable. This averaging is the reason why the model 5.5 does not include a separate error term, as with 100 independent replications the simulated averages will be close to the true averages.

After identifying the appropriate empirical fitted model, we examine this model to ensure that it provides an adequate approximation to the true responses and to verify that none of the assumptions for least squares regression are violated. There are several techniques for testing the model adequacy and the goodness of the fit, for example: testing for normality of residuals, analysis of variance (ANOVA), lack of fit test, R^2 and others. We will show some results of these tests in Section 6.2.2. If the fitted surface is an adequate approximation of the true response function, then analysis of the fitted surface will be approximately equivalent to analysis of the actual model.

A key information that can be derived from the regression model is the percentage contribution of each factor (and all considered factor combinations) in explaining the variation of results across the parameter combinations taken into account.

6.2.1 Factor Screening

The first step in the RSM is to identify potential factors affecting the response being measured (factor screening). Since the average total energy-consumption is the main response, we consider the following factors:

- Factor A – Transmission power: the transmission power is the power consumed for transmitting data frames and control frames such as synchronization frames.
- Factor B – Reception power: the receiving power is the power consumed while receiving data or control frames.
- Factor C – Listening power: the listening power is the radio power consumption when the radio is on but not receiving or sending any frames.
- Factor D – Sleeping power: the sleeping power is the power consumption while the radio is in the low-power state.
- Factor E – Turnaround power: the turnaround power is the power consumed while switching the radio state between different modes.
- Factor F – Management rate: The energy consumed for maintaining the WHART network operations. This is controlled by the frequencies of synchronization, advertisement, join request/response, commands, keep-alive and health report control frames.

Table 6.3: The factors and the levels of each factor.

Term	Factor	Level 1(-1)	Level 2(+1)
A	Tx power	32.67mW	57.42mW
B	Rx power	31.68mW	62.04mW
C	Listen power	31.68mW	62.04mW
D	Sleep power	0.72mW	1.41mW
E	Turnaround power	31mW	62mW
F	Management rate	60 sec	120 sec

Please note that for a realistic application and hardware platform there would be some more factors influencing the energy consumption, for example the power consumed by the sensors. However, we have identified the above listed factors as the most relevant for our purposes. Table 6.3 lists the factors and the levels of each factor considered in our study.

6.2.2 Analysis of the Results

Table 6.4 shows the percentage which the individual factors and their pairwise combinations contribute to the variation of total energy-consumption over all 2^6 different factor combinations.

From the statistical analysis it can be seen that factor C (listening power), D (sleep power) and F (Management rate) contribute most to the total energy-consumption. Please note that most of the quadratic factors $x_{i,i}$ have negligible weights (see our technical report [95] for complete tables of all factors). The regression analysis using the least-squares estimation method on the values of the response obtained from the various combinations of the factors yields the following Equation (6.1) for the total energy-consumption:

$$\begin{aligned} T_E = {} & 111.33 + 2.59A + 2.83B + 11.94C + 13.77D \\ & + 0.71E - 12.73F - 0.19AB - 0.34AC + 0.74AD \\ & - 0.34AE + 0.39AF - 0.34BC + 0.74BD - 0.34BE \\ & - 0.41BF + 0.59CD - 0.19CE - 3.66CF - 0.34DE \\ & - 0.11DF - 0.40EF + 0.34ABC - 0.74ABD \end{aligned} \tag{6.1}$$

It is always necessary to examine the fitted model to ensure that it provides an adequate approximation to the true system model. To check the model adequacy, and to get more statistical details of the factors affecting the total energy-consumption, we perform an analysis of variance (ANOVA) as

Table 6.4: The percentage of factors contribution.

Term	Sum of Squares	Percentage contribution
A	428.82	1.26
B	513.17	1.51
C	**9131.35**	**26.92**
D	**12129.30**	**35.76**
E	32.02	0.09
F	**10375.50**	**30.59**
AB	2.38	$7.022 \cdot 10^{-3}$
AC	7.41	0.02
AD	34.90	0.10
AE	7.41	0.02
AF	9.70	0.03
BC	7.41	0.02
BD	34.91	0.10
BE	7.41	0.02
BF	10.71	0.03
CD	22.37	0.07
CE	2.38	$7.025 \cdot 10^{-3}$
CF	859.56	2.53
DE	7.41	0.02
DF	0.72	$2.11 \cdot 10^{-3}$
EF	10.03	0.03
ABC	7.41	0.02
ABD	34.91	0.10

shown in Table 6.5 (this and the other statistical tests were carried out with MATLAB). The ANOVA shows the effect of each design factor on the total energy-consumption and their statistical significance through the F-test and associated probability ($Prob > F$). The F-value column reports the ratio of the mean squares of the model over the mean squares of the residual. The F-value is compared to the reference distribution for F, in order to determine the probability of observing this result due to error. If the value in the last column of the table is less than 0.05 (at a 95% significance level), then the factor is statistically significant. In other words, there is a very small probability, near 0.01% that the differences in the factors model averages are due to the chance variation. The results given in Table 6.5 demonstrate that all the elementary factors A to F and the compound factor (CF) are highly significant, together they explain almost all the variation. Based on the ANOVA test we have simplified the regression model by excluding insignificant factors to become:

$$\begin{aligned} T_E = 111.33 + 2.59A + 2.83B + 11.94C + 13.77D \\ + 0.71E - 12.73F - 3.66CF \end{aligned} \tag{6.2}$$

We have applied a number of further tests (comparison of observed responses with responses predicted by the regression model, testing whether the residuals of total energy consumption are normally distributed, and a check for lack of correlation between the residuals and the order in which runs are carried out), and all these tests confirm that the regression model is a very good approximation of the real total energy consumption. The details are reported in the technical report [95].

6.2.3 Impact of Traffic Density

In our next analysis we have looked at the impact of the intensity of data traffic on the overall energy consumption picture. To keep the presentation simple, we have analyzed how the coefficients β_i for the main factors A, B, C etc. (which in total have far more impact on the response than the combined factors AB, AC and so on) change when either the number of data sources or the data generation rate changes.

Specifically, in Figures 6.1 and 6.2 we show the impact of the number of sources and traffic rates, respectively. Please note that for Figure 6.1 we vary the number of sources from 2 to 10 sources. It can be seen that the most sensitive parameter is the regression coefficient β_C for factor C (sleeping power). Figure 6.2 shows the impact of the traffic rate on the regression coefficients β_i (here the number of sources is fixed to 10). One can see from this Figure (6.2) that the regression coefficients β_i are more of less the same

Table 6.5: ANOVA for total energy-consumption.

Source	df	Sum of squares	Mean Square	F-value	Prob > F value
Model	23	$3.367E \cdot 10^4$	1464.23	238.30	0.0001
A	1	428.82	428.82	69.79	0.0001
B	1	513.17	513.17	83.52	0.0001
C	1	9131.35	9131.35	1486.08	0.0001
D	1	12129.30	12129.30	1973.99	0.0001
E	1	32.02	32.02	5.21	0.0278
F	1	10375.54	10375.54	1688.57	0.0001
AB	1	2.38	2.38	0.39	0.5371
AC	1	7.41	7.41	1.21	0.2787
AD	1	34.90	34.90	5.68	0.0220
AE	1	7.41	7.41	1.21	0.2786
AF	1	9.70	9.70	1.58	0.2162
BC	1	7.41	7.41	1.21	0.2787
BD	1	34.91	34.91	5.68	0.0220
BE	1	7.41	7.41	1.21	0.2786
BF	1	10.71	10.71	1.74	0.1942
CD	1	22.37	22.37	3.64	0.0636
CE	1	2.38	2.38	0.39	0.5370
CF	1	859.56	859.56	139.89	0.0001
DE	1	7.41	7.41	1.21	0.2787
DF	1	0.72	0.72	0.12	0.7346
EF	1	10.03	10.03	1.63	0.2088
ABC	1	7.41	7.41	1.21	0.2787
ABD	1	34.91	34.91	5.68	0.0220
Error	40	$2.4578 \cdot 10^{-6}$	$6.14 \cdot 10^{-8}$		
$R^2 = 0.99$					

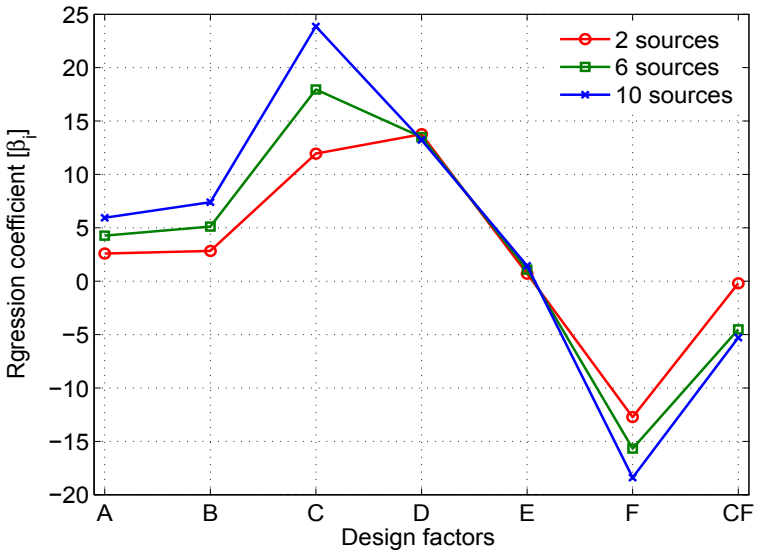


Figure 6.1: Impact of the number of sources on regression coefficients β_i

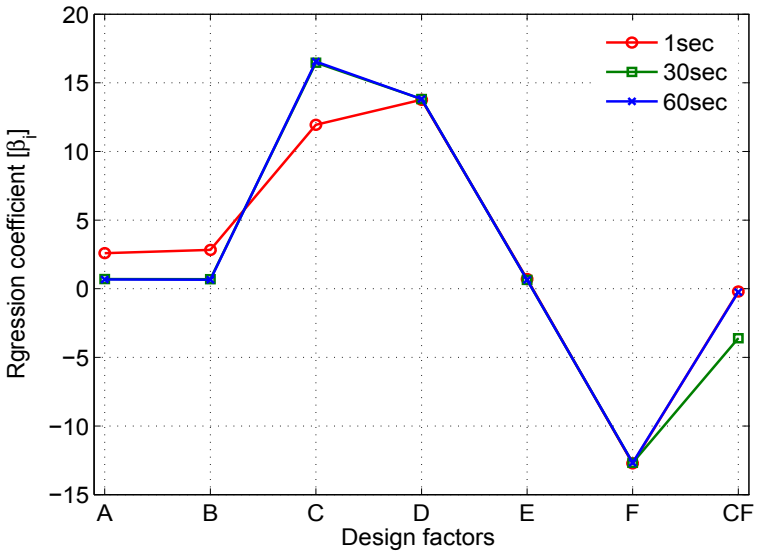


Figure 6.2: Impact of traffic density on regression coefficients β_i

in case of low traffic rate such as 30 and 60 seconds.

6.2.4 Impact of Control frames on the Energy-Consumption

In order to investigate the impact of the control frames (time synchronization and management frames) on the total energy-consumption, we compare the same setup as described in Section 6.1.1 with and without running the synchronization protocol as well as the management frames. Please note that in the WHART standard there are two separate superframes. One is used for data frames (data slots) and the other one used for management frames (management slots). Thus, management frames are transmitted separately and cannot be piggybacked onto data frames. We distinguish two types of control overhead slots: synchronization control overhead slots (SCOS) and management control overhead slots (MCOS). The SCOS includes the slots that are responsible for synchronization, such as keep-alive frames. On the other hand, MCOS includes the frames (advertisements, request/response frames, commands frames, health report frames etc.) send throughout the network to establish and maintain the WHART operations.

To show how the SCOS and MCOS affect the total energy consumption, we first analyze the impact on the regression coefficients β_i for different management traffic rates. In Figures 6.3, 6.4, and 6.5 we show these results for management / synchronization traffic periods of 1 s, 30 s, and 60 s, respectively. The variations in management traffic rates have significant impact on the regression coefficients for factors C (Listen power) and F (Management rate). Note that the curve "w/o SCOS" refers to a setup in which neither synchronization nor management slots are present. We can also observe from Figures 6.4 and 6.5 that with low management traffic rates of 30 and 60 seconds, the β_i parameters are very stable for all the factors. Please note that, the remaining higher order terms such as AA,AB,AC etc. have negligible impact on the regression analysis.

In addition, we also provide results showing how serious the impact of SCOS and MCOS is on nodes being in different hop distance to the gateway. Specifically, we compute the average total energy consumption for all nodes being one hop away from the gateway (H1), for all nodes being two hops away (H2) and so on. The energy-consumption of the nodes near the gateway varies is based on the number of control frames received by these nodes, this is shown in Figure 6.6 for a scenario with ten sources and a data generation period of 1 second (the results for data generation periods of 30 and 60 seconds exhibit the same pattern and are shown in the technical report [95]).

We can also observe from Figure 6.6 that all the nodes have the same energy-consumption with respect to hop number for the no-synchronization scenario (w/o SCOS). The energy-consumption in case of synchronization frames (SCOS) and management frames (MCOS) becomes higher as the nodes gets closer to the gateway. This is due to the high number of slots needed in

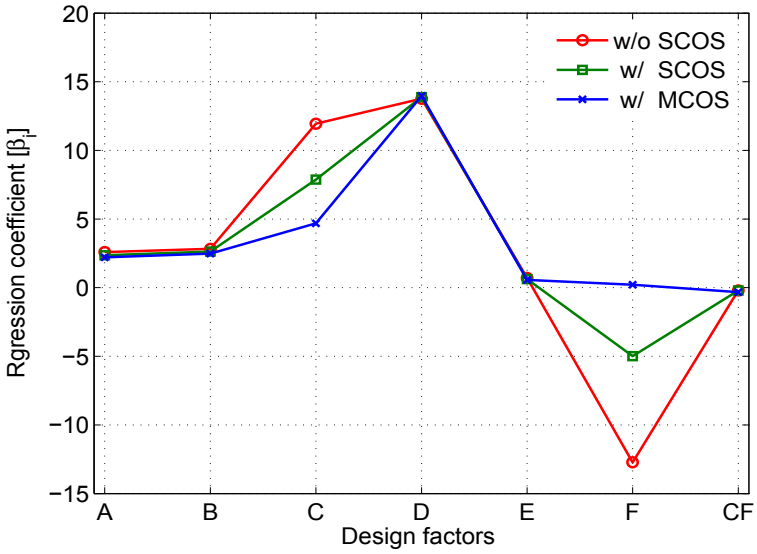


Figure 6.3: Impact of control frames on regression coefficients β_i in case of 1sec.

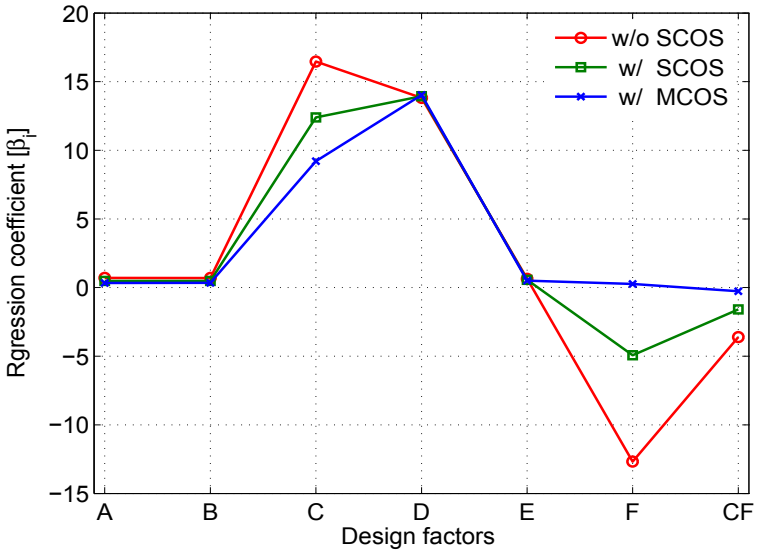


Figure 6.4: Impact of control frames on regression coefficients β_i in case of 30sec.

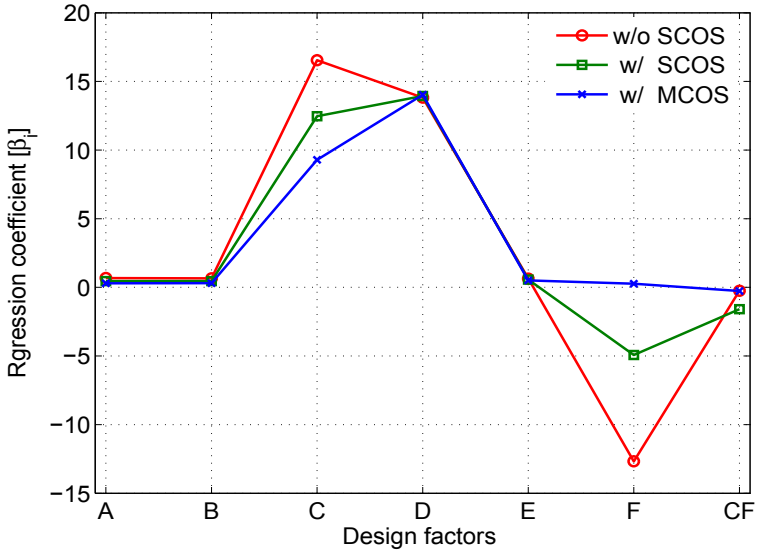


Figure 6.5: Impact of control frames on regression coefficients β_i in case of 60sec.

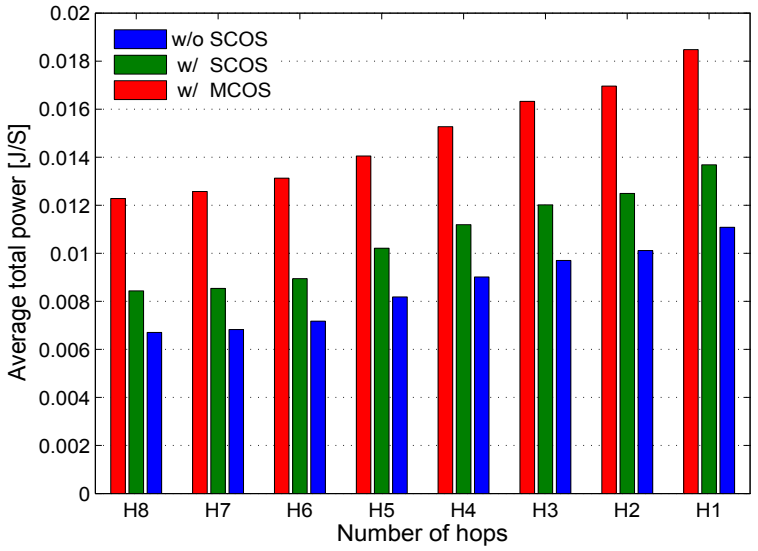


Figure 6.6: Control overhead cost in case 1sec: each hop corresponds to the set of neighbors that are n hop away from the gateway.

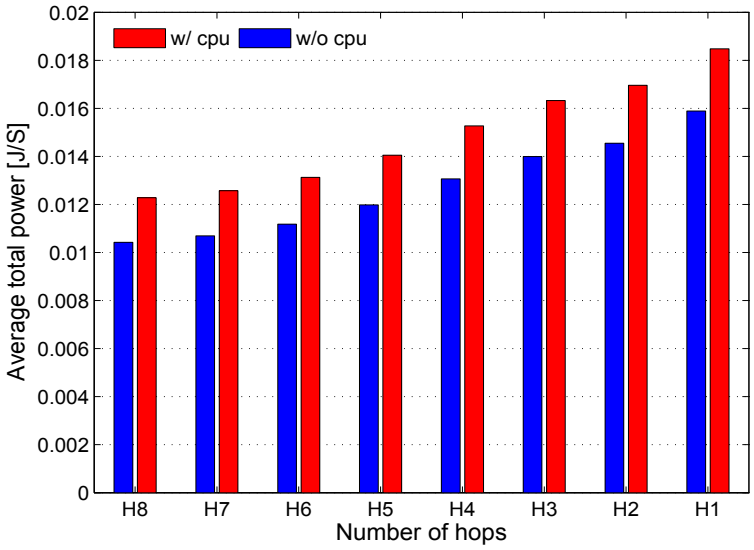


Figure 6.7: Average energy consumption with and without considering the micro-controller for traffic period of 1 s. Each hop corresponds to the set of neighbors that are n hops away from the gateway.

order to forward the MCOS traffics. In light of that, it is much more economic to piggyback some percentage of the MCOS traffics on the data frames to save energy. Indeed, this might save a significant amount of energy in case of high data rates.

Finally, we have a closer look at the influence of the power consumption of the micro-controller. Specifically, in Figure 6.7 we compare the average total power consumption for varying hop distance to the sink with and without taking into account the micro-controllers power consumption. It can be seen that the micro-controller (which is active at the same time as the transceiver) accounts for difference of approximately 15% of the total power consumption, but clearly the transceiver has much bigger influence on total power consumption.

6.2.5 Discussion

Our results indicate that the time spent in the sleeping state is the major contributor to the total energy consumption. This is due to the long sleeping intervals that accumulate over the lifetime of the network and restricting to the lightest possible sleep state. The second main contributor is channel listening in the data slots and management slots. Most of the activity within these slots (waking up from sleep mode, switching transceiver into right mode, transmitting or receiving data or ACK frames) is inevitable, but especially re-

ceiving nodes have to spend some of their listening time (in total about 1 ms) to accommodate clock inaccuracies. The energy consumption increases when data frames are lost and there are retransmissions.

The presence of management and synchronization slots also has significant influence on total energy consumption. These slots require nodes to switch the transceiver on, and to transmit and receive frames without effecting useful data transfer. The frequency of these frames has a significant influence on the total energy consumption. WHART control frames include re-synchronization, advertisements, request/response, commands, keep-alive and health report frames. The relative energy cost incurred from the management frame frequency becomes especially significant in case of low traffic rates. We believe that the issue of selecting the right management frequency is worth further investigation. One idea is to adapt the frequency of management traffic by starting with a high frequency, and as soon as the network becomes somewhat stable, the management rate can be reduced. This can significantly reduce energy saving at the expense of longer joining times and slower network update times resulting from topology changes. Another approach would be to use piggybacking more extensively, for example to use periodic data packets also for management purposes by piggybacking additional information (e.g. keep-alive and health reports).

6.3 Local Dynamic Sleep State Scheduling for TDMA Protocols

In this section we analyze how an improved usage of the transceiver sleep states can substantially reduce the overall energy-consumption, thereby increasing the WHART TDMA system energy-efficiency. Based on the previous WHART sensitivity analysis using only the lightest sleep state we found that indeed the energy spent in sleep state is one of the major factors influencing the total energy-consumption.

As we have discussed in Section 6.2.5, in the WHART TDMA system a node sleeps for the most part of its life and the energy consumed in sleeping state reaches a substantial share of overall energy consumption over 24 hours when sleep-mode-1 is used throughout (which works with any schedule) – the sleep energy amounts to almost 40% of the total observed variation in responses.

One approach to exploit the other sleep states as well would be to make the fixed-length WHART time slot of 10 ms duration somewhat longer, so as to allow wakeup from deeper sleep states within a time slot. However, this would require a change to the standard itself, and it would also affect existing implementations.

In the remainder of this section we propose a simple approach that does

not require any changes to the standard and which each node can apply individually, based on its schedule.

6.3.1 Energy Management Mechanism

In what follows, we have adapted the energy management mechanism DM3S introduced in 5.2 to improve the energy efficiency for TDMA-based systems. It exploits the multiple sleep states of the CC2420 radio and utilizes them without any modification of the WHART TDMA standard. This approach is independent of the underlying link scheduling algorithm, but a node uses its given schedule to determine the right sleep states. Since many other modern radios do also have multiple sleep states with the same type of trade-off between power consumption at sleep time and wakeup time, we believe that the general approach of DM3S is transferable to other such radios as well.

Generally speaking, in WHART nodes activities are constrained to certain slots (whether these are exclusive or shared does not matter for the following presentation), whereas in all other slots they can sleep. We call the slots that a node might be involved in its **active slots**. There will generally be some active slots in which a node will have to wake up unconditionally, for example those slots in which the node is scheduled to receive, or those transmit slots where a frame is transmitted the first time. On the other hand, retransmission slots are only used when a transmission in a previous transmit slot has failed (i.e. the sender has not received an acknowledgement). A key observation is that at the end of a transmit slot the sender will know if it has to utilize a retransmission slot or not. More generally, based on its schedule and the transmission outcomes in the current active slot, at the end of the current slot a node can determine how much time will elapse before its next active slot starts.

The second key ingredient is borrowed from a technique used in dynamic power management to control the device's operational states, see [78, 15]. Specifically, since the number of transceiver states and their switching time is known a-priori, it is possible to construct a function $\phi(\cdot)$, which takes a non-negative time duration τ as a parameter and which returns a sleeping schedule that:

(i) ensures that after τ seconds the node transceiver is ready to transmit or receive, (ii) sends the transceiver through a "monotone" sequence of sleep states (the deepest state at the beginning and the lightest state at the end), and that (iii) ensures that the chosen sequence of states (and the times being spent in each visited state) has the smallest energy consumption over the time horizon of τ seconds.

For the CC2420 transceiver this function $\phi(\cdot)$ is straightforward to construct. Specifically, we need to determine three threshold values:

(i) a duration τ_1 that is minimally needed to make sleep-state-1 more energy-efficient than to stay awake; (ii) a duration $\tau_2 > \tau_1$ that is minimally needed to make an initial choice of sleep-state-2, followed by a transition through sleep-state-1 and subsequent wakeup more energy-efficient than to start initially with sleep-state-1; and (iii) a duration $\tau_3 > \tau_2$ that is minimally needed to make an initial choice of sleep-state-3, followed by a transition through sleep-state-2, sleep-state-1 and subsequent wakeup more energy-efficient than to initially start with sleep-state-2.

When at the end of an active slot it takes a time τ before the next active slot starts, it is a simple matter of comparing τ to the three thresholds τ_1, τ_2 and τ_3 to figure out which sleep state (if any) should be entered next.

6.3.2 Evaluation and Results for the DM3S Approach

In order to evaluate the efficiency of the DM3S approach we perform simulations using the same setup as described in Section 6.1.1. We first conduct a regression analysis similar to the one in Section 6.2, then we provide a breakdown of the average energy consumption based on the hop distance of nodes to the gateway.

The results of the regression analysis when DM3S is used are shown in Table 6.6. Specifically, this table shows the contribution of the individual factors and their pairwise combinations to the variation of total energy-consumption. From this analysis it can be seen that the impact of factor D (Sleep power) has been reduced drastically, this factor now accounts for only $\approx 1\%$ instead of 35% of the total variation in energy consumption. We can also observe that now factors F (Management rate) and C (Listen power) contribute most to the total energy-consumption. Please note that again most of the quadratic factors have negligible weights (see our technical report [95] for complete tables of all the factors). The analysis of variance (ANOVA) analysis (not shown here, but in [95], together with the results of the other statistical tests) confirms that, similar to the case without DM3S, all the elementary factors A to F and the compound factor (CF) are highly significant, together they explain almost all the variation. Based on the ANOVA test we have simplified the regression model using the least-squares estimation method by excluding insignificant factors to become:

$$\begin{aligned} T_E = {} & 68.98 + 2.37A + 2.61B + 11.72C + 1.84D \\ & + 0.49E - 13.21F - 3.44CF \end{aligned} \tag{6.3}$$

Please note that in comparison to Equation 6.2 the intercept term has been reduced from 111.33 to 68.98 , and the coefficient for factor D (sleeping energy)

has reduced from 13.77 to 1.84.

Table 6.6: The percentage of factors contribution for DM3S.

Term	Sum of Squares	Percentage contribution
A	358.72	1.65
B	436.15	2.00
C	**8796.50**	**40.38**
D	216.85	1.00
E	15.14	0.07
F	**11167.52**	**51.27**
AB	0.05	$2.319 \cdot 10^{-4}$
AC	0.91	$4.177 \cdot 10^{-3}$
AD	1.56	0.01
AE	0.91	$4.176 \cdot 10^{-3}$
AF	1.81	$8.311 \cdot 10^{-3}$
BC	0.91	$4.177 \cdot 10^{-3}$
BD	1.56	0.01
BE	0.91	$4.176 \cdot 10^{-3}$
BF	2.26	$1.039 \cdot 10^{-2}$
CD	0.01	$2.398 \cdot 10^{-5}$
CE	0.05	$2.322 \cdot 10^{-4}$
CF	759.01	3.48
DE	0.91	$4.175 \cdot 10^{-3}$
DF	0.10	$4.724 \cdot 10^{-4}$
EF	1.96	$8.976 \cdot 10^{-3}$
ABC	0.91	$4.176 \cdot 10^{-3}$
ABD	1.56	0.01

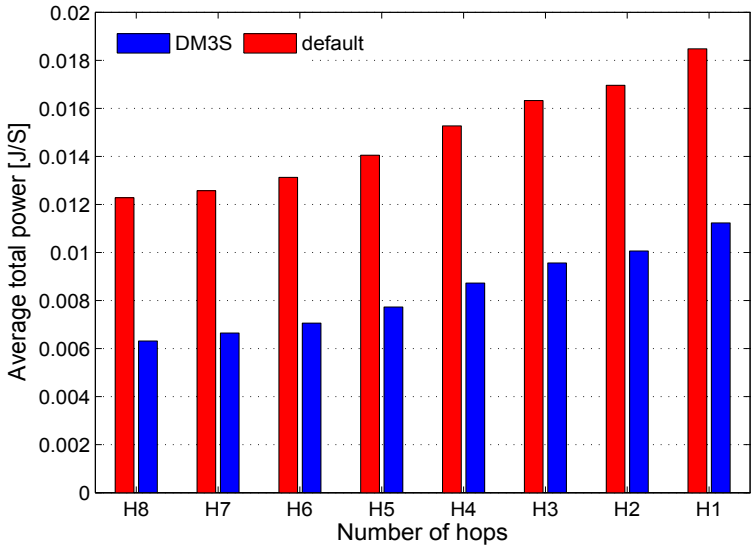


Figure 6.8: Average energy-consumption between default mode and DM3S mode for 1s rate

Moreover, in Figure 6.8 we show the average energy-consumption of each hop for both normal operation and DM3S approach in a scenario with ten sources generating traffic with a period of one second. We can see that the multiple sleep state scheduling leads to significantly lower energy consumption than the operation with just sleeping-mode-1. Similar trends are observed also for scenarios with 30 and 60 seconds traffic generation period.

6.3.3 Model Validation

A real-world experiments have been used to validate our simulation models. We compared the results against those from the real-world experiment to assess the validity of the underlying simulation models. In particular, The following procedures have been performed:

- We run the simulation under a variety of settings of the input parameters, and check to see that the output is reasonable accurate.

- We also used the most powerful techniques for validation the simulation through the use of deterministic input scheduling data and compute its operations with a hand calculations to see if the simulation is operating as intended. This includes, superframe and time slot lengths, switching time within the slots, etc.

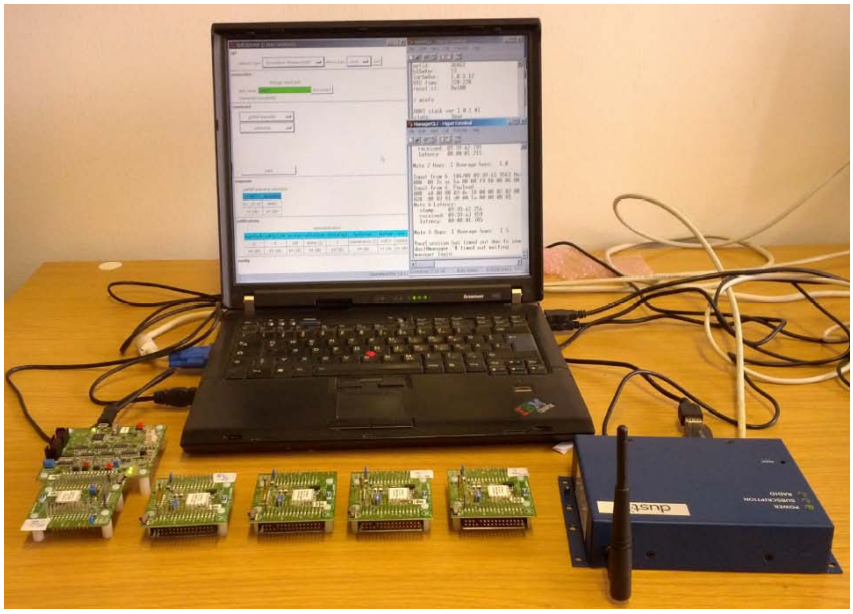


Figure 6.9: WirelessHART experiment setup

- In addition, we conduct real-world measurements to validate the simulation models. Specifically, we bought DUST WirelessHART evaluation Kit from Linear technology. The WirelessHART Kit includes the following hardware: five evaluation and development Motes (Eterna WirelessHART sensor nodes), one Eterna interface card and WirelessHART network manager (see Figure 6.9).

6.3.4 Experimental Setup

A schematic of the WirelessHART validation scenario are shown in Figures 6.9 and 6.10 for real hardware and simulation, respectively. We assume a single source, 4 forwarders, and one gateway. The forwarder nodes are arranged in linear topology. The sink/GW node connected to the last forwarding nodes as depicted in Figures 6.9. The source periodically generates packets up to 133 bytes in total size, The generation period was varied, ranging from 1 to 60 seconds. We change the traffic periodicity by varying the inter arrival period of the generated sample by 5 seconds.

Within one run we generate 10000 packets. MAC-layer acknowledgments are enabled and the size of the ACK packet is 26 bytes. If the packet is lost due to channel errors the sender tries to transmit the packet for a maximum of two retries.

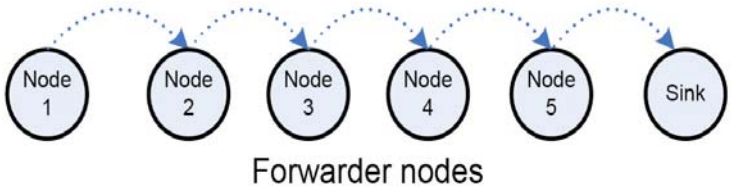


Figure 6.10: WirelessHART simulation setup

6.4 Validation Results

We first compare the results from the real experiment which conducted in a real WirelessHART kit with the simulation results. By validation we mean we want to find our whether the simulation can provide us enough confidence in the claims we made of the packet delay and packet reception rate. Figure 6.11 shows the packet delay for both experiment and simulation. The simulation result is closely match the one from the WirleessHART measurements (compare Figure 6.11.

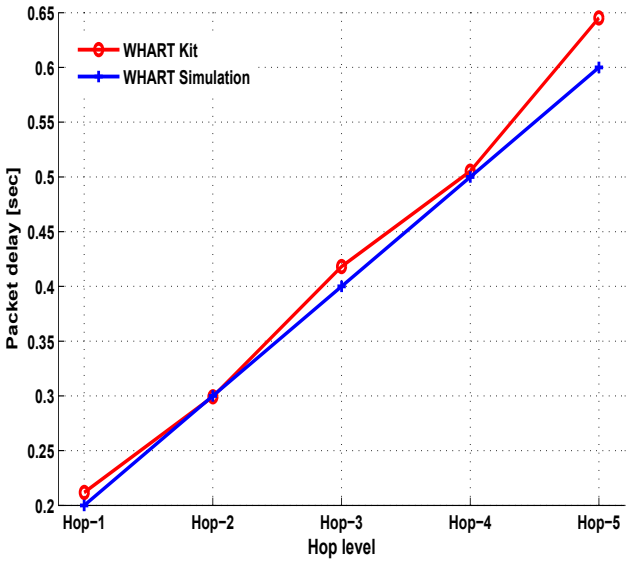


Figure 6.11: Packet delay: WHART kit vs. WHART simulation

We also compare the packet reception ratio. Figure 6.12 shows the results from both simulation and real measurement. The simulation models accurately reflect the actual system (Compare Figure 6.12 and Figure 6.14). The average

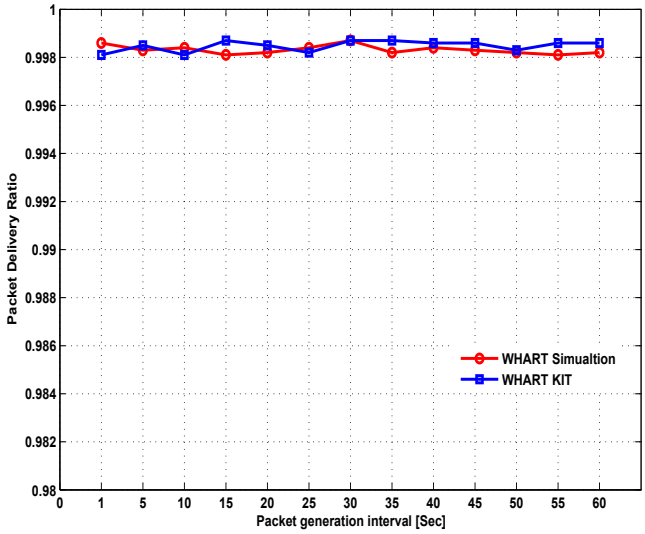


Figure 6.12: Packet delivery ratio validation: simulation and experiment

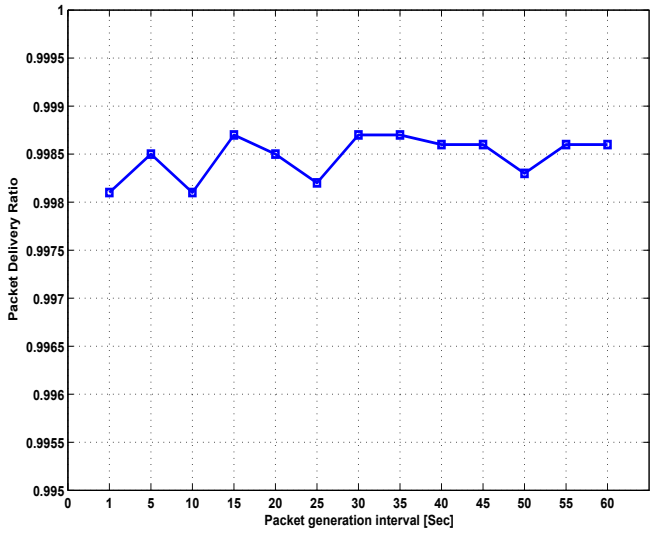


Figure 6.13: Packet delivery ratio wirelessHART kit experiment

variation between the real experiments and the simulation results is less than 4%.

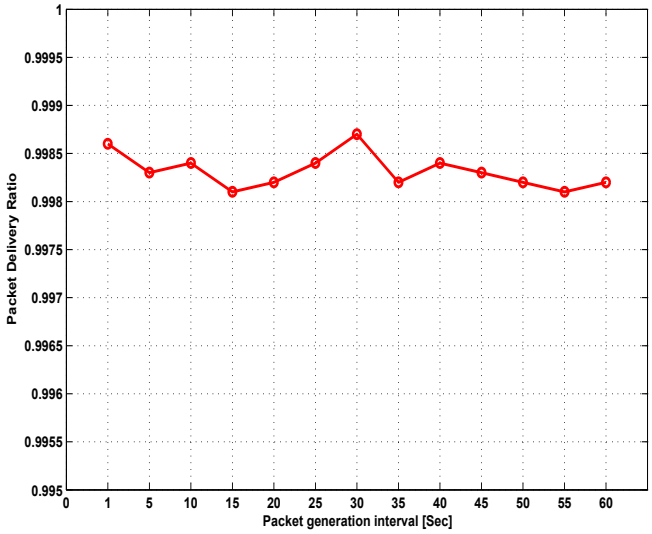


Figure 6.14: Packet delivery ratio wirelessHART simulation

6.5 Summary

In this chapter we have presented a detailed evaluation of the energy consumption of the WirelessHART protocol using realistic simulation.

- We have performed a sensitivity analysis using the response surface methodology to obtain some insights on how the overall energy consumption breaks down into different factors. By identifying the factors contributing most to the overall network energy consumption, one can obtain useful insights on where to start with any effort geared towards saving energy.

- We have evaluated the impact of synchronization and management packets including: advertisement, join request/response, commands, keep-alive and health report control frames on the performance of WHART TDMA protocol.

- We have also analysis the impact of the traffic density and the number of hops on the performance of the WirelessHART solution in terms of energy.

- We also adapted and evaluated an energy management policy (used in our autonomous framework) for TDMA-based protocols by exploiting

the available sleep modes of the transceiver. Our result shows significant savings can be achieved when using our DM3S energy management policy.

- Further, we validate our simulation models of the WirelessHART system using WirelessHART kit. We conducted a real-world measurements and then compare the simulation results with the results obtained from the real-world measurements.

CHAPTER 7

Comparison Study: Autonomous Framework versus WirelessHART System

It is a long-standing hypothesis in the realm of wireless sensor networks that TDMA protocols, despite their advantages in scheduling the sleeping activities of nodes, have associated costs (e.g. time synchronization, slot allocation schemes, slot re-allocation after network failures or change of topology) that are too high in terms of energy and delay for multi-hop networks. However, it is hard to find numbers comparing a full TDMA-based system with non-TDMA solutions. This chapter aims to scrutinize this hypothesis and to provide a balanced answer on the advantages and disadvantages of our developed solution (autonomous framework) and the WirelessHART solution for supporting periodic traffic in WSNs.

7.1 Methodology and Setting

The performance of both systems is evaluated using a combination of both real experiment traces and simulation. We believe that this is the most appropriate methodology for such evaluation. This is due to the fact that using only theoretical channel models usually does not capture complex phenomena such as multi-path fading, or the impact of a dynamic environment. On the other hand, real-world experiments do not provide the ability to evaluate different schemes or algorithms under repeatable conditions, as the RF environment is time-varying.

For the traces, both autonomous framework and WirelessHART system are evaluated using connectivity traces gathered in a real-world deployment. This

Table 7.1: Main autonomous framework parameters.

Parameter	Value
Frame size	128
Acknowledgment frame size	12
Channel switching time	192 μs
Length of learning phase	5 packets
Allowable packet loss rate α	2
Loss threshold	3 packets

Table 7.2: Main WirelessHART system parameters

Main Radio and MAC parameters	
Parameter	Value
Data frame size	128
Acknowledgment frame size	26
Channel switching time	192 μs
Number of slots per superframe	1024
Slot-time length	10ms
Synchronization frame size	26 Bytes
Re-synchronization rate	60s
Health report rate	15min

was already explained in Chapter 3 Section 3.3.2

7.1.1 Simulation Setup

In order to have a fair and consistent comparison study between autonomous framework and the sate-of-the-art solution WirelessHART. We have used the same simulation environment which was covered in Section 3.3.3.

We also used the same power consumption parameters of the CC2420 transceiver and of the MSP430 micro-controller for both autonomous and WirelessHART system (see Table 3.1).

The system-specific parameters are listed in Table 7.1 and Table 7.2 for both autonomous framework and WirelessHART, respectively.

Unlike the setting conducted in Chapter 5, in this evaluation we enable the dynamic multiple sleep states scheduling for both systems.

7.1.2 Network Topology and Traffic

We have generated 150 random topologies and for each setting of simulation parameters we correspondingly performed 150 replications. For each random topology we have placed 45 nodes in an area of size 225 × 225 feet, using uniform distribution for node positions. The sink is placed in the upper right corner of the nodes (see Figure 3.3). Out of the 45 nodes we randomly pick five nodes as source nodes. Each of these sources periodically generates packets with a payload of 80 Bytes (not including PHY and MAC overhead). The sources are chosen such that the path length is 4 to 8 hops. Any path less than 3 hops is discarded and not considered in the evaluation. Please note that for all our evaluation we only consider the nodes that are involved in the forwarding duties and all the other nodes are excluded from the evaluation for both our autonomous framework and the WirelessHART. Unless otherwise specified, all the sources transmit with the same period, however, the starting phase is set randomly. The generation period was varied, ranging from 1 to 60 seconds. We change the traffic periodicity by varying the inter arrival period of the generated sample by 5 seconds step. During each simulation run, each source transmits packets based on its periodicity and then forwards these packets to the sink node via some forwarders. MAC-layer acknowledgments are enabled in both systems and the size of the ACK packet is 12 Bytes for the autonomous framework and 26 Bytes for the WirelessHART. If the packet is lost then the sender tries to transmit the packet for a maximum of two retries (unless otherwise specified).

7.1.3 Major Performance Measure

The simulation time is fixed to 168 hours (one week) and the three main performance metrics are the total energy spent by the radio transceiver of a node over this period, the end-to-end packet delivery ratio (PDR), i.e. the fraction of all packets sent by the sources that reach the destination. The third important performance metric is the end-to-end packet delay. The simulation records the amount of time spent in various states (transmit, receive, listen, sleep and turnover) and calculates from this the total energy consumption of a node over a span of 168 simulated hours. We also take into consideration the energy consumed by the node's micro-controller using the same method described in Section 3.3.5.

At the end of each run, the simulator computes the total energy consumed for all nodes in the network using the amount of energy consumed by the radio and micro controller in each state. We also measure the time a forwarder node requires in order to detect and adapt to changes in traffic load requirements for both systems.

7.2 Comparison Results

In order to study the performance of our autonomous framework with a WirelessHART system, we compare both systems under the same settings except for system-specific parameters. Unless otherwise specified, we use the default values for these system-specific parameters as listed in Table 7.1 and Table 7.2 for both autonomous and WirelessHART systems, respectively. We first investigate the total energy consumption vs the traffic generation period, hop level, number of flows on the performance of both systems. We then compare the end-to-end packet delay and packet delivery ratio for both approaches. We also investigate the impact of the traffic change on the energy consumption, packet delay and packet delivery ratio for both systems. Furthermore, we evaluate both systems under different operation scenarios as explained in the following sections.

7.2.1 Impact of Data Reporting Rate on Power Consumption

Figure 7.1 shows the impact of the energy consumption vs the packet generation interval in WirelessHART system. We change the traffic periodicity by varying the inter arrival period of the generated sample by 5 seconds step. Please note that we use the default parameters for both systems. The number of flows are set to five.

This figure shows that, deceasing the packet generation interval from 1 to 5 seconds per packet may save roughly more than the half of the energy, and therefore, the lifetime might be extended almost by factor of 2. This is not true for low traffic period and most importantly there is almost no gain after 35 second generation period. On other words, using WielessHART system one can't really gain extra energy by generating and transmitting packets each 60 second instead of 35 second period. This is due to the control packets that have to be transmitted frequently. Therefore, the minimum energy consumption is bounded by the periodicity of the control packets which is 30 seconds in the WirlessHART standard.

On the other hand, Figure 7.2 shows the average energy consumption for the autonomous framework over 8 hops uses the same scenario as the WirelessHART.

We can observe from the figure that the energy consumption of the autonomous system achieves highly superior performance compared to the WirelessHART solution. The energy consumption of the WirelessHART is significantly higher compared to the autonomous framework. This is due to the high number of control packets (such as: synchronization, advertisement, joint request/response, commands, keep alive and health report) that are essential

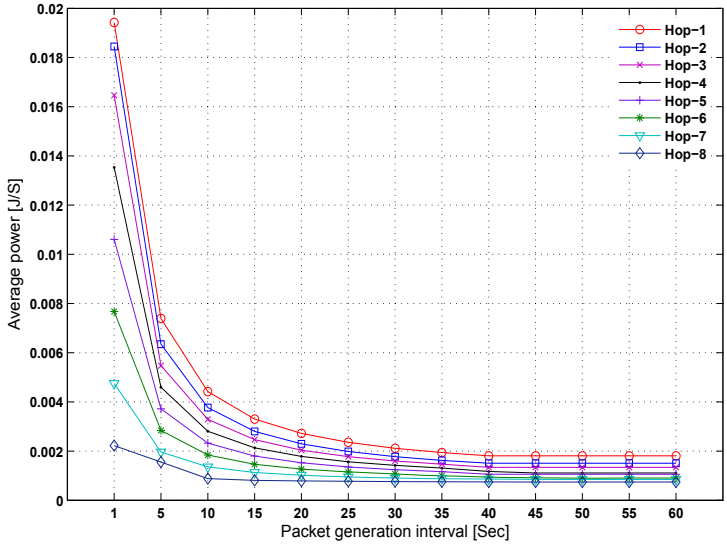


Figure 7.1: Average energy consumption vs. packet generation interval for WirelessHART

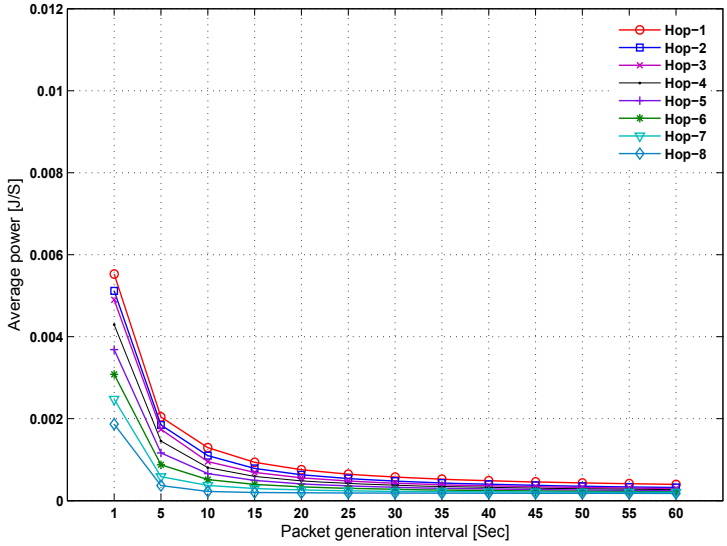


Figure 7.2: Average energy consumption vs. packet generation interval for autonomous approach

for such solution. For the autonomous framework, it is important to observe

from Figure 7.2 that building a network that has a reasonably uniform battery life time and with a very flat energy consumption is possible. Thus, our proposed approach achieves a uniform battery lifetime across the whole nodes when moderate or low traffic generation loads are present.

7.2.2 Impact of the Number of Flows on The Energy Consumption

For many applications all batteries should be replaced at the moment the first sensor node dies (for example, due its battery drain). This means that the first hop (the closest node(s) to the sink side) should be considered as an indicator for replacing the batteries, as its responsible for forwarding all the traffic flows on behalf of other nodes toward the sink node.

Figures 7.3 and 7.4 show the impact of the number of flows on the energy consumption of the first hop for both autonomous framework and WirelessHART, respectively. We also plot the last hop (the closest node(s) to the sources) as well.

For all figures, the energy consumption increases with the number of flows, presumably due to the extra traffic load.

One main observation of these experiments is that the different in energy consumption across the flows in WirelessHART system is almost increased equally between the flows (see Figure 7.4). This might be due to the fact that WirelessHART allows only one packet to be transmitted at each time slot. On other words, if the buffer has more than one packet and the time slot is turned up, then only one packet is allowed to be transmitted and the node should wait for the next round in order to transmit one more packet (the next buffered packet). Consequently the node can not enter a deep sleep mode unless the buffer is empty, so a node stays at a light sleep mode which consume more energy than the deep sleep mode.

The energy consumption between the flows in the autonomous framework is more or less close to each other, especially after the second flow. Therefore, the amount of accumulated energy consumed by the different flows is not that significant compared to the WirelessHART case (compare Figure 7.3 and 7.4). This demonstrates the efficiency of our multi-flow sleep/wakeup scheduling algorithm that utilizes the wake-up window in order to transmit all the buffered packet in one go.

Furthermore, Figure 7.5 and Figure 7.6 show side by side the performance of the energy consumption vs the generation period and the number of flows for both systems in the first and last hops, respectively.

Both figures show that our autonomous framework performs much better than the WirelessHART system for all the considered traffic rates. This is not only due to the control packets that used in the WirelessHART but also

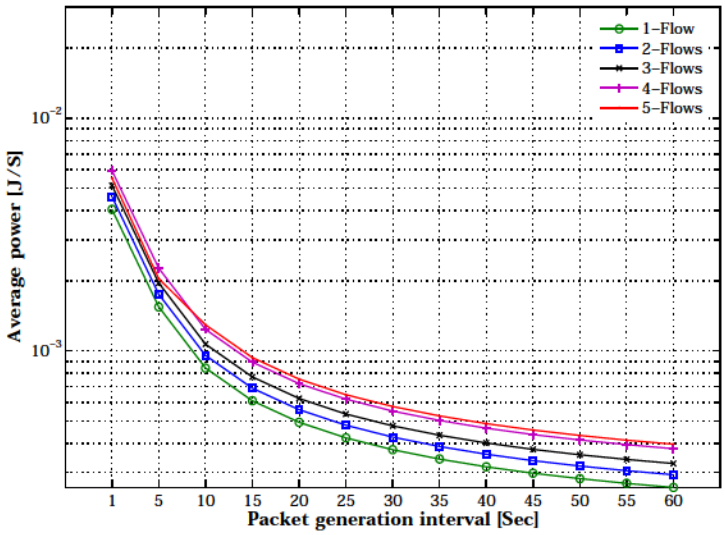


Figure 7.3: Average energy consumption vs. number of flows for autonomous approach

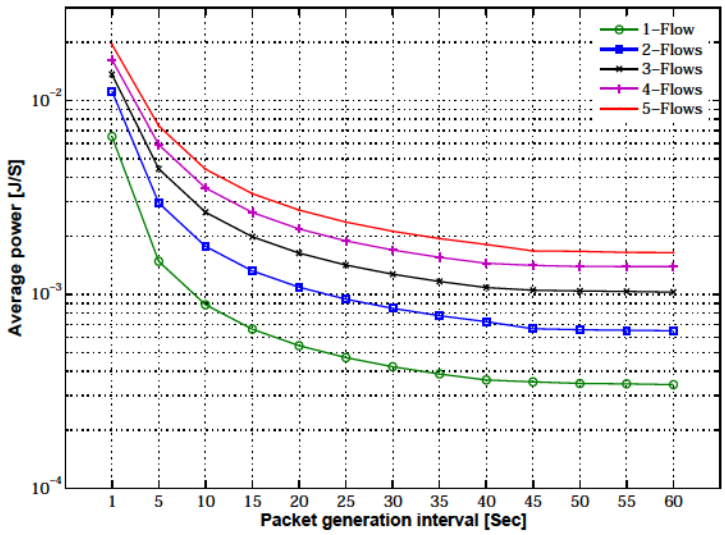


Figure 7.4: Average energy consumption vs. number of flows for WirelessHART approach

because of the smaller acknowledgment frame size used in our approach which is 12 Bytes instead of 26 Bytes used in the WirelessHART system.

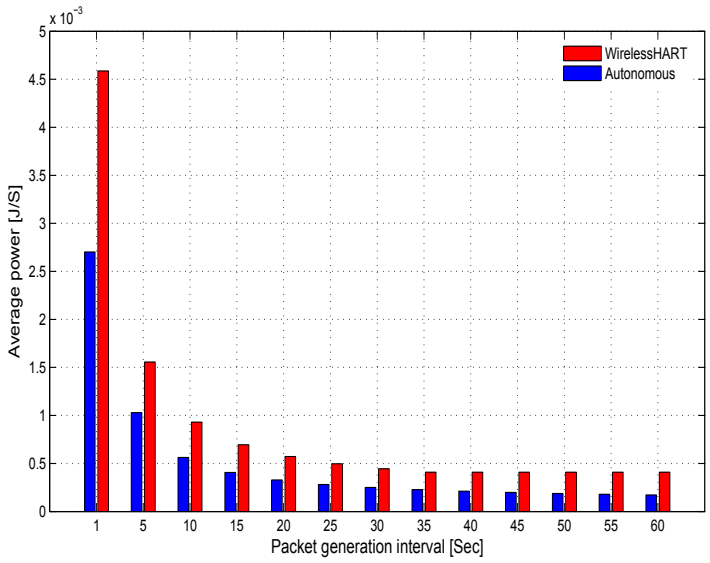


Figure 7.5: Average energy consumption for both systems for one flow: first hop

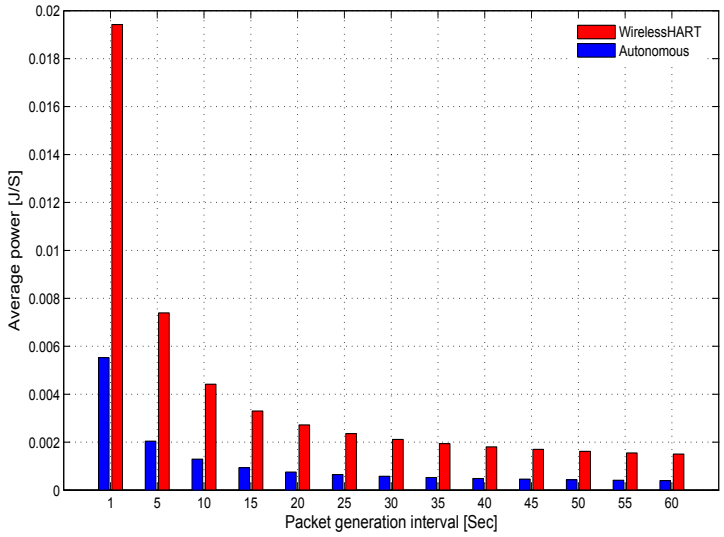


Figure 7.6: Average energy consumption for both systems for five flow: first hop

7.2.3 End-to-End Packet Delay

One important design goal of the autonomous framework is not only to minimize the energy consumption and improve the communication reliability, but

also to minimize the end-to-end packet delay. To achieve this goal we build our adaptive and online algorithms in such way that the forwarder node transmits its incoming packet immediately (if the channel is sensed idle). Thus, the forwarder node does not keep a packet in its buffer unless its associated acknowledgment is lost. Please note that the forwarder node updates its estimation parameters per-packet basis, only if the incoming packets are in sequence, therefore the schedule of the node activities is done in a consistent manner. Figure 7.7 shows the box plot for the end-to-end packet delay at each hop towards the sink for the autonomous framework.

The packet delay increases linearly with the number of hops and the autonomous framework achieves highly superior performance compared to the WirelessHART system (compare 7.7 and 7.8).

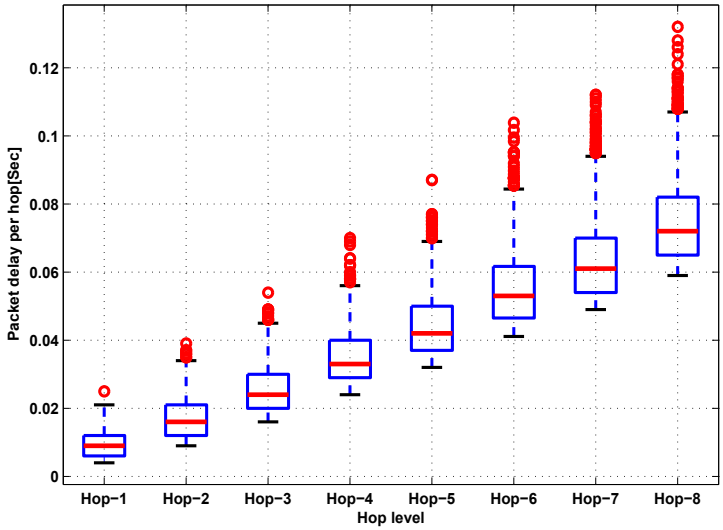


Figure 7.7: End-to-end packet delay for autonomous framework

The delay for autonomous framework is significantly lower compared to the WirelessHART because the node aligns its wake-up window and forwards the packet immediately once a packet is received and the channel is free. In order to transmit a packet in the WirelessHART system, the node has to wait for its scheduled time slot to turn up. This contributes to the high delay in the centralized solution. Moreover, WirelessHART requires additional slots to be assigned not only for control packets but also for retransmission; this again contributes to higher end-to-end delay. Unfortunately, if the node successfully transmits its packet, then all the assigned slots for retreansmission are wasted and can't be reused to forward further packets (if packets are available at that

instance of time). Consequently, the node has to delay its transmission for the next superframe. Thus, the slots for retransmssion are not utilized and this is also contributing to additional packet delay as shown in Figure 7.8. Please note that, WirelessHART can improve the end-to-end delay for particular flow when the assignment of time slots is done in a sequential manner. For instance, the network manager may try to assign the control packet and the retransmission slots as far as possible from data packets. For the autonomous framework one can even improve the end-to-end delay by reducing the back-off contention interval. This is more useful in the case of moderate and low traffic rate.

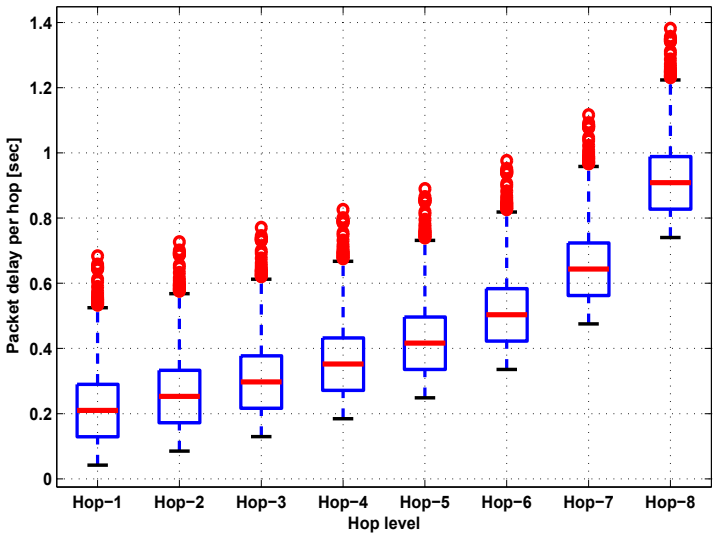


Figure 7.8: End-to-end packet delay for WirelessHART system

Please note that in a box-plot, the box has lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the box to show the extent of the rest of the data. Outliers are data with values beyond the ends of the whiskers.

7.2.4 Packet Delivery Ratio

In order to evaluate the packet delivery ratio, we run the same connectivity traces on both approaches and then observe the end-to-end packet delivery ratio. We vary the number of flows to analyze the impact of number of flows on the packet delivery ratio for both approaches (see Figure 7.9). WirelessHART achieves slightly better results compared to our autonomous framework, when the load is high. This is due to the deterministic time slot approach, in which

the schedule is computed in advance so internal collision is avoided. Please note that in this setting we use only one retry for both approaches. Another important observation is that in moderate and low data rate, the packet delivery ratio using our autonomous framework achieves almost the same performance of packet delivery ratio as the WirelessHART system. This confirms that, the higher sampling rate is, the higher the probability of collision. Therefore, WirelessHART performs better in such scenarios. However, because the difference in packet delivery ratio result is not that significantly high, thus increasing the number of retransmissions is a good idea to improve the packet delivery ratio of the autonomous framework.

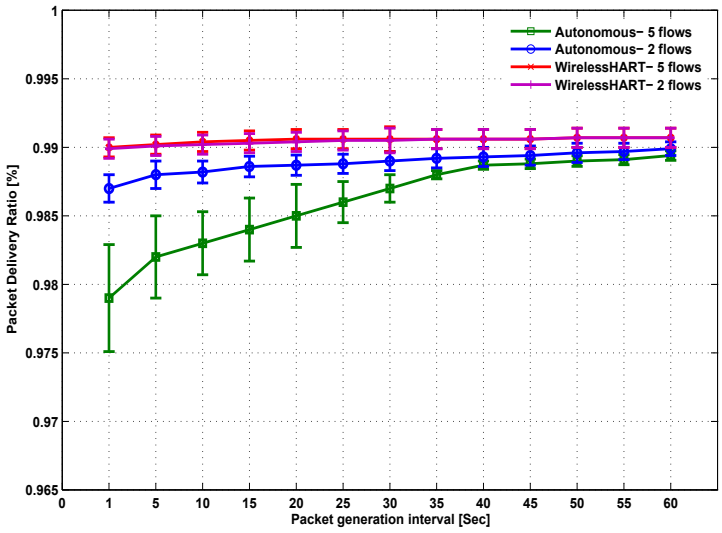


Figure 7.9: Packet delivery ratio between WirelessHART and autonomous for 1 re-transmission

7.2.5 Impact of Control Packet on the Performance

In this section we compare both systems at a setting when both WirelessHART and the autonomous systems are operating in the steady state only. More precisely, once the network is established, we disable all the control packets and therefore, there is no change in the traffic load nor on the environment. This setting is not realistic, as in a real-world scenarios the control packets are essential in order to reflect the changes in the topology, traffics, channels and etc. However, the following results demonstrate the impact of the control packets on energy consumption, per-hop packet delay and packet delivery ratio.

Impact of control packet on per-hop delay

We also show the per-hop packet delay for both systems in case of excluding the control packets. For WirelessHART the per-hop packet delay reduced to more than the half, if compared to the normal mode (when control packets are enabled) as there are no time slots to be assigned for the control packets. Thus, the network manager can assign the time slots without additional constrains of the control packets. According to the WirelessHART standard, the control packets have priority over the data packets, and therefore, they have to be assigned and activated first. However, there is no significant change in the per-hop packet delay in case of the autonomous framework as there is more or less no control packet associated with the autonomous framework. To sum it up, the autonomous framework achieves a highly superior performance compared to the WirelessHART in terms of end-to-end packet delay.

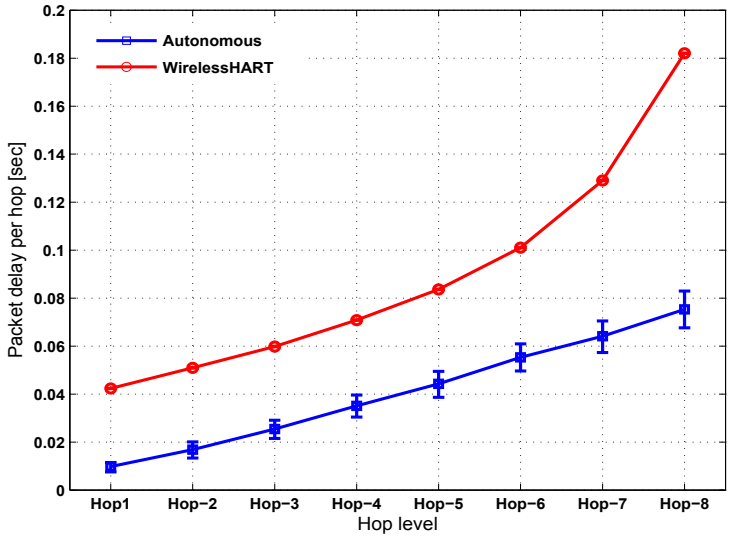


Figure 7.10: Per-hop packet delay for both WirelessHART and autonomous systems in case of no control packet

Impact of control packet on packet delivery ratio

The end-to-end packet delivery ratio under the same setting is evaluated for both systems. Figure 7.11 shows the packet delivery ratio for both systems in static topology and with fixed traffic load.

The result shows that both protocols work pretty well in such scenarios. However, WirelessHART might achieve better performance when traffic load

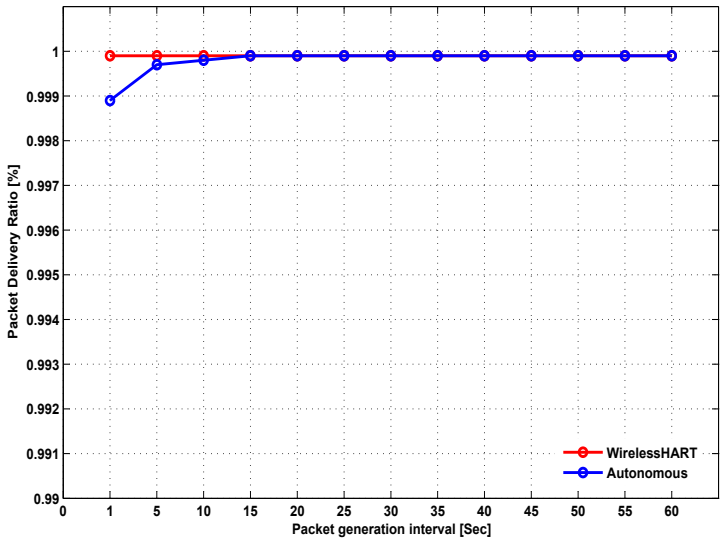


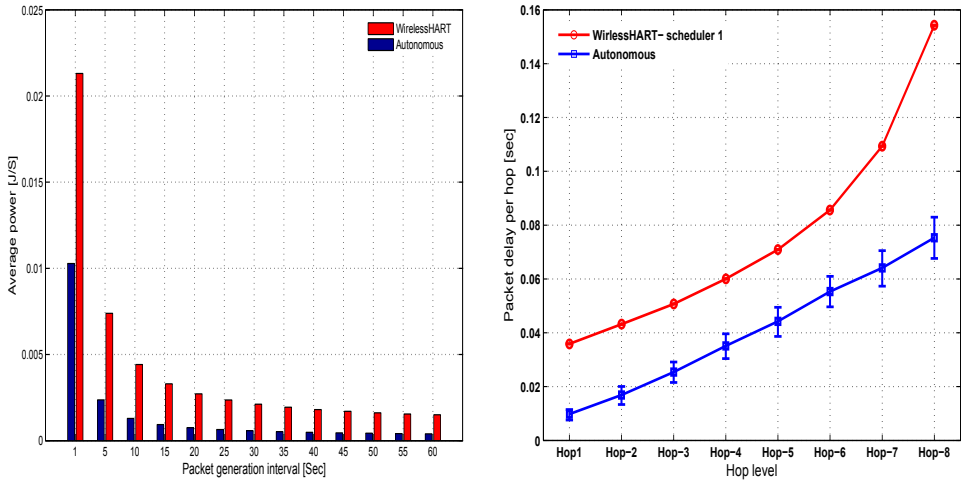
Figure 7.11: Packet delivery ratio between WirelessHART and autonomous systems in case of no control packet

is really high, which is usually not the case in many WSNs applications.

7.2.6 Impact of the Type of Slot Assignment Algorithm on the Performance

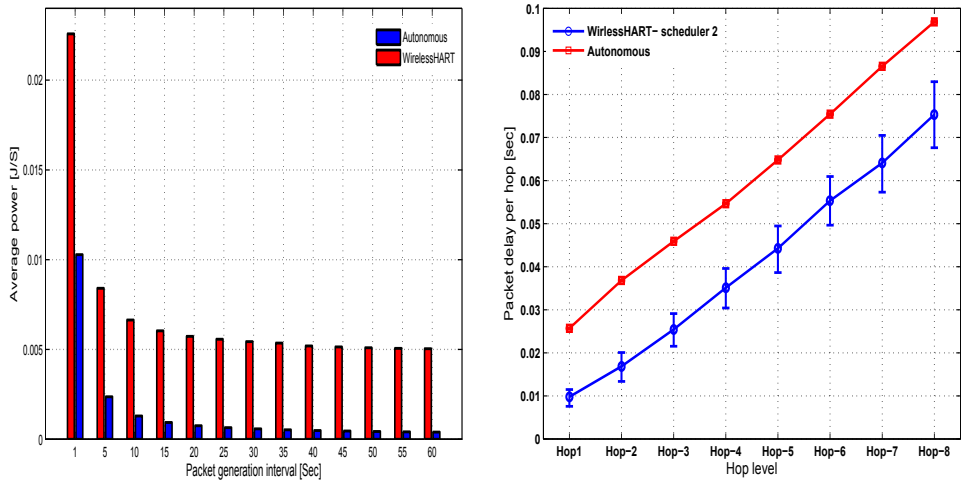
In this section we evaluate the impact of the slot assignment algorithms on the performance of the WirleessHART. We use two different slot assignment algorithms, the first is explained in [61] and the second is based on the breadth first assignment and explained in Appendix B.1.3. Please note that, since the autonomous framework relies in a decentralized way, there is no change in its configuration during these evaluations. For both scenarios we fix the traffic periodicity to 10 seconds. Figure 7.12(a) and Figure 7.12(b) show the energy consumption and per-hop packet delay under the first slot assignment algorithm.

Figure 7.13(a) and Figure 7.13(b) show how the WirelessHART system performs under the second slot assignment algorithm. Comparing these figures one can observe that the selected type of slot assignment algorithm has a great impact on the energy consumption and per-hop packet delay. It seems that if the delay is important then one has to select the second slot assignment algorithm. Otherwise, the first slot assignment performs better, especially if the energy conservation is critical and more important than the packet delay.



(a) Average energy consumption (b) Per-hop packet delay

Figure 7.12: impact of type of slot assignment algorithm in the performance of the WirlessHART: first schedule



(a) Average energy consumption (b) Per-hop packet delay

Figure 7.13: impact of type of slot assignment algorithm in the performance of the WirlessHART: second schedule

So optimizing both energy consumption and packet delay is a challenging task in such centralized protocols. Please note that, all the other hops level have similar results as the first hop.

7.2.7 Impact of Traffic Changes on The Performance

Depending on the underlying application, source nodes may have different traffic requirements which could change over time. In centralized based protocols this takes long time as a node should first wait for control slot that is reserved in advance for such circumstances, then acquires the new traffic requirements and finally waits for new schedule table update from the network manager, which could be several hops away from that particular source. Thus, a source node has to wait for all that time events in order to adapt to the new traffic requirements.

As explained in Section 5.3, we proposed on-the-fly traffic adaptation mechanism that relies on local information to allow the forwarder react to traffic requirements in agile manner. Our traffic adaptation mechanism allows the source nodes to increase, or decrease, its generation sampling rate based on their sensing requirements by setting one bit flag. If the flag is set to one then the forwarders knows that it has to enter the learning phase and starts to adapt to the new traffic load immediately.

In this section we demonstrate the efficiency of our approach versus the classical approach used in WirelessHART system. We run the same setup in both systems and compute the time it takes for the first forwarding node to react to the traffic change. Please note that each source node randomly selects its starting period, then it selects another periodicity randomly after transmitting about 1000 packets. We also evaluate the impact of the traffic change under different control signaling rate. This control signaling is responsible for maintaining the adaptability of the WirelessHART system. Table 7.3 lists the average required time for the first forwarder in order to adapt to the new traffic requirements for both WirelessHART (using slot assignment algorithm one) and autonomous framework. Please note that for this table (Table 7.3) we configure the network manager to assign control packets every 60 seconds interval (60s is the interval between two control events).

Table 7.3 illustrates how fast our approach reacts to the traffic change compared to the WirelessHART system. The average convergence time of the new traffic is almost a function of the current periodicity. So the higher the generation time, the faster is the convergence time to the traffic change. For the WirelessHART, it takes significantly long time to adapt to the new requirements compared to the autonomous framework. This is due to the centralized nature of the WirelessHART, as a forwarder can not change its current slot schedule locally but has to wait for an update from the centralized authority.

Table 7.4 shows the average required time under 30 second control signaling rate. In other words the network manager assigns a control time slot for such traffic every 30 second interval. The node could use this time slot to notify the

network manager via a set of forwarding nodes when new traffic requirement is observed.

When comparing Table 7.3 with Table 7.4 we can see the impact of control packet rate: by increasing the control packet rate, the adaptability to traffic changes may increase as well. However, this is at the cost of energy consumption and packet delay due to the periodic control packets. For all the scenarios the autonomous framework achieves highly superior performance compared to the WirelessHART system in terms of adaptability to varying traffic.

7.3 Sensitivity Analysis Comparison

In the previous section we have analyzed the impact of energy consumption over different settings (such as number of flows, sampling rate, generation period, etc.) on the performance of both autonomous framework and WirelessHART systems.

In this section we perform sensitivity analysis in order to analyze how much the energy consumption (taken over 168 hours, see Section 7.1.3) is influenced by various factors, including important physical layer parameters for both systems. Identifying the factors contributing most to the overall energy consumption can provide useful guidance for further optimization of both our framework and the benchmark protocol.

Please refer to 5.7.1 for an introduction of the methodology used in this section. Mainly we use the Response Surface Methodology (RSM) [144, 123, 119] which a statistical tool that is frequently applied to analyze systems in which a response of interest is influenced by several variables, called factors.

7.3.1 Factor Screening

The first step in the RSM is to identify potential factors affecting the response being measured (factor screening). Since the average total energy-consumption is the main response, we consider the following factors:

- Factor A – Transmission power: the transmission power is the power consumed for transmitting data frames and control frames such as synchronization frames.

- Factor B – Reception power: the receiving power is the power consumed while receiving data or control frames.

- Factor C – Listening power: the listening power is the radio power consumption when the radio is on but not receiving or sending any frames.

Table 7.3: adaptability to the traffic periodicity under 60 second control packet rate

Change of period	WirelessHART Average adaptive time	Autonomous Average adaptive time
From 1s to 2s	$\approx 42.31 minutes$	$\approx 1.53 second$
From 2s to 5s	$\approx 37.20 minutes$	$\approx 2.09 seconds$
From 10s to 30s	$\approx 32.32 minutes$	$\approx 10.34 seconds$
From 30s to 1s	$\approx 25.60 minutes$	$\approx 30.13 seconds$
From 60s to 2s	$\approx 20.20 minutes$	$\approx 60.12 seconds$

Table 7.4: adaptability to traffic periodicity under 30 second control packet rate

Change of period	WirelessHART Average adaptive time	Autonomous Average adaptive time
From 1s to 2s	$\approx 29.60 minutes$	$\approx 1.5 second$
From 2s to 5s	$\approx 26.41 minutes$	$\approx 2 seconds$
From 10s to 30s	$\approx 21.25 minutes$	$\approx 10.3 seconds$
From 30s to 1s	$\approx 17.80 minutes$	$\approx 30.3 seconds$
From 60s to 2s	$\approx 15.90 minutes$	$\approx 60.1 seconds$

Table 7.5: The factors and the levels of each factor.

Term	Factor	Level 1(-1)	Level 2(+1)
A	Tx power	32.67mW	57.42mW
B	Rx power	31.68mW	62.04mW
C	Listen power	31.68mW	62.04mW
D	Sleep power	0.72mW	1.41mW
E	Turnaround power	31mW	62mW

- Factor D – Sleeping power: the sleeping power is the power consumption while the radio is in the low-power state.

- Factor E – Turnaround power: the turnaround power is the power consumed while switching the radio state between different modes.

Table 7.5 lists the factors and the levels for each factor considered in our study.

7.3.2 Analysis of The Sensitivity Results

Table 7.6 shows the percentages which the individual factors and their pairwise combinations contribute to the variation of total energy-consumption over all 2^5 different factor combinations.

From the table we can observe that our autonomous framework provides significant energy saving compared to the WirelessHART system. Specifically, the autonomous framework can save about 25% in the Tx, Rx and listening operations. In addition the autonomous framework stays in the sleep mode about 25% longer compared to the WirelessHART system, thus being more energy-efficient.

7.4 Summary

In this chapter we compared our autonomous framework with the WirleessHART solution in terms of the energy consumption, packet delay and packet delivery ratio. We also compared the adaptability to varying traffic for both systems. Following are the most important conclusions:

- Our developed solution supports periodic traffic flows and frequency hopping without requiring an expensive protocol infrastructure providing

Table 7.6: The percentage of factors contribution for both systems

Term	Autonomous-System: Percentage contribution	WHART-System: Percentage contribution
A	1.01	1.68
B	1.20	2.01
C	**28.76**	**52.51**
D	**67.25**	**42.50**
E	0.13	0.17
AB	0.01	0.01
AC	0.05	0.05
AD	0.20	0.14
AE	0.05	0.05
BC	0.05	0.05
BD	0.20	0.14
BE	0.05	0.05
CD	0.12	0.05
ABC	0.05	0.05
ABD	0.20	0.14
ACD	0.12	0.05
BCD	0.12	0.05
ABCD	0.12	0.05

synchronization features (time synchronization, hopping synchronization) by relying entirely on the periodicity of the traffic itself for synchronization purposes.

- We have evaluated the proposed scheme in a range of scenarios using trace-based simulations, and we have shown that it, indeed, reaps the benefits of frequency hopping and also improves the energy consumption over centralized system such as WirelessHART.

- The results also show that the proposed schemes in the autonomous framework work at a very good level of reliability, and in addition it has very little implementation complexity.

- Further, we also designed and evaluated an efficient approach (On-the-

fly traffic adaptation mechanism) that allows a forwarder to react fast to the traffic change. In this mechanism source node exploits the currant traffic situation to notify its neighbor about the new traffic requirement (piggybacking on existing traffic) in a decentralized manner. When a source node wants to change its traffic data rate, it just sets the adaptive learning bit to 1, otherwise the bit is set to zero. The forwarder, upon receiving the packet, checks the bit to determine whether it has to enter new learning phase or stay the operational phase.

- We also provided a sensitivity analysis which shows how the energy consumption depends on certain parameters, including the power consumption of the transceiver in different modes of operation (Tx, Rx, Listening, Sleeping, and turnover). In this analysis we use the response surface methodology. These results show that our autonomous framework saves about 25% of the total energy consumption in the Tx, Rx and listening operations. Moreover our developed approach allows the node to stay in the sleeping mode about 25% more compares to the WirelessHART system.

CHAPTER 8

Conclusion and Outlook

In this thesis we considered a multi-hop sensor network with a significant share of periodic traffic, coming from different sources and at different reporting rates. A very important design goal is to run the network as energy-efficient as possible while supporting high reliability and low packet delay. The key motivation for developing this autonomous framework was based on the observations that the full TDMA operation including time synchronization, maintenance and schedule represent too much overhead for lightly loaded networks. Moreover, the adaptivity of traffic changes is crucial in such centralized solution as it requires communication schedules to be computed and distributed in advance. Therefore, it takes relatively longer time to adapt to such change in traffic loads. Nonetheless we wanted to support periodic transmissions and leverage frequency hopping.

In order to address these challenges, we developed a distributed and self-learning framework integrating an estimation and identification of the flows, asynchronous channel hopping, local dynamic multiple sleep state scheduling, On-the-fly traffic adaptation mechanism and an overlapping controller for periodic reporting applications in WSNs.

The following are the main important conclusions of the thesis:

- We proposed and evaluated novel channel hopping mechanism, which allows the nodes to switch between the available channels without relying in an explicit time synchronization protocol. Each forwarder node exploits both the flow period information and packet sequence number for selecting the next channel. This done with the help of a translation function which maps the packet sequence number and other parameters to a specific channel number. In our autonomous framework, there is no explicit time synchronization, but instead each forwarder learns the traffic period and jitter distribution from observing the traffic. Based on

this information a forwarder determines suitable times for sleeping and for waking up to receive the next packet.

- We designed and evaluated on-the-fly adaptive traffic mechanism, which achieves a highly superior performance compared to the WirelessHART system. The node relies on immediate updates whenever there is traffic change. These updates also make the solution robust, as traffic changes are immediately reflected in the forwarder node. Because updates are entirely piggybacked inside the packet (using single bit only), the solution is nearly overhead-free and thus energy-efficient.

- Moreover, we have looked at two different strategies for exploiting the sleep modes of the CC2420 transceiver and have highlighted that significant savings can be achieved with only moderate increases in run-time complexity. We then proposed a practical and effective energy management scheme which exploits the multiple sleep states of a transceiver and utilizes them in efficient manner. We also adapted this approach for TDMA-based protocols. This approach is independent of the underlying link scheduling algorithm, but a node uses it's given schedule to determine the right sleep states.

- In this thesis we also evaluated and benchmarked our proposed approach with the state-of-the-art solution WielessHART. WirelessHART combines frequency hopping with a TDMA scheme utilizing a centralized a-priori slot allocation mechanism. The comparison provides a careful and balanced answer on the advantages and drawbacks of the WirelessHART and autonomous solutions for supporting periodic traffic. We evaluated both approaches in a range of scenarios using trace-based simulations, and we have shown that our approach, indeed, reaps the benefits of frequency hopping and also improves the adaptability to varying traffic, energy consumption and end delay over centralized system such as WirelessHART.

- Further, we studied sensitivity analysis which showed how the energy consumption depends on certain parameters such as the power consumption of the transceiver in different modes of operation (Tx, Rx, Listening, Sleeping, and turnover), as long as the length of learning phase and length of wakeup window factors. In this analysis we used the response surface methodology and showed the most influenced parameters to the energy consumption that one can optimize further.

- Based on our analysis of the WirelessHART we suggested several ideas to improve the WirelessHART standard performance in terms of energy and delay. One idea is to adapt the frequency of management traffic

by starting with a high frequency, and as soon as the network becomes somewhat stable, the management rate can be reduced. This can significantly reduce energy saving at the expense of longer joining times and slower network update times resulting from topology changes. Another approach would be to use piggybacking more extensively, for example to use periodic data packets also for management purposes by piggybacking additional information (e.g. keep-alive and health reports).

- Our approach is extremely light in terms of signaling, as only ACK packets need to carry few bits of information. We believe that our approach is an attractive alternative to WirelessHART and similar systems in lightly loaded networks with periodic traffic.

8.1 Future Works

In this section, we identified some future directions which we believe are worthwhile for farther future works for both autonomous framework and WirelessHART system.

8.1.1 Autonomous framework

The autonomous approach developed so far has some potential for optimization by considering the following ideas:

- One can enable traffic shaping mechanism for forwarder nodes to prevent large deviation in the periodicity and thus reduce the jitter.
- Light forms of signaling could be used, in which for example one node can piggyback local estimates onto data packets, helping an upstream node with estimating period and jitter.
- One can enhance the end-to-end delay by reducing the back-off contention interval dynamically. This is more useful in the case of moderate and low traffic rate.
- It would be also worthwhile to investigate different channel hopping patterns such as adaptive channel hopping in which a dynamic estimate of the channels is maintained with dynamic blacklisting.

8.1.2 WirelessHART

In the following we identified some future directions we believe are promising for WirelessHART system:

- Design and implement TDMA scheduling algorithms which explicitly take the presence of a local sleep scheduling algorithm and multiple sleep states into account by constructing schedules in which the slots of individual nodes have larger separations in time, so as to allow them to enter deeper sleep modes.

- Develop supporting features within the slot assignment algorithm to exploit the re-transmission slots and reuse them in efficient manner. For example the retransmission slots could be used to transmit buffered packets instead of wasting the retransmission slots in case of successful transmission. Also assignment algorithm could improve both energy consumption and packet delay by assigning same slots for multiple nodes (shared slots) for re-transmission.

- Another important improvement could be the design of hybrid slot assignment in which both centralized and distributed algorithms are to be selected based on the traffic load. As an example if the load is low then one can use distributed algorithm, otherwise use centralized algorithm. This could be triggered based on the traffic information on the network manager or even could be done by examining the buffer size of the nodes. Another idea would be to separate the two approaches; to use centralized algorithm for control packet only and distributed algorithm for data packet exchange.

- We also believe that a further investigation of different channel hopping patterns such as adaptive channel hopping in which a dynamic estimate of the channels is maintained is needed for such centralized solutions.

- Another interesting enhancement would be to adapt our proposed approach (on-the-fly adaptive traffic mechanism) to the WirelessHART standard. This would definitely improve adaptability of the WirelssHART and hence, improve the energy consumption and end-to end delay. Only one single bit should be reserved in the data or control packet. The network manager then could react faster to the new traffic demands and compute a new scheduler much faster than waiting for long time (about 15 minutes in light loaded network and much more in heavy loaded network).

APPENDIX A

A Simulation Model for the Autonomous Framework Protocol

A.1 Castalia Simulator

In order to develop a simulation model for both our developed autonomous framework and the benchmark, WirelessHART protocol, we first looked to the most known simulators in the area of WSNs. The survey for such simulators are beyond the scope of this thesis, however an interest reader is pointed to [42, 85, 101] for a comprehensive survey of the current WSNs simulators. For our work we are interested for a simulator that at least support modularity, realistic wireless channel and radio models for low power communication. Among the currently available WSNs tools and frameworks the Castalia WSNs simulator emerges for its quality and completeness [115, 97, 116, 155].

Castalia [128] is an OMNet++ based framework designed specifically for wireless sensor networks. There is an increasing number of researchers using Castalia to support their investigations [17, 161, 16, 50, 133, 162]. OMNet++ is an open-source discrete-event simulator, that support modularity. This makes OMNet an excellent choice for supporting frameworks for specialized research area.

In addition Castalia provides bundled support for the popular IEEE 802.15.4-compliant ChipCon CC2420 radio transceiver [30] which is the transceiver choice for the TelosB and TmoteSky platforme. This is particularly important since we use the TmoteSky platform in all of our investigations. An interested reader is refereed to the Castalia manual [128] for more details.

The main concepts of the OMNeT++ simulator are both modules and messages. The main abstraction model of the OMNeT++ is the Module.

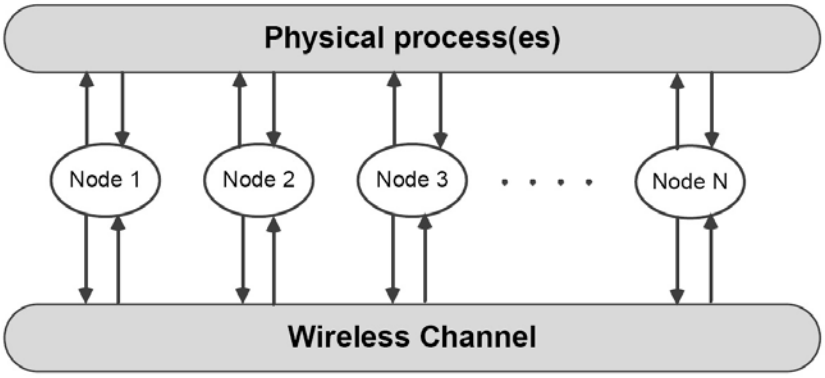


Figure A.1: Castalia's basic modules

Each module can accept message which called from other modules or from the same module (itself). Depending on the arrival of the message a module can then execute a set of instructions (small piece of code) and interact accordingly. This interactions could be a call of other module, schedule another message, or even keep some parameters state values. and example of a typical Castalia's basic modules is illustrated on Figure A.1;

A.2 Autonomous Framework Design

Figure A.2 depicts the high level architecture of our autonomous framework. This includes the state machine, estimation and identification, asynchronous channel hopping, local dynamic multiple sleep states scheduling, overlap mechanism, on-the-fly mechanism, timer, and buffer modules. The state machine module compose the main components of the MAC layer. It includes Tx-engine, Rx-engine, a clear channel assessment, and a backoff components (to be detailed in Section A.3). The autonomous module interacts with both the network and physical modules via the Network and MAC modules, respectively. Please note that our autonomous framework is independent from the main MAC functionalities. In the following sections we describe in detail the components of the autonomous framework. The module also implements the same connection to the Network and MAC modules following the same default standard Castalia's modules.

A.3 Autonomous State Machine Design

As explained in section 4.1.2 the autonomous framework has two main phases the learning phase and the operational phase. The operation of these phases are shown in Figure A.3. There are two main engines defined on the state

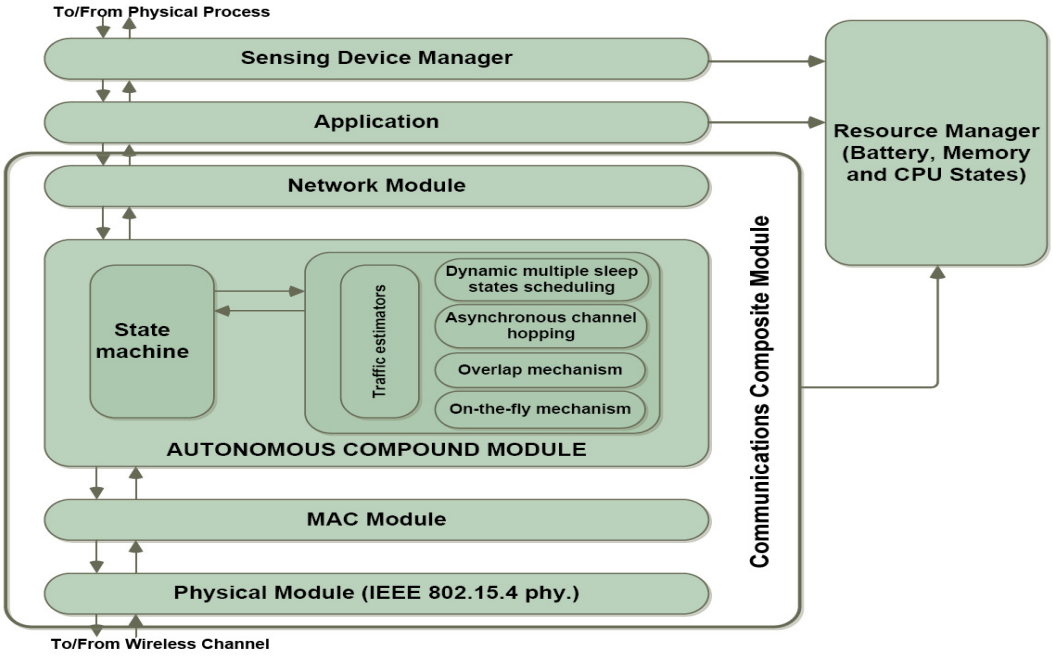


Figure A.2: Autonomous framework simulation model

machine of the autonomous framework: the Tx-engine and the Rx-engine. These are explained in the following sections.

A.3.1 Tx-Engine

When the transmit event starts (after the learning phase), node enters Tx state by invoking the Tx-engine and performs CCA (clear channel assessment). If the channel is clear then it transmits the packet and waits for an ACK. (see Figure A.4). If the channel is not clear then the radio backs-off for some short random period of time before attempting to transmit again. If packet transmission fails then node will try to retransmit its packet on different frequency. This happen after the ACK time-out is expired. The hopping policy is explained in Section 5.1. Each node updates all the parameters and goes to sleep after receiving a successfully ACK.

A.3.2 Rx-Engine

The receive event is managed by the Rx-engine. (see Figure A.5). Rx-engine is invoked to acquire a data or ACK packet that is being sent by one of the forwarders. When the Rx-engine is invoked, the transceiver is configured by selecting the correct channel. In addition, the time-out window is started and the *Receive packet* state is entered.

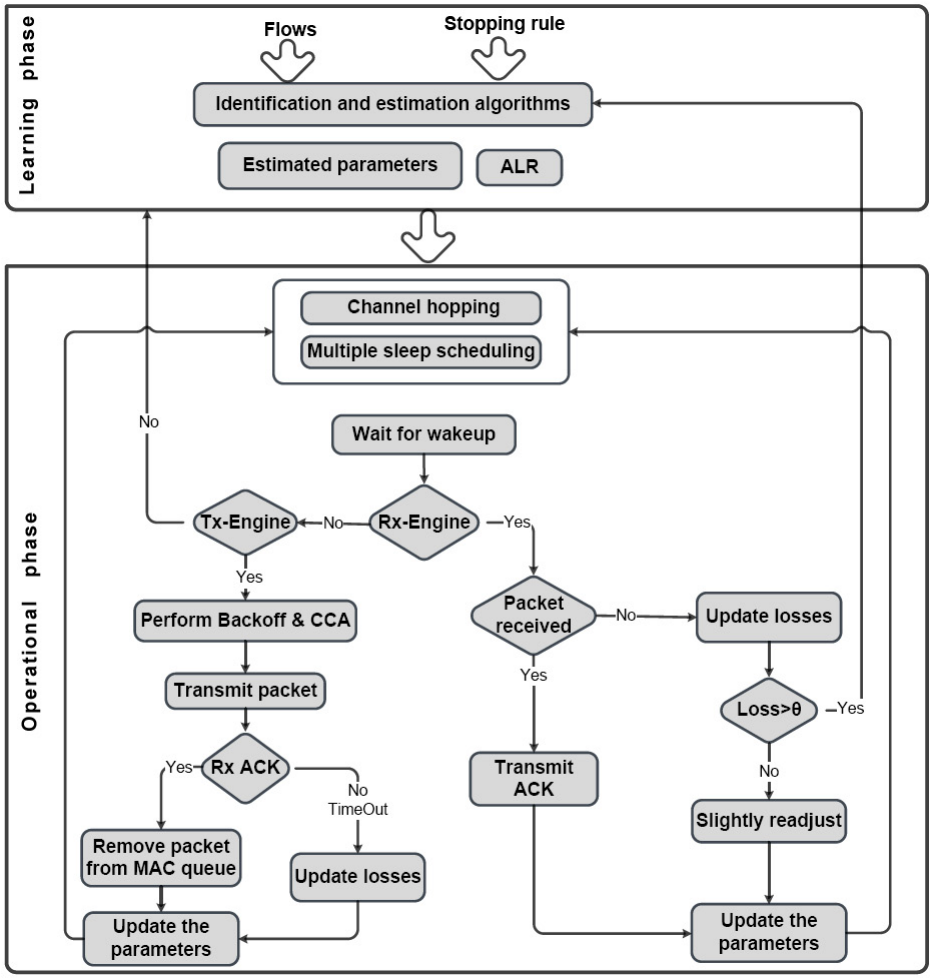


Figure A.3: Autonomous machine state diagram

The *Receive packet* state is left when the time-out window expired. The time-out of the *Receive packet* timer is set by the autonomous estimation algorithms to allow the receiver to become active at the beginning of the wakeup window. The duration of the wakeup window is also controlled by the estimation and sleep/wakeup algorithms. The node remains in the *Receive packet* state until either 1) the start of a packet is detected or 2) the *Receive packet* timer expires. If the receive packet timer is expired, then the node switches the channel according to our policy (will be explained in the next section). Consequently, the node readjusts the wakeup window and updates its parameters before it goes to sleep. Each forwarder checks its activity phase after receiving a potential incoming packet or after the wakeup window timer is expired.

we need to packet format for ACK special flag to let the forwarder delay its backoff when there is a collision.

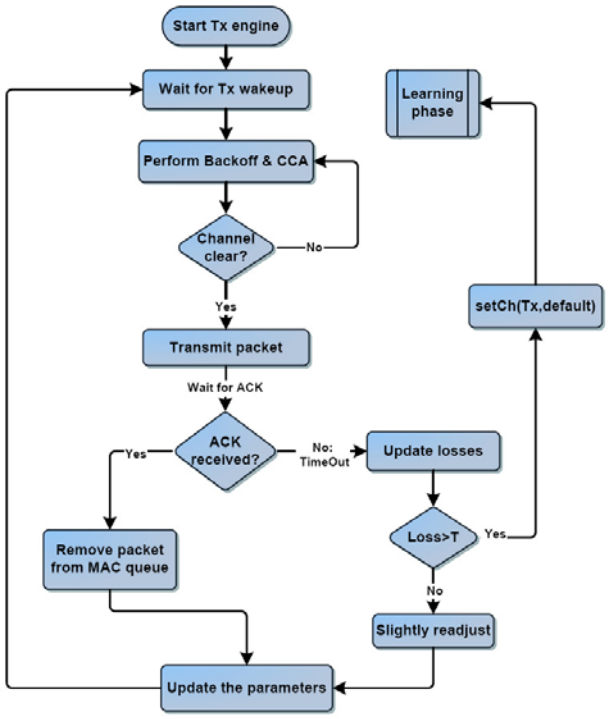


Figure A.4: Autonomous Tx state diagram

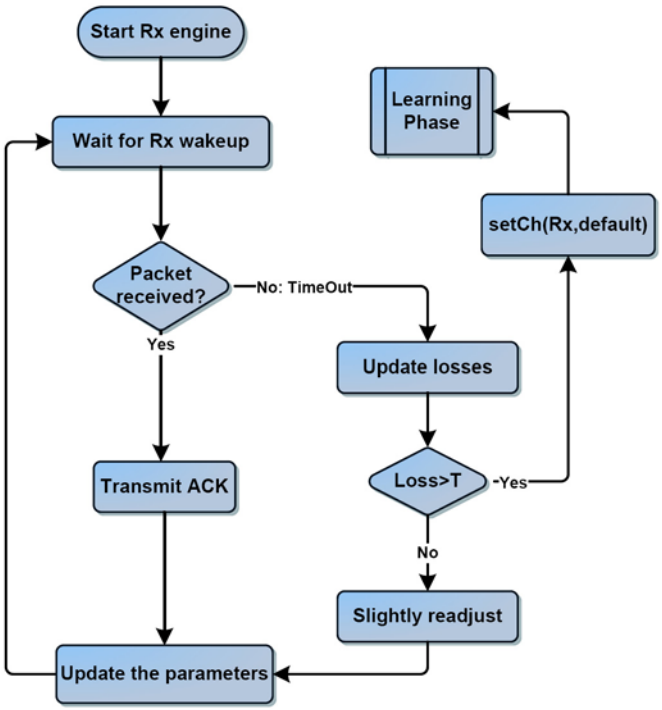


Figure A.5: Autonomous Rx state diagram

APPENDIX B

A Simulation Model for WirelessHART TDMA Protocol

In this appendix we present the design and implementation details of the simulation model for the WirelessHART TDMA protocol. This simulation model includes the design of the core state machine as well as the network manager components (slot assignment algorithms, and synchronization). In this appendix we also conduct a validation tests for the simulation model using a real WirelssHART hardware experiments from the main provider of this technology (Linear and Dust networks).

B.1 WirelessHART Simulation Model

Figure B.1 shows the basic simulation architecture for the wirelessHART compound module. This includes the state machine, link scheduler, timer, buffer, and communication tables modules. The communication tables module contains the following tables: superframe table, link table, neighbor table and graph table.

As show in the same Figure (B.1), WirelessHART module interacts with both the network and physical modules via the Network and Radio modules, respectively. The WHART module implements the same connection to the Network and Radio modules following the same default standard Castalia's modules. However, WHART poses some constraints on the underlying Network and Radio layers. Thus, we modified both the Radio and Network layers to fulfill the WirelessHART requirements.

The values of the relevant parameters of our implementation of Wire-

lessHART are set to the default values in the network definition NED files of the correspondent modules.

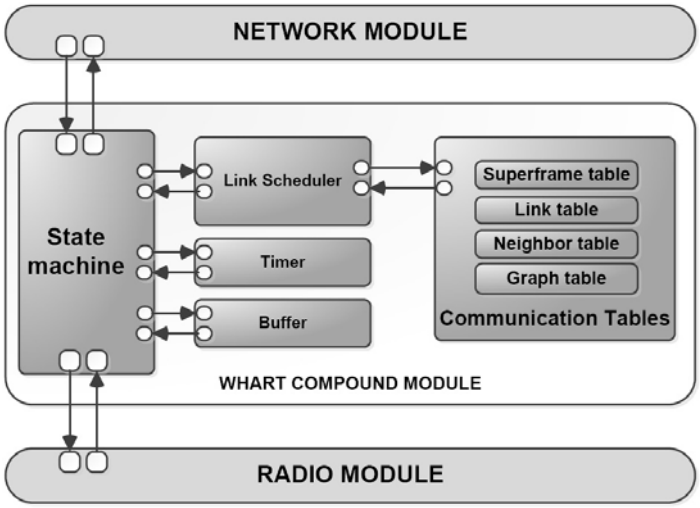


Figure B.1: Architecture of the WirelessHART compound module.

B.1.1 State machine

In Figure B.2 we show a schematic of the state machine that governs the operation field devices in the TDMA scheme. The three main operations of this state machine are: managing schedules, transmitting/receiving packets, and maintaining time synchronization. This machine starts when a device joins the network. Each device is configured with a superframe and link tables beforehand. After joining the network, the device is in *Idle* state.

The following functions may occur while the node in *Idle* state:

- SlotTimeOut - One of the core functions inside the WirelessHART TDMA simulator is slotTimeout function. This function is called periodically at the beginning of each time slot. In this function, the WirelessHART TDMA agent first decides if the slot is assigned to particular node by checking the communication tables. If so, the TDMA machine serves this event that indicates transmit (link=Tx) or receive (link=Rx). If the receiving slot times out the device enters *Talk* state for transmitting the packet or else it goes in the *Listen* state.

- A modification of link or superframe - this function is called whenever there is any modification of the device's list of superframe or links tables. Any modification of the superframe or links (e.g., in case of transmission

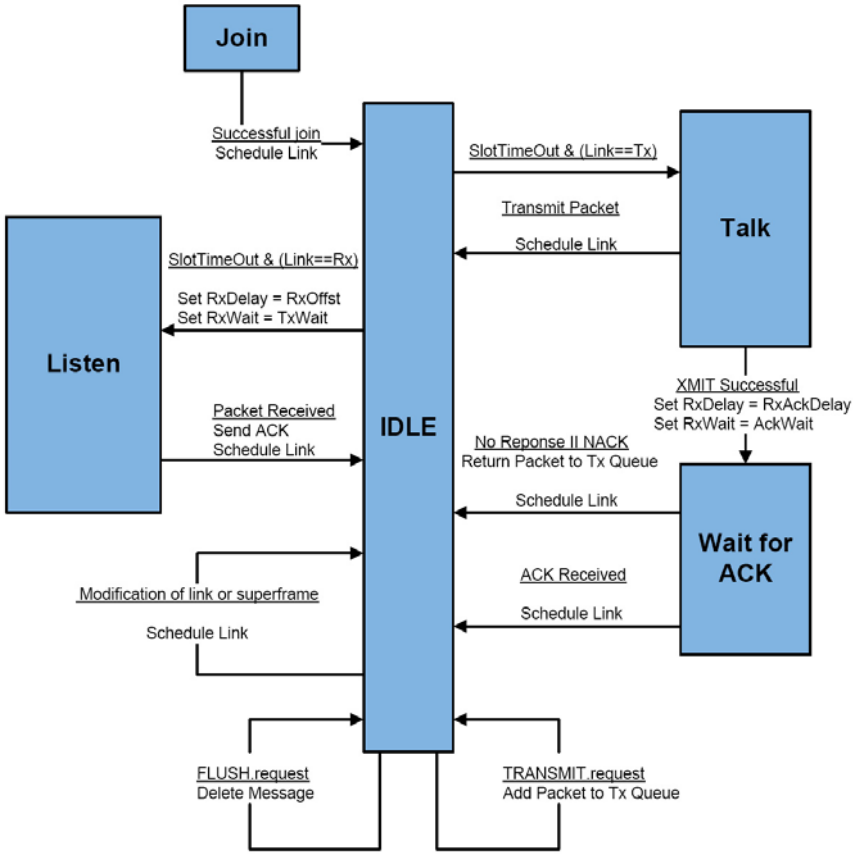


Figure B.2: State machine of WirelessHART [67].

error a node has the ability to enable extra slots for retransmission) affects link scheduling.

- a FLUSH.request - this function is invoked whenever a packet has to be deleted from the queue.
- TRANSMIT.request - This function is invoked to add a packet to be transmitted to the device's packet queue.

Each one of these function is tightly couple with a specific behavior of the module. Thus, when a transmit slot occurs ($slotTimeout\&link == Tx$), the device enters the *Talk* state by invoking the XMIT engine (see Section 2.4.5) and will attempt to transmit the packet to its next hop destination. The *Talk* state waits for its completion, upon successful transmission, a node enters the *WAIT-for-ACK* state by initializing the following two statements: (RxDelayTimer= TsRxAckDelay;) and the receiver window (RxWait= AckWait;). After performing these two statements, the state machine calls the

RECV engine (see Section 2.4.5). The RxDelay timer allows the receiving device to process the packet before any further activity of the device such as sending an ACK packet. The RxWait timer is used to set the duration of the receive window. The state machine stays in *WAIT-for-ACK* state until the RECV engine completes. Upon the reception of the ACK packet which indicates a successful transmission, the device enters the sleep state otherwise, the transmission fails and link scheduler is re-evaluated.

B.1.2 Communication Tables

each device maintains a set of tables that controls the communications performed by itself and collects statistics on those communications. The communication tables module includes the superframe table and link table which store communication configuration created by the network manager [70, 68, 69]. The neighbor table contains a list of all neighbor nodes that the device may able to reach directly, and the graph table is used to route packet from the source to the destination. The node does not know the entire rout rather, the graph indicates the next hop toward the destination.

Figure B.3 (UML digram) shows the relationship between the communication tables. Each node has a neighbor table which contains a list of all connected nodes.

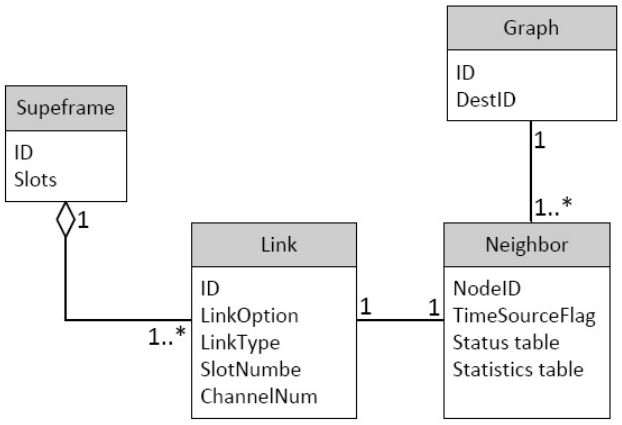


Figure B.3: Communication tables relationship diagram

A graph may specify more than one neighbor of which one may used for forwarding the packet to the next hop. In other words, when forwarding a packet using a graph routing, the device can transmit the packet to any of the neighbors associated with that packet's Graph ID. Each superframe has one or more links. The links specify the slot and associated information required

to receive or forward a packet. Please note that a link can belong to only one superframe.

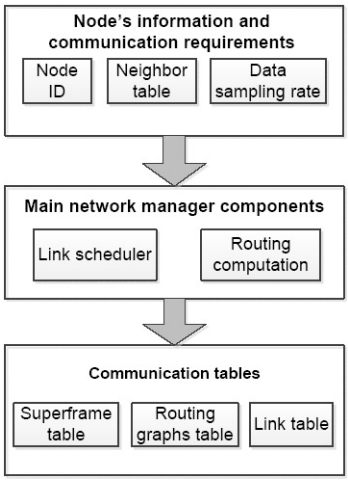


Figure B.4: Core network manager components.

B.1.3 Link scheduler

One of the main tasks in designing a TDMA protocol is the allocation of time slots to sender-receiver pairs. The WHART standard leaves many details of the slot allocation open, but provides a number of constraints that we follow in our implementation. These include:

1. Management slots have priority over data slots.
2. Each device gets three slots every 15 minutes for health reports.
3. Each device gets at least one slot every minute for management packets (advertisement, join request/response, command request/response).
4. Each device gets a slot for keep-alive packets every 60 seconds.
5. Slots for stations having the fastest transmission periods are allocated first. We refer to this as the fastest-periodic-flow-first (FPFF) policy.
6. Allocate at least one backup slot to each data slot to handle a retry.

Please note that even with the FPFF, scheduling can be done in several ways. We have used two different approaches, depth-first and breadth-first scheduling, both explained in the next two sections.

B.1.4 Breadth-first approach

Figure B.5 shows an example of the breadth-first approach. The network manager assigns time slots starting from the outmost sources. In the example, nodes S1, S2, and S3 are the sources, whereas nodes F1, F2, and F3 are the forwarders. The Network manager assigns one data slot for each source. For the first forwarder F1 it assigns one slot to forward the data of S1. F2 has two slots to

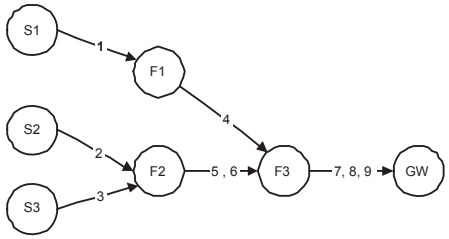


Figure B.5: Example of breadth based approach

forward the data for S2 and S3. F3 has three slots to forward the data it receives from F1 and F2. According to the breadth-first approach, time slots 1, 2 and 3 are assigned to sources S1, S2 and S3, respectively. Slot 4 is assigned to forwarder F1, slots 5 and 6 to forwarder F2, and slots 7, 8, and 9 to forwarder F3.

The time-slot assignment in the breadth-first approach is organized so that sources get the first slots, followed by the forwarders directly attached to sources (first-wave forwarders), followed by forwarders directly attached to first-wave forwarders and so on. This might cause a buffer problem for the later forwarders, since such a forwarder has to buffer all packets from forwarders from previous waves before getting a chance to empty his own buffer. In our example, it might happen that forwarder F3 has only two packet buffers. This would result in frequent buffer overflow.

B.1.5 Depth-first approach

The network manager may apply the depth-first approach as shown in Figure B.6. In this figure, S1 generates a packet in time slot 1, time slot 2 is assigned to F1 to forward the received packet to F3,

time slot 3 assigned to F3 for forwarding the packet to the GW. S2 generates its packet in time slot 4, time slot 5 is assigned to F2 and time slot 6 to F3. Similarly, S3 assigned time slot 7, time slots 8 and 9 are assigned to F2 and F3, respectively.

This assignment method avoids buffer overflows in forwarders.

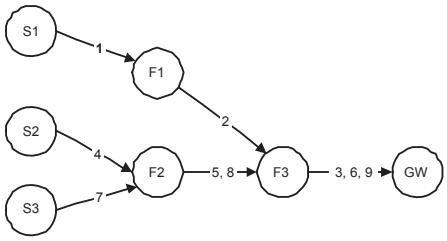


Figure B.6: Example of depth based approach

There are two types of superframe data super-frame and management superframe. The data superframe is used to transmit the generated data from the nodes to the sink and the management packet superframes is used for transmitting control and management packets from/to nodes and the sink. Please note that the length of the data superframe and the management superframe are not necessary the same size.

We created and maintained a two dimensional array data structure A, to represent both time slot and channels. Each entry in the $A_{i,j}$ represents the slot usage at time-slot i and on channel j. Each entry in the multi-dimensional array A is associated with a particular type of operation. These types indicate whether the slot is dedicated for particulate transmission, shared within different nodes, or free to assign. We also maintain different data structure for construction the communication assignments table. One data structure represents the data superframe which denoted as S_i per packet generation period p_i and a management superframe S_m. Details of these procedures are presented in [132].

Bibliography

[1] N. AbouGhazaleh, P. Lanigan, S. Gobriel, D. Mosse, and R. Melhem. Dynamic rate-selection for extending the lifetime of energy-constrained networks. In *Proceedings of the 23rd IEEE International Conference on Performance, Computing, and Communications*, pages 553 – 558, Phoenix, Arizona, USA, June 2004.

[2] M. Adler, R. K. Sitaraman, A. L. Rosenberg, and W. Unger. Scheduling time-constrained communication in linear networks 1998, pp. 269 278. In *Proc. ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 1998.

[3] R. Adler, M. Flanigan, J. Huang, R. Kling, N. Kushalnagar, L. Nachman, C.-Y. Wan, and M. Yarvis. Intel mote 2: An advanced platform for demanding sensor network applications. In *Proceedings of the Second ACM Conferences on Embedded Networked Sensor Systems (SenSys)*, San Diego, California, USA, November 2005.

[4] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou. A multi-radio unification protocol for ieee 802.11 wireless networks. In *Proceedings of the First International Conference on Broadband Networks (BROADNETS'04)*, pages 344 – 354, San Jose, California, USA, October 2004.

[5] I. F. Akyildiz and I. H. Kasimoglu. Wireless sensor and actor networks: research challenges. *Elsevie, In International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 1–23, May 2004.

[6] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *The International Journal of Computer and Telecommunications Networking*, 38(4):393–422, March 2002.

[7] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 38(4):393–422, 2002.

[8] American National Standards Institute, Washington, DC. *ANSI/ISA-100.11a-2011 – Wireless systems for industrial automation: Process control and related applications*, 2011.

[9] G. Asada, M. Dong, T. S. Lin, F. Newberg, G. Pottie, W. J. Kaiser, and H. O. Marcy. Wireless integrated network sensors: Low power systems on a chip. In *Proceedings of the 24th European Solid-State Circuits Conference, (ESSCIRC '98)*, pages 9–16, Los Angeles, California, September 1998.

[10] F. Ashraf, R. Crepaldi, and R. H. Kravets. Synchronization vs. signaling: Energy-efficient coordination in wsn. In *Proceedings of the fifth IEEE Wireless Mesh Networks (WIMESH 2010)*, pages 1–6, Boston, MA, USA, June 2010.

[11] A. Bachir, D. Barthel, M. Heusse, and A. Duda. Micro-frame preamble mac for multihop wireless sensor networks. In *Proceedings of International Conference on Communications (ICC), IEEE ICC,06*, pages 3365–3370, Turkey, Istanbul, June 2006.

[12] A. Bachir, M. Dohler, T. Watteyne, and K. K. Leung. Mac essentials for wireless sensor networks. *IEEE Communications Surveys and Tutorials*, 12(2):222 – 248, April 2010.

[13] P. Bahl, R. Chandra, and J. Dunagan. Ssch: Slotted seeded channel hopping for capacity improvement in ieee 802.11 ad-hoc wireless networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking, MobiCom '04*, pages 216–230, New York,NY,USA, september 2004.

[14] S. Bapat, V. Kulathumani, and A. Arora. Analyzing the yield of exscal, a large-scale wireless sensor network experiment. In *Proceedings of the 13TH IEEE International Conference on Network Protocols, (ICNP '05)*, pages 53 – 62, NY, USA, November 2005.

[15] L. Benini, A. Bogliolo, and G. D. Micheli. A survey of design techniques for system-level dynamic power management. In *Proc. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 2000,*, volume vol.8, NO. 3, pages pp. 299–316, NJ, USA, June 2000.

[16] L. Bergamini, C. Crociani, A. Vitaletti, and M. Nati. Validation of wsn simulators through a comparison with a real testbed. In *Proc. ACM PE-WASUN '10, the 7th ACM workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 103–104, New York, NY, USA, October 2010.

[17] A. Boulis, A. Fehnker, M. Fruth, and A. McIver. Simulation and model checking for wireless sensor networks. In *Proc. QEST '08, the fifth International Conference on Quantitative Evaluation of Systems (QEST 2008)*, pages 37–48, Saint Malo, France, September 2008.

[18] A. Boulis, Y. Tselishchev, L. Libman, D. Smith, and L. Hanlen. Impact of wireless channel temporal variation on mac design for body area networks. *ACM Transactions on Embedded Computing Systems (TECS, Issue S2)*, 11(51):51:1–51:18, August 2012.

[19] M. I. Brownfield, K. Mehrjoo, A. ad S. Fayez, and N. J. Davis. Wireless sensor network energy-adaptive mac protocol. In *Consumer Communications and Networking Conference, CCNC 2006. 3rd IEEE*, pages 778–782, Jan 2006.

[20] M. Buettner, G. Yee, E. Anderson, and R. Han. X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceeding ACM SenSys '06 Proceedings of the 4th international conference on Embedded networked sensor systems, 2006*, pages 307–320, New York, USA, November 2006.

[21] N. Bulusu and S. Jha. *Wireless Sensor Networks: A System Perspective*. Artech House Mems and Sensors Library, ISBN:1580538673, Copyright 2005.

[22] A. Burns, B. R. Greene, M. J. McGrath, T. J. OShea, B. Kuris, S. M. Ayer, F. Stroiescu, and V. Cionca. A lightweight medium access protocol (lmac) for wireless sensor networks: Reducing preamble transmissions and transceiver state switches. *IEEE Sensors Journal*, 10(9):1527–1534, September 2010.

[23] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazzo. An implicit prioritized access protocol for wireless sensor networks. In *Proceedings Real-Time Systems Symposium, RTSS. 23rd IEEE*, pages 39 – 48, IL, USA, December 2002.

[24] B. Carbunar, A. Grama, J. Vitek, and O. Carbunar. Redundancy and coverage detection in sensor networks. *ACM Transactions on Sensor Networks, (TOSN)*, 2(1):94–128, February 2006.

[25] M. Cardei, M. T. T. Y. Li, and W. Wu. Energy -efficient target coverage in wireless sensor networks. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM'5 Vol.3*, pages 1976 – 1984, Miami, USA, March 2005.

[26] S. Chatterjea and P. Havinga. An adaptive and autonomous sensor sampling frequency control scheme for energy-efficient data acquisition in wireless sensor networks. In *Proceedings of the 4th IEEE international conference on Distributed Computing in Sensor Systems, DCOSS '08*, pages 60–78, Berlin-Germany, June 2008.

[27] D. Chen, M. Nixon, and A. Mok. *WirelessHART(TM): Real-Time Mesh Network for Industrial Automation*. Springer, April 2010.

[28] X. Chen, P. Han, Q.-S. He, S.-L. Tu, and Z.-L. Chen. A multi-channel mac protocol for wireless sensor networks. In *Proc. of Computer and Information Technology, 2006. CIT '06. The Sixth IEEE International Conference*, pages 224 – 230, Fudan University, China, September 2006.

[29] O. Chipara, C. Lu, and G.-C. Roman. Efficient power management based on application timing semantics for wireless sensor networks. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, (ICSCS'05)*, pages 361 – 370, Columbus, OH, USA, June 2005.

[30] Chipcon. *CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver Available: http://www.chipcon.com.*, 2004.

[31] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore. Environmental wireless sensor networks. *Proceedings of the IEEE*, 98(11):1903–1917, 2010.

[32] D. Culler, D. Estrin, and M. Srivastava. Overview of sensor networks. In *IEEE Computer, Special Issue in Sensor Networks*, 37(8): 41-49, Aug 2004.

[33] W. Dargie, X. Chao, and M. K. Denko. Modelling the energy cost of a fully operational wireless sensor network. *Telecommunications Systems*, 44(1-2):3–15, June 2010.

[34] J. Degesys, I. Rose, A. Patel, and R. Nagpal. Desync: Selforganizing desynchronization and tdma on wireless sensor networks. In *Proceedings of the 7th international conference on Information processing in sensor networks, IPSN '07*, pages 11 – 20, Cambridge, Massachusetts, USA, April 2007.

[35] M. Dhanaraj, B. S. Manoj, and C. S. R. Murthy. A new energy efficient protocol for minimizing multi-hop latency in wireless sensor networks. In *Proc. 3rd IEEE Intl Conf. on Pervasive Computing and Communications (PerCom 2005)*, 2005.

[36] L. Doherty, W. Lindsay, and J. Simon. Channel-specific wireless sensor network path data,. In *IEEE 16th Internatioanl Conference on Computer Communications and Networks (ICCCN), 2007*, pages 89 – 94, Turtle Bay Resort, Honolulu, Hawaii, USA, August 13-16 2007.

[37] W. Dron, S. Duquennoyy, T. Voigt, K. Hachicha, and P. Garda. An emulation-based method for lifetime estimation of wireless sensor networks. In *Proceedings of the 10th IEEE international conference on Distributed Computing in Sensor Systems, DCOSS '14*, pages 1–8, California, USA, May 2014.

[38] S. Duquennoy, O. Landsiedel, and T. Voigt. Let the tree bloom: Scalable opportunistic routing with orpl. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys'13)*, pages 1–14, Rome, Italy, November 2013.

[39] Dust Networks. *Wirelesshart technical data sheet. White paper, Dust Networks, September 2007.*

[40] P. Dutta and D. Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *Proceedings of the 6th ACM conference on Embedded network sensor systems, SenSys '08*, pages 71–84, New York,USA, November 2008.

[41] P. Duttat, J. Huitt, J. Jeongt, S. Kimt, C. Sharp, J. Tanejat, G. Tollett, K. Whitehouset, and D. Cullertt. Trio: Enabling sustainable and scalable outdoor wireless sensor network deployments. In *Proceedings of Fifth International Conference on Information Processing in Sensor Networks, (IPSN'06)*, pages 407 – 415, Nashville,TN, USA, April 2006.

[42] E. Egea-Lopez, J. Vales-Alonso, A. S. Martinez-Sala, P. Pavon-Marino, and J. Garcia-Haro. Simulation tools for wireless sensor networks. In *Proc. SPECTS 2005, the International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, pages 2–9, PA, USA., July 2005.

[43] A. El-Hoiydi, J.-D. Decotignie, C. Enz, and E. L. Roux. Poster abstract: Wisemac, an ultra low power mac protocol for the wisenet wireless sensor network. In *Proc. ACM SenSys 03*, Los Angeles, California, Nov. 2003. Poster Abstract.

[44] J. Elson and D. Estrin. Random, ephemeral transaction identifiers in dynamic sensor networks. In *Proc. 21st International Conference on Distributed Computing Systems (ICDCS-21)*, Apr. 2001.

[45] J. Elson and K. Romer. Wireless sensor networks: A new regime for time synchronization. In *Proc. HotNets-I First Workshop on Hot Topics In Networks*, New Jersey, USA, October 28-29 2002.

[46] S. C. Ergen and P. Varaiya. Pedamacs: Power efficient and delay aware medium access protocol for sensor networks. *IEEE TRANSACTIONS ON MOBILE COMPUTING*, 5(7):920–930, JULY 2006.

[47] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. Low-power wireless bus. In *Proceedings of the 10th ACM Conference on Embedded Networked Sensor Systems (SenSys'12)*, pages 1–14, Toronto, ON, Canada, November 2012.

[48] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo. The collection tree protocol (ctp). TEP 123, TinyOS Network Working Group, Aug. 2006.

[49] H. C. Foundation. *HART Communication Foundation. WirelessHart Technology. Available online: http://www.hartcomm.org/protocol/wihart/wirelesstechnology.html*.

[50] O. Gama, P. Carvalho, J. A. Afonso, and P. M. Mendes. Trade-off analysis of a mac protocol for wireless e-emergency systems. In *Proc. S-CUBE'09,*, pages 222–235, Pisa, Italy, September 2009.

[51] D. Gay, P. Levis, R. V. Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *Proc. ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI)*, San Diego, California, USA., June 2003.

[52] D. Gay, P. Levis, and D. Culler. Software design patterns for tinyos. *ACM Transactions on Embedded Computing Systems (TECS) Special Section LCTES*, 6(2):1–40, 2007.

[53] A. Giridhar and P. Kumar. Maximizing the functional lifetime of sensor networks. In *Proceedings of the 4th international conference on Information processing in sensor networks, IPSN '05*, pages 5–12, Los Angeles, California, USA, April 2005.

[54] S. D. Glaser. Some real-world applications of wireless sensor nodes. In *Proc. (SPIE) Symposium on Smart Structures and Materials/ NDE 2004*, San Diego, California, Mar. 2004.

[55] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys 2009)*, pages 1 – 13, Berkeley, CA, USA,, November 2009.

[56] A. Gonga, O. Landsiedel, P. Soldati, and M. Johansson. Multi-channel communication vs. adaptive routing for reliable communication in wsns. In *Proceedings of the the 11th international conference on Information Processing in Sensor Networks, IPSN'12*, pages 125–126, Beijing, China, April 2012.

[57] A. Gonga, O. Landsiedel, P. Soldatiz, and M. Johansson. Revisiting multi-channel communication to mitigate interference and link dynamics in wireless sensor networks. In *Proceedings of the IEEE 8th International Conference on Distributed Computing in Sensor Systems (DCOSS,2012)*, pages 186 – 193, hangzhou, china, May 2012.

[58] B. S. I. Group. *Specification of the Bluetooth System Version 4.0 [Vol 0] std*, 2009.

[59] D. Gustafsson. Wirelesshart - implementation and evaluation on wireless sensors, available: http://www.ee.kth.se/php/modules/publications/reports/2009/xr-ee-rt2009003.pdf. Technical report, KTH School of Electrical Engineering, Masters's Degree Project, XR-EE-RT 2009:003, Stockholm, Sweden, April 2009.

[60] G. Halkes and K. Langendoen. Crankshaft: An energy-efficient mac-protocol for dense wireless sensor networks. In *Proceedings of the 4th European Conference on Wireless Sensor Networks (EWSN 2007)*, pages 228–224, LNCS 4373, Springer 2007.

[61] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon. Reliable and real-time communication in industrial wireless mesh networks. In *Proceedings Real-Time and Embedded Technology and Applications Symposium (RTAS), 2011 17th IEEE*, pages 3–12, Chicago, IL, USA, April 2011.

[62] V. Handziski, A. Kpke, A. Willig, and A. Wolisz. Twist: A scalable and reconfigurable testbed for wireless indoor experiments with sensor network. In *Proc. of the 2nd Intl. Workshop on Multi-hop Ad Hoc Networks: from Theory to Reality, (RealMAN 2006)*, pages 63–70, Florence, Italy, May 2006.

[63] HART Communication Foundation. *HART Communication Protocol Specification, HCF SPEC 13 Revision 7.1, 05 June, 2008.*

[64] HART Communication Foundation. *Network Management Specification, HCF SPEC 085 Revision 1.1, 30 May, 2008.*

[65] HART Communication Foundation. *Network Management Specification, HCF SPEC 085 Revision 1.1 May, 2008.*

[66] HART Communication Foundation. *TDMA Data Link Layer Specification, HCF SPEC 075 Revision 1.1, 17 May, 2008.*

[67] HART Communication Foundation. *TDMA Data Link Layer Specification, HCF SPEC 075 Revision 1.1 May, 2008.*

[68] HART Communication Foundation. *Wireless Command Specification, HCF SPEC 155 Revision 1.1 May, 2008.*

[69] HART Communication Foundation. *WirelessHART Command Response Code Specification, HCF SPEC 307 Revision 6.0 September, 2007.*

[70] HART Communication Foundation. *WirelessHART Command Tables Specification, HCF SPEC 183 Revision 19.0 May, 2008.*

[71] HART Communication Foundation. *WirelessHART Device Specification, HCF SPEC 290 Revision 1.1, 22 May, 2008.*

[72] T. He, P. Vicaire, T. Yan, Q. Cao, G. Zhou, L. Gu, L. Luo, R. Stoleru, J. A. Stankovic, and T. F. Abdelzaher. Achieving long-term surveillance in vigilnet. In *ACM Transactions on Sensor Networks (TOSN)*, pages 13–24, Baltimore, MD, USA, February 2009.

[73] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proc. ACM of the 9th Intl. Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS),*, pages 83–104, 2000.

[74] B. Hohlt, L. Doherty, and E. Brewer. Flexible power scheduling for sensor networks. In *Proc. of IPSN*, Berkeley, California, USA., April 2004.

[75] http://www.xbow.com. *XBOW MICA2 Mote Specifications.*

[76] J. W. Hui and D. E. Culler. Ipv6 in low-power wireless networks. *In Proceedings of the IEEE/ACM*, 98(11):1865 – 1878, November 2010.

[77] P. Hurni, T. Braun, and M. Anwander. Evaluation of wisemac and extensions on wireless sensor nodes. *springer-tcs*, 43(1-2):49–58, Feb. 2010.

[78] C.-H. Hwang and A. C.-H. WU. A predictive system shutdown method for energy saving of event-driven computaion. In *Proc. ACM Transactions on Design Automation of Electronic Systems 2000*, volume vol.5, pages pp. 226–241, New York, USA, April 2000.

[79] e. IEEE standards association. *IEEE Standard for Local and metropolitan area networks. Part 15.4: LowRate Wireless Personal Area Networks (LRWPANs). Number 802.15.4-2011 in IEEE Std. IEEE, New York, September*, 2011.

[80] I.-S. IEEE Standards Association. *IEEE standard for Information Technology, "IEEE std. 802.15.4e, Part. 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendament 1: MAC sublayer". Available: http://standards.ieee.org/about/get/802/802.15.html*, 2012.

[81] O. D. Incel, S. Dulman, and P. Jansen. Multi-channel support for dense wireless sensor networking. *In First European Conference on Smart Sensing and Context (EuroSSC) Lecture Notes in Computer Science 4272, Springer*, 4:1–14, 2006.

[82] O. D. Incel, S. Dulman, P. Jansen, and S. Mullender. Multi-channel interference measurements for wireless sensor networks. In *Proceedings of the 31st IEEE Conference on Local Computer Networks*, pages 694–701, Tampa, FL, USA, November 2006.

[83] ISA. *ISA 100: wireless system for automation. Available: http:// isa.zigbee.org.*

[84] I.-a.-. ISA. *Wireless System for Industrial Automation: Process Control and Related Applications*, May 2011.

[85] M. Jevtic, N. Zogovic, and G. Dimic. Evaluation of wireless sensor network simulators. In *Proc. of the 17th Telecommunications Forum (TELFOR 2009)*, pages 1303–1306, Serbia, Belgrade, November 2009.

[86] A. Kansal, A. Ramamoorthy, M. B. Srivastava, and G. J. Pottie. On sensor network lifetime and data distortion. In *Proceedings of the International Symposium on Information Theory (ISIT 2005)*, pages 6–10, Adelaide, Australia, September 2005.

[87] H. Karl and A. Willig. *Protocols and Architectures for Wireless Sensor Networks.* John Wiley and Sons, Ltd. ISBN: 0-470-09510-5, Copyright 2005.

[88] N. Karowski, A. C. Viana, and A. Wolisz. Optimized asynchronous multi-channel discovery of ieee 802.15.4-based wireless personal area networks. *IEEE Transactions on Mobile Computing*, 12(10):1972–1985, August 2012.

[89] B. Kerkez and K. Pister. Adaptive time synchronization and frequency channel hopping for wireless sensor networks. *Electrical Engineering and Computer Sciences University of California at Berkeley, Technical Report No. UCB/EECS-2012-56*, May 2012.

[90] O. Khader and A. Willig. An energy consumption analysis of the wireless hart tdma protocol. *Computer Communications*, 36(7):804–816, 2013.

[91] O. Khader, A. Willig, and A. Wolisz. Dynamic adaptive wake-up scheduling scheme for supporting periodic traffic in wsns. Technical report, Telecommunication Networks Group, Technical University Berlin, TKN Technical Report Series TKN-08-012, Berlin, Germany, Nov. 2008.

[92] O. Khader, A. Willig, and A. Wolisz. Distributed wakeup scheduling scheme for supporting periodic traffic in wsns. In *Proc. European Wireless (EW 2009)*, pages 287–292, Aalborg, Denmark, May 2009.

[93] O. Khader, A. Willig, and A. Wolisz. A simulation model for the performance evaluation of wirelesshart tdma protocol. Technical report, Telecommunication Networks Group, Technical UniversityBerlin, TKN Technical Report Series TKN-11-001, Berlin, Germany, May 2011.

[94] O. Khader, A. Willig, and A. Wolisz. Wirelesshart tdma protocol performance evaluation using response surface methodology. In *Proc. 6th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, Barcelona, Spain, Oct. 2011.

[95] O. Khader, A. Willig, and A. Wolisz. Analyzing the energy consumption of wirelesshart tdma protocol. Technical report, Telecommunication Networks Group, Technical UniversityBerlin, TKN Technical Report Series TKN-12-00x, Berlin, Germany, April 2012.

[96] O. Khader, A. Willig, and A. Wolisz. An autonomous framework for supporting energy efficiency and communication reliability in wsns. In *Proc. IEEE/IFIP Wireless and Mobile Networking Conference (WMNC'2013)*, Dubai, UAE, Apr. 2013.

[97] A. Khadivi and M. Hasler. Fire detection and localization using wireless sensor networks. In *Proc. of SENSAPPEAL 2010*, volume vol.29, pages 16–26, New York, NY, USA, October 2010.

[98] A. N. Kim, F. Hekland, S. Petersen, and P. Doyle. When hart goes wireless: Understanding and implementing the wirelesshart standard. In *Proceedings of IEEE International Conference on Emerging Technologies and Factory Automation (ETFA,2008)*, pages 899–907, Hamburg, Germany, September 2008.

[99] Y. Kim, H. Shin, and H. Cha. Y-mac: An energy-efficient multi-channel mac protocol for dense wireless sensor networks. In *Proc. of IPSN '08, Proceedings of the 7th international conference on Information processing in sensor networks*, pages 53 – 63, Washington, DC, USA, April 2008.

[100] A. Kopke. Engineering a communication protocol stack to support consensus in sensor networks. Technical report, Telecommunication Networks Group, Technical University Berlin, PhD thesis, D 83, Berlin, Germany, Oktober 2012.

[101] M. Korkalainen, M. Sallinen, N. Karkkainen, and P. Tukeva. Survey of wireless sensor networks simulation tools for demanding applications. In *Proceedings of the fifth International Conference on Networking and Services, ICNS '09*, pages 102–106, Valencia, Spain, April 2009.

[102] N. Kushalnagar, G. Montenegro, and C. Schumacher. *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*, August 2007.

[103] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson. Low power, low delay: Opportunistic routing meets duty cycling. In *Proceedings of 11th International Conference on Information Processing in Sensor Networks, (IPSN'12)*, pages 185–196, Beijing, China, April 2012.

[104] K. Langendoen and G. Halkes. Energy-efficient medium access control. In *Embedded systems handbook, CRC Press*, August 2005.

[105] K. Langendoen and A. Meier. Analyzing mac protocols for low data-rate applications. *ACM Transactions on Sensor Networks TOSN*, 7(1):13–24, August 2010.

[106] A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, third edition, 2000.

[107] H. K. Le, D. Henriksson, and T. Abdelzaher. A practical multi-channel media access control protocol for wireless sensor networks. In *Proceedings of the 7th international conference on Information processing in sensor networks, IPSN '08*, pages 70–81, Missouri, USA, April 2008.

[108] T. Lennvall and S. Svensson. A comparison of wirelesshart and zigbee for industrial applications. In *Proceedings of the 2nd international conference on Embedded networked sensor systems, SenSys '04*, pages 85 – 88, Dresden , Germanmy, May 2008.

[109] M. Li and Y. Liu. Underground coal mine monitoring with wireless sensor networks. *ACM Transactions on Sensor Networks, Article No. 10*, 5(2):1–29, March 2009.

[110] E.-Y. A. Lin, J. M. Rabaey, S. Wiethoelter, and A. Wolisz. Receiver initiated rendezvous schemes for sensor networks. In *Proceedings of the IEEE Global Telecommunications Conference, GLOBECOM '05. IEEE*, pages 3117–3122, St. Louis, MO, USA, December 2005.

[111] E.-Y. A. Lin, J. M. Rabaey, and A. Wolisz. Power-efficient rendez-vous schemes for dense wireless sensor networks. In *Proceedings of the IEEE ICC International Conference on Communications, Volume 7*, pages 3769 – 3776, Paris, France, June 2004.

[112] Y. Liu, Y. He, M. Li, J. Wang, K. Liu, L. Mo, W. Dong, Z. Yang, M. Xi, J. Zhao, and X.-Y. Li. Does wireless sensor network scale? a measurement study on greenorbs. In *Proceedings of the IEEE INFOCOM*, pages 873 – 881, Shanghai, April 2011.

[113] B. K. M. Zuniga. Analyzing the transitional region in low power wireless links. In *First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON)*, Santa Clara, CA,, October 2004.

[114] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems, SenSys '04*, pages 39–49, Baltimore, Maryland, USA, November 2004.

[115] A. Meier, M. Motani, H. Siquan, and S. Kunzli. Dimo: Distributed node monitoring in wireless sensor networks. In *Proc. ACM MSWiM '08, the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems,(MSWiM 2008).*

[116] A. Meier, M. Woehrle, M. Weise, J. Beutel, , and L. Thiele. Nose: Efficient maintenance and initialization of wireless sensor networks. In *Proc. SECON'09, the 6th Annual IEEE communications society conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 395–403, NJ, USA, June 2009.

[117] V. P. Mhatre, D. Kofman, R. Mazumdar, and N. Shroff. A minimum cost heterogeneous sensor network with a lifetime constraint. *IEEE Transactions on Mobile Computing*, 4(1):4–15, January 2005.

[118] W. Mo, D. Qiao, and Z. Wang. Mostly-sleeping wireless sensor networks: Connectivity, k-coverage, and alpha-lifetime. In *Proceedings of the 43rd Annual Allerton Conference on Communication, Control, and Computing*, pages 1–10, Monticello, Illinois, USA, September 2005.

[119] D. C. Montgomery. *Design and Analysis of Experiments*. John Wiley and Sons,Inc, 2012.

[120] S. Moon, T. Kim, and H. Cha. Enabling low power listening on ieee 802.15.4-based sensor nodes. In *The 5th IEEE Wireless Communications & Networking Conferencen (WCNC)*, Hong Kong, China, March 2007.

[121] D. Moss, J. Hui, P. Levis, and J. I. Choi. *TEP 126: CC2420 Radio Stack. [Online]. Available: http://www.tinyos.net/tinyos-2.x/doc/html/tep126.ht*.

[122] Moteiv Corporation, November. *Tmote Sky - Ultra low power IEEE 802.15.4 compliant wireless sensor module*, 2006.

[123] R. Myers and D. Montgomery. *Response Surface Methodology*. John Wiley and Sons,Inc, 2002.

[124] B. A. Nahas, S. Duquennoy, V. Iyery, and T. Voigt. Low-power listening goes multi-channel. In *Proceedings of the 10th IEEE international conference on Distributed Computing in Sensor Systems, DCOSS '14*, pages 1–8, California, USA, May 2014.

[125] A. Nasipuri and S. R. Das. Multichannel csma with signal powerbased channel selection for multihop wireless networks. In *Proceedings Vehicular Technology Conference. IEEE-VTS Fall VTC.*, volume 1, pages 211–218, Boston, MA, September 2000.

[126] A. Nasipuri, J. Zhuang, and S. R. Das. A multichannel csma mac protocol for multihop wireless networks. In *Proceedings Wireless Communications and Networking Conference, WCNC. IEEE*, volume 3, pages 1402–1406, New Orleans, LA, September 1999.

[127] A. Nasipuri, J. Zhuang, and S. R. Das. A multichannel csma mac protocol for multihop wireless networks. In *Proceedings of Wireless Communications and Networking Conference, 1999. IEEE WCNC. 1999*, pages 1402–1406, New Orleans,USA, september 1999.

[128] NICTA. *The Castalia simulator for Wireless Sensor Networks. Available: http://castalia.npc.nicta.com.au.*

[129] T. I. of Electrical and e. Electronics Engineers Inc. *IEEE Standard for Information technologyTelecommunications and information exchange between systemsLocal and metropolitan area networksSpecific requirements Part 11:Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Number 802.112007 in IEEE Std. IEEE, New York, December*, 2007.

[130] L. S. C. of the IEEE Computer Society. *IEEE STD 802.15.4-2006. Wireless Medium Access Contril (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs) Task Group. Available: http:// www.ieee802.org/15/pub/TG4.html*, 2006.

[131] J. Ortiz and D. Culler. Multichannel reliability assessment in real world wsns. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, (IPSN '10)*, pages 162–173, Stockholm, Sweden, April 2010.

[132] A. W. Osama Khader and A. Wolisz. Self-learning and self-adaptive framework for supporting high reliability and low energy expenditure in wsns. *IEEE Special issue of Telecommunication Systems Journal (To Appear 2014)*, pages 1–36, 2014.

[133] D. Pediaditakis, Y. Tselishchev, and A. Boulis. Performance and scalability evaluation of the castalia wireless sensor network simulator. In *Proc. ACM SIMUTools '10, the 3rd International ICST Conference on Simulation Tools and Techniques*, volume vol.53, Brussels, Belgium, Belgium, September 2010.

[134] K. S. J. Pister and L. Doherty. Tsmp: Time synchronized mesh protocol. In *Proceedings of International Sympsiu Distributed Sensor Networks, IASTED, (DSN)*, Orlando,Florida, USA, November 2008.

[135] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proc. ACM SenSys '04*, Baltimore, Maryland, USA., November 2004.

[136] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proc. 2nd international conference on Embedded networked sensor systems (ACM SenSys)*, Baltimore, MD, Nov. 2004.

[137] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proceedings Information Processing in Sensor Networks. IPSN 2005.*, pages 364 – 369, CA, USA, April 2005.

[138] G. J. Pottie and W. Kaiser. Wireless integrated network sensors. *ACM Transactions on Magazine Communications of the ACM*, 43(5):51–58, May 2000.

[139] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava. Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, August 2002.

[140] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. In *Proc. ACM SenSys 03*, Los Angeles, California, Nov. 2003.

[141] I. Rhee, A. Warrier, M. Aia, and J. Min. Zmac: a hybrid mac for wireless sensor networks. *IEEE/ACM, Networking Transactions*, 16(3):511–524, June 2008.

[142] M. Ringwald and K. Romer. Bitmac: A deterministic collision-free and robust mac protocol for sensor networks. In *European Workshop on Wireless Sensor Networks*, pages 1–13, Istanbul, Turkey, April 2005.

[143] D. Rodenas-Herraiz, A.-J. Garcia-Sanchez, F. Garcia-Sanchez, and J. Garcia-Haro. Current trends in wireless mesh sensor networks: A review of competing approaches. *In Sensors 13 (5), doi:10.3390/s130505958*, 13(5):51–58, May 2013.

[144] R. Y. Rubinstein and B. Melamed. *Modern Simulation and Modeling*. Wiley Interscience, New York, 1998.

[145] C. Schurgers, V. Tsiatsis, and M. B. Srivastava. Stem: Topology management for energy efficient sensor networks. In *Proc. IEEE Aerospace Conference 2002*, volume vol.3, pages pp. 1099–1108, March 2002.

[146] D. Sexton, M. Mahony, and M. Lapinski. Radio channel quality in industrial wireless sensor networks. In *Proceedings of Sensors for Industry Conference, SICON'05*, pages 88–94, Houston,TX,USA, February 2005.

[147] Z. Shelby, K. Hartke, C. Bormann, and B. Frank. *Constrained Application Protocol (CoAP)*, February 2011.

[148] X. Shi, G. Stromberg, Y. Gsottberger, and T. Sturm. Micro-frame preamble mac for multihop wireless sensor networks. In *Proceedings of Global Telecommunications Conference, 2004. GLOBECOM '04*, pages 3619–3623, Tx,USA, November 2004.

[149] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan. Physical layer driven protocol and algorithm design forenergy-efficient wireless sensor networks. In *Proceedings of the seventh annual international conference on Mobile computing and networking, MobiCom 2001*, pages 272–287, Rome, Italy, July 2001.

[150] S. Singh and C. Raghavendra. Pamas power aware multi-access protocol with signalling for ad hoc networks. In *Proceedings of ACM SIGCOMM Computer Communication Review, 1998*, pages 5–26, New York, NY, USA, July 1998.

[151] J. So and N. Vaidya. Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing, MobiHoc '04*, pages 222–233, Roppongi, Japan, May 2004.

[152] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications, IEEE*, 7(5):16–27, August 2002.

[153] M. Stemm and R. H. Katz. Measuring and reducing energy consumption of network interfaces in hand-held devices. In *Proceedings of the IEICE Trans. on Communications, vol. E80-B, no. 8*, pages 1125–1131, Los Angeles, California,USA, August 1997.

[154] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proc. ASM SenSys '04*, pages 214–226, 2004.

[155] I. Talzi, A. Hasler, S. Gruber, , and C. Tschudin. Permasense: Investigating permafrost with a wsn in the swiss alps. In *Proc. EmNets '07, the 4th workshop on Embedded networked sensors*, pages 8–12, New York, NY, USA, June 2007.

[156] L. Tan, C. Su, and J. Li. Eers: Energy efficient rate selection approach for wireless battery operated networks. In *Proceedings of Wireless Communications and Networking Conference, WCNC'7*, pages 3424 – 3429, Kowloon, March 2007.

[157] P. Thubert, R. Assimiti, and T. Watteyne. *An Architecture for IPv6 over the TSCH mode of IEEE IEEE802.15.4e*, October 2013.

[158] P. Thubert, T. Watteyne, M. R. Palattella, X. Vilajosana, and Q. Wang. Ietf 6tsch: Combining ipv6 connectivity with industrial performance.

In *Proceedings of the Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS'13)*, pages 541–546, Taichung,China, July 2013.

[159] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In *Proc. ACM SenSys '05*, pages 51–63, 2005.

[160] E. Toscano and L. L. Bello. Cross-channel interference in ieee 802.15.4 networks. In *Proceedings of the IEEE International Workshop on Factory Communication Systems, WFCS'08*, pages 139–148, Dresden, Germany, May 2008.

[161] S. Tschirner, L. Xuedong, and W. Yi. Model-based validation of qos properties of biomedical sensor networks. In *Proc. EMSOFT '08, the 8th ACM International Conference On Embedded Software*, pages 69–78, Atlanta, GA, USA, October 2008.

[162] Y. Tselishchev, A. Boulis, and L. Libman. Experiences and lessons from implementing a wireless sensor network mac protocol in the castalia simulator. In *Proceedings IEEE Wireless Communications and Networking Conference (WCNC'2010)*, pages 1 – 6, Sydney, Ausralia, April 2010.

[163] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh. Power-saving protocols for ieee 802.11-based multi-hop ad hoc networks. In *Proceedings of the IEEE INFOCOM*, pages 23–27, New York,USA, June 2002.

[164] T. van and D. K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proc. ACM SenSys '03'*, Los Angeles, California, USA., November. 2003.

[165] L. van Hoesel and P. Havinga. A lightweight medium access protocol for wireless sensor networks. In *Proc. INSS, 2004*, 2004.

[166] L. van Hoesel and P. Havinga. A lightweight medium access protocol (lmac) for wireless sensor networks: Reducing preamble transmissions and transceiver state switches. *In International Conference on Networked Sensing Systems (INSS)*, June 2004.

[167] A. Varga. *OMNeT++ Discrete Event Simulation System. Available: http://www.omnetpp.org.*

[168] W. Wang, V. Srinivasan, and K.-C. Chua. Using mobile relays to prolong the lifetime of wireless sensor networks. In *Proceedings of the 11th annual international conference on Mobile computing and networking (MobiCom'05)*, pages 270–283, Cologne, Germany, September 2005.

[169] T. Watteyne, S. Lanzisera, A. Mehta, and K. S. Pister. Mitigating multipath fading through channel hopping in wireless sensor networks. In *Proceedings of the IEEE International Conference on Communications (ICC), ICC'10*, pages 1–5, Cape Town, South Africa, May 2010.

[170] T. Watteyne, A. Mehta, and K. Pister. Reliability through frequency diversity: why channel hopping makes sense. In *Proceedings of the 6th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks (PE-WASUN '09)*, pages 116–123, New York,USA, October 2009.

[171] T. Watteyne, M. Palattella, , and L. Grieco. *Using IEEE802.15.4e TSCH in an LLN context: Overview, Problem Statement and Goals*, May 2013.

[172] J. Werb, M. Newman, V. Berry, S. Lamb, D. Sexton, and M. Lapinski. Improved quality of service in ieee 802.15.4 mesh networks. In *Proc. International Workshop on Wireless and Industrial Automation, IWWIA*, San Francisco, California, March 2005.

[173] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7th symposium on Operating systems design and implementation, OSDI '06*, pages 381–396, sealte, WA, USA, November 2006.

[174] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur, and R. Alexander. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*, March 2012.

[175] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the First International Conference on Embedded Network Sensor Systems, SenSys '03*, pages 14–27, Los Angeles, California, USA, November 2003.

[176] I. . working group. *IEEE 802.11.Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: higher-speed physical layer extension in the 2.4 GHz Band. Available: http://grouper.ieee.org/groups/802/11/*, 1999.

[177] Y. Wu, J. A. Stankovic, T. He, and S. Lin. Realistic and efficient multi-channel communications in wireless sensor networks. In *Proceedings of the 27th Conference on Computer Communications, IEEE INFOCOM '08*, pages 1867–1875, Phoenix, Arizona,USA, April 2008.

[178] S.-L. Wut, C.-Y. Lint, Y.-C. Tsengt, and J.-P. Sheu. A new multi-channel mac protocol with on-demand channel assignment for multi-hop

mobile ad hoc networks. In *Proceedings of International Symposium on Parallel Architectures, Algorithms and Networks, 2000. I-SPAN 2000.*, pages 232–237, Dallas, TX ,USA, september 2000.

[179] S.-L. Wut, C.-Y. Lint, Y.-C. Tsengt, and J.-P. Sheul. A new multi-channel mac protocol with on-demand channel assignment for multi-hop mobile ad hoc networks. In *Proceedings Parallel Architectures, Algorithms and Networks. I-SPAN.*, volume 3, pages 232–237, Dallas, TX, December 2000.

[180] G. Xing, M. Sha, J. Huang, G. Zhou, X. Wang, and S. Liu. Multi-channel interference measurement and modeling in low-power wireless networks. In *Proceedings of the 30th IEEE Real-Time Systems Symposium, RTSS'09*, pages 248–257, Washington, DC, USA, December 2009.

[181] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *Proceedings of the 2nd international conference on Embedded networked sensor systems, SenSys '04*, pages 13–24, Baltimore, MD, USA, November 2004.

[182] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proc. IEEE Infocom, 2002*, volume vol.3, pages pp.1567–1576, June 2002.

[183] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 12(3):493–506, June 2004.

[184] W. Ye, F. Silva, and J. Heidemann. Ultra-low duty cycle mac with scheduled channel polling. In *Proceedings of the 4th ACM conference on Embedded network sensor systems, SenSys '06*, pages 321–334, Boulder,Colorado,USA, November 2006.

[185] P. Zand, S. Chatterjea, J. Ketema, and P. Havinga. A distributed scheduling algorithm for real-time (d-sar) industrial wireless sensor and actuator networks. Technical report, Technical Report TR-CTIT-11-09, Centre for Telematics and Information Technology University of Twente, ISSN 1381-3625, Enschede, Netherlands, May 2011.

[186] P. Zand, S. Chatterjea, J. Ketema, and P. Havinga. A distributed scheduling algorithm for real-time (d-sar) industrial wireless sensor and actuator networks. In *Proceedings of the 17th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2012*, pages 1–4, Krakow, Poland, September 2012.

[187] N. Zhang and A. Anpalagan. Sensitivity of swan qos model in manets with proactive and reactive routing: a simulation study. *Telecommunications Systems*, 44(1-2):17–27, June 2010.

[188] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the First International Conference on Embedded Network Sensor Systems, SenSys '03*, pages 1–13, Los Angeles, California, USA, November 2003.

[189] G. Zhou, C. Huang, T. Yan, T. He, and J. A. Stankovic. Mmsn: Multi-frequency media access control for wireless sensor networks. In *Proc. of INFOCOM 2006. 25th IEEE International Conference on Computer Communications.*, pages 1 – 13, Barcelona, Catalunya, Spain, April 2006.