



TECHNISCHE
UNIVERSITÄT
DARMSTADT

DESIGNING SENSOR NETWORKS FOR SMART SPACES

Unified Interfacing and Energy-Efficient Communication
between Wireless Sensor and Actuator Nodes

Vom Fachbereich Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertationsschrift

von

DIPL.-ING. ANDREAS REINHARDT

Geboren am 4. Dezember 1981 in Groß-Gerau

Vorsitz: Prof. Dr.-Ing. Thomas Weiland
Erstreferent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Dr.-Ing. Adam Wolisz
Korreferent: Prof. Dr.-Ing. Matthias Hollick

Tag der Einreichung: 18. Oktober 2011
Tag der Disputation: 12. Dezember 2011

Darmstadt, 2011
Hochschulkennziffer D17

ABSTRACT

Wireless sensor and actuator networks are comprised of embedded systems with sensing, actuation, computation, and wireless communication capabilities. Their untethered character provides installation flexibility and has in consequence led to their application in a large range of domains, e.g. environmental and habitat monitoring, or industrial process surveillance and control. Besides these traditional application areas, the vision of smart spaces foresees the transparent integration of sensing and actuation components into everyday environments. Smart services that rely on information about the current situation and the possibility of physical interaction are envisioned to emerge in versatile ways, such as context-aware building automation or support for ambient assisted living.

From a technological perspective, wireless sensor and actuator networks represent an adequate infrastructure for the realization of smart spaces. As a result of the different application scenarios however, concepts resulting from research on traditional sensor and actuator networks can only be applied to a limited extent. Most prominently, the heterogeneous nature of devices in smart environments necessitates dedicated means to cater for their interoperability. At the same time, the need for small-sized devices entails tight resource and energy constraints, which need to be carefully regarded during application design. Finally, the collection and wireless transmission of data from mobile entities play a vital role in smart environments, whereas they are rarely considered in traditional sensor network deployments.

We address the requirements of smart environments by presenting the Sensor-RPC framework, which enables the generic interoperability between diverse wireless sensor and actuator devices. The presented solution applies the remote procedure call paradigm to abstract from the underlying hardware platforms, i.e. sensing, processing, and actuation functionalities are encapsulated into remotely invocable functions. Sensor-RPC makes use of binary packet representations and a modular parameter serialization concept in order to ensure its efficient applicability on resource-constrained embedded systems.

In order to maximize the utilization of the available energy budget, Sensor-RPC is complemented by Squeeze.KOM, a framework for lossless packet payload compression. Squeeze.KOM takes temporal correlations between successive data packets into account and exploits the observed similarities in order to reduce the size of transmitted packets, and thus the energy demand of their transmission. Depending on the characteristics of the underlying data, the actual data compression step is realized by means of binary distance coding of packet differences, or by applying adaptive Huffman coding with a code tree of limited size. Both take advantage of the specific properties of real-world sensor data sets, in which strongly biased symbol distributions are frequent. Besides the lossless compression of packet payloads, the further reduction of packet sizes by means of header compression is presented. Our stateful header compression mechanism SFHC.KOM omits header fields with constant or deterministically changing values from their transmission by encapsulating them into so called compression contexts. Tailored to its application in smart spaces, SFHC.KOM adapts to the presence of both static and mobile nodes.

The practicality of the devised solutions is investigated through prototypical implementations and the validation of their function on widely adopted wireless sensor and actuator node platforms. We substantiate the evaluations of the presented solutions by detailed analyses of their resource and energy demands. In order to assess the applicability of the contributions in smart environments, real-world data traces from the envisioned application scenario have been collected and extensively used in simulations.

ZUSAMMENFASSUNG

Drahtlose Sensor- und Aktor-Netzwerke setzen sich aus eingebetteten Systemen mit Sensor- und Aktor-Komponenten, sowie einem Prozessor und einer drahtlosen Kommunikationseinheit zusammen. Durch ihren drahtlosen Charakter bieten Sensor- und Aktor-Knoten eine hohe Flexibilität bei der Installation, was zu ihrem Einsatz in einer Vielzahl von Anwendungsgebieten geführt hat, etwa der Erfassung von Umgebungsparametern oder der Überwachung und Steuerung industrieller Prozesse und Anlagen. Neben diesen traditionellen Anwendungsfeldern für verteilte Sensorsysteme hat sich in den vergangenen Jahren zudem die Vision intelligenter Umgebungen, so genannter Smart Spaces, entwickelt. Die diesen zugrunde liegende Intelligenz beruht dabei auf der Nutzung heterogener Sensor- und Aktor-Komponenten, die unsichtbar in das alltägliche Umfeld integriert sind. Hierdurch werden neuartige Dienste ermöglicht, z.B. die Kontext-bewusste Automatisierung von Gebäuden oder die Unterstützung von selbstbestimmtem Leben im Alter.

Aus technologischer Sicht stellen drahtlose Sensor- und Aktor-Netzwerke eine geeignete Plattform für die Realisierung intelligenter Umgebungen dar. Bedingt durch die unterschiedlichen Anwendungsszenarien lassen sich jedoch nicht alle bestehenden Forschungsergebnisse aus dem Bereich der konventionellen drahtlosen Sensornetzwerke direkt auf Smart Spaces übertragen. Insbesondere muss in intelligenten Umgebungen die Plattform-übergreifende Kommunikation zwischen multimodalen und heterogenen Systemen möglich sein, um eine langfristige Erweiterbarkeit sicherzustellen. Mit dem Bestreben nach der Miniaturisierung von Sensoren und Aktoren gehen zudem starke Beschränkungen der zur Verfügung stehenden Betriebsmittel und des Energiebudgets einher. Darüber hinaus sind die Erhebung und drahtlose Übertragung von Daten mobiler Entitäten grundlegend für intelligente Umgebungen, wohingegen diese in klassischen Anwendungsfeldern drahtloser Sensornetzwerke zumeist nicht betrachtet werden.

In dieser Dissertation wird eine wesentliche Voraussetzung für die erfolgreiche Umsetzung intelligenter Umgebungen geschaffen, das Sensor-RPC-Verfahren für die Vereinheitlichung des Zugriffs auf heterogene Sensoren und Aktoren. Sensor-RPC setzt hierbei das Paradigma der entfernten Funktionsaufrufe ein, um Interoperabilität zwischen heterogenen Plattformen zu schaffen. Ein besonderer Schwerpunkt wurde auf den leichtgewichtigen Charakter der Lösung gelegt, um ihre Anwendbarkeit auch auf Ressourcen-beschränkten eingebetteten Systemen bei gleichzeitigem geringen Energiebedarf zu gewährleisten.

Um eine möglichst lange Laufzeit der Sensor- und Aktorknoten zu erzielen, wird Sensor-RPC im Rahmen dieser Arbeit durch verlustlose Datenkompressionsverfahren ergänzt. Die Squeeze.KOM-Kompressionsschicht detektiert zeitliche Korrelationen in Datenströmen, um diese nachfolgend zu komprimieren und somit Energieeinsparungen durch die Reduktion der Betriebsdauer der Kommunikationseinheit zu erzielen. Squeeze.KOM ist speziell auf die Eigenschaften der Symbol-Verteilungen in Sensornetzwerken angepasst und setzt zur Kompression die Distanz-Kodierung von Paketdifferenzen sowie das adaptive Huffman-Coding-Verfahren mit beschränkten Baumgrößen ein. Als Ergänzung zur verlustfreien Kodierung von Paket-Nutzlasten wird in dieser Arbeit ebenfalls die zustandsbehaftete Kompression von Paket-Headern betrachtet. Das vorgestellte SFHC.KOM-Verfahren basiert auf der Verkapselung von Header-Informationen, die sich nicht oder nur deterministisch ändern, in so genannte Kompressions-Kontexte. Statt der kompletten Paket-Header werden bei Anwendung des Verfahrens nur Identifikationsmerkmale der Kompressions-Kontexte ausgetauscht. SFHC.KOM ist speziell auf den Einsatz in intelligenten Umgebungen abgestimmt und sowohl auf fest installierten als auch auf mobilen Knoten lauffähig.

Die Praxistauglichkeit der vorgestellten Beiträge wird durch Proof-of-Concept-Umsetzungen untersucht, in denen insbesondere auch Datensätze aus realen Umgebungen zum Einsatz kommen. Zudem werden im Rahmen dieser Arbeit alle präsentierten Lösungen gezielt auf ihren Energiebedarf hin analysiert.

ACKNOWLEDGMENTS

First and foremost, I would like to thank Prof. Ralf Steinmetz for his supervision and his constant support. I would further like to thank my second examiners Prof. Adam Wolisz and Prof. Matthias Hollick for their valuable feedback.

I would like to express my gratitude for the support from my colleagues in the *Mobile Communications and Sensor Networking* group: Diego Costantini, Farid Zaid, Johannes Schmitt, Matthias Kropff, Parag Mogre, Paul Baumann, and Sonja Bergsträßer. Their help during the final stages of writing this thesis was truly invaluable.

I am grateful for the good atmosphere and team spirit at the Multimedia Communications Lab, and would like to thank all former and current colleagues that have contributed to it. In particular, many thanks to Sebastian Zöller, André König, and Stefan Schulte for their very useful advice and the constructive discussions, as well as Silvia Rödelsperger and Sandra Siebert for their encouraging words, and André Miede for his outstanding \LaTeX support. I also want to thank my students for their contributions to my research and the many interesting and insightful discussions.

Furthermore, I want to thank my family as well as Beena, Jan and Bai-Xiang, Charlotte, Stefan, and Martin for supporting me whenever necessary. Finally, I would like to express my sincere gratitude to you, Delphine, for being part of my life.

Darmstadt, 2011

A.R.

CONTENTS

1	INTRODUCTION	1
1.1	Motivation	1
1.2	Challenges	3
1.3	Goals	4
1.4	Contributions	4
1.4.1	Lightweight Remote Procedure Calls	4
1.4.2	Lossless Payload Compression	4
1.4.3	Stateful Header Compression	5
1.5	Outline	5
2	FOUNDATIONS AND DEFINITIONS	7
2.1	Wireless Sensor and Actuator Networks	7
2.1.1	Platform Operation	7
2.1.2	Networking Concepts	9
2.2	Energy Considerations	10
2.2.1	Definitions Revisited	11
2.2.2	Energy Suppliers	11
2.2.3	Energy Consumers	13
2.2.4	Summary	17
2.3	Smart Spaces	17
2.3.1	Smart Objects and the Internet of Things	18
2.3.2	Context Awareness and Ambient Intelligence	19
2.3.3	Smart Homes and Buildings	19
2.4	Summary	21
3	PROBLEM STATEMENT AND RELATED WORK	23
3.1	Generic Interfacing and Unified Access to Sensors and Actuators	23
3.1.1	Related Work	24
3.1.2	Applicability of Existing Solutions in Smart Spaces	26
3.2	Lossless Data Compression in Wireless Sensor and Actuator Networks	27
3.2.1	Related Work	29
3.2.2	Applicability of Existing Solutions in Smart Spaces	32
3.3	Summary and Problem Statement	33
4	LIGHTWEIGHT REMOTE PROCEDURE CALLS	37
4.1	Requirements and Design Decisions	37
4.1.1	Requirements	37
4.1.2	Design decisions	38
4.2	Lightweight Remote Procedure Calls	40
4.2.1	Data Representation Layer	41
4.2.2	Serialization Modules	42
4.2.3	Message Transport and Error Handling	44
4.3	Evaluation	44
4.3.1	Resource Utilization	45
4.3.2	Message Size	46
4.3.3	Invocation Delay	48
4.3.4	Energy Consumption	49
4.4	Summary	50

5	LOSSLESS PAYLOAD COMPRESSION	53
5.1	Feasibility of Data Compression in Sensor Networks	53
5.1.1	Simulation Setup	53
5.1.2	Results for Synthetic Data	54
5.1.3	Results for Real-World Data	56
5.1.4	Summary	58
5.2	The Squeeze.KOM Compression Layer	59
5.2.1	Compressibility of Sensor Network Data Streams	60
5.2.2	Compression of Packet Differences	62
5.2.3	The Squeeze.KOM Compression Layer	65
5.2.4	Evaluation	69
5.2.5	Summary	76
5.3	Adaptive Huffman Coding with Trimmed Huffman Trees	77
5.3.1	Huffman Coding Revisited	77
5.3.2	Compressibility of Sensor Readings	79
5.3.3	Tailoring Adaptive Huffman Coding to Sensor and Actuator Networks	81
5.3.4	Evaluation	85
5.3.5	Summary	89
5.4	Validation against Real-World Data Sets	89
5.4.1	Data Set Selection	89
5.4.2	The TWiNS.KOM Sensor Network Testbed	90
5.4.3	Characteristics of Real-World Sensor Data	92
5.4.4	Applicability of Lossless Payload Compression on Real-World Data	95
5.4.5	Applicability of Lossless Payload Compression on S-RPC Messages	97
5.4.6	Summary	99
6	STATEFUL HEADER COMPRESSION	101
6.1	Compressibility of Header Fields	101
6.2	Requirements and Design Decisions	103
6.3	SFHC.KOM: Stateful Header Compression	105
6.3.1	Compression States	105
6.3.2	Assignment of Connection Identifiers	106
6.3.3	Compressed Header Structure	108
6.4	Evaluation	112
6.4.1	Compression Gain	113
6.4.2	Energy Consumption	115
6.5	Summary	115
7	CONCLUSIONS AND OUTLOOK	117
7.1	Summary and Conclusions	117
7.2	Outlook	119
	BIBLIOGRAPHY	121
	LIST OF ACRONYMS	141
A	APPENDIX	143
A.1	Notations of Squeeze.KOM	143
A.2	Environmental Monitoring and Body Area Sensing Data Traces	143
A.3	Complete Data Traces Collected in an Office Environment	146
A.4	SFHC.KOM Packet Types and Corresponding Header Field Values	148
B	CURRICULUM VITÆ	149
C	AUTHOR'S PUBLICATIONS	151
D	ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG	155

INTRODUCTION

It is the job that is never started
that takes longest to finish.

J.R.R. Tolkien

1.1 MOTIVATION

The increasing miniaturization of Integrated Circuits (ICs) and Micro-Electro-Mechanical Systems (MEMS) has enabled the creation of small embedded devices with sensing, computing, and radio transmission capabilities [KW05]. The combination of these functions into a single platform has opened versatile new application domains, ranging from environmental monitoring to industrial automation and the creation of ambient intelligence. Networked embedded sensing devices represent the core components of such solutions. Their compact size allows them to be deployed close to the origin of the physical phenomena they monitor. Their local processing power can be utilized to extract relevant information from the sensed data. An on-board wireless communication interface combines the capabilities of exchanging sensor readings with the flexibility of untethered operation. Finally, a portable energy supply, commonly realized in the form of batteries, caters for deployment flexibility. Right from the start, these devices were envisioned to shrink continuously, eventually reaching sizes at which they are invisible to the human eye. As a result, these embedded sensing systems are often referred to as *motes* [WLLP01], which the Oxford dictionary defines as “a tiny piece of a substance” [Oxf11]. Networks of motes are commonly referred to as Wireless Sensor Networks (WSNs), which have been identified as one of the top ten emerging technologies for the last decade in [MIT03] and are contained in Gartner’s hype cycle for emerging technologies [Bro11]. Their finite energy budgets however make the use of energy-efficient hard- and software solutions a necessity. As a matter of fact, energy efficiency is deemed one of the grand challenges in sensor networks research [AV10].

The extensible nature of motes also allows them to be augmented by actuation capabilities, effectively turning WSNs into Wireless Sensor and Actuator Networks (WSANs) [AK04]. The resulting possibility to interact with the physical world creates potential for actuation based on different triggers, such as sensor measurements from both local and remote devices, or external commands from coordinating nodes. Actuation capabilities play an important role in many scenarios, e.g. industrial control systems, home automation, or pervasive computing [PGP⁺00]. The rise of *participatory sensing*, which utilizes mobile devices to capture sensor data about their owners and the environment [BEH⁺06], is no replacement for the aforementioned WSN technology. Instead, both sensing solutions complement each other. Participatory sensing systems are typically deployed on mobile devices (such as smartphones or tablet computers), which are carried by a single user. In consequence, all available sensor readings are correlated to the user or his current environment. Users may refrain from contributing sensor data for several reasons, such as degraded user experience (e.g. reduced battery lifetime) or privacy concerns [CRKH11]. In contrast, WSNs are generally composed of fixed sensor installations, which continuously provide information about their surroundings. The perpetual availability of sensor data, even in the absence of users, and the low device cost thereby represent the major benefits of sensor deployments. Furthermore, the fixed installation of WSAN nodes provides interaction capabilities with the physical environment which are unthinkable on mobile participatory sensing devices. In conclusion, even user-centric application scenarios, such as ubiquitous computing, benefit from the combination of both user-oriented mobile sensing devices and the installation of a WSAN infrastructure, which provides a broad sensor base with versatile actuation capabilities.

1.2 CHALLENGES

For the realization of smart environments, comprised of millions of invisible computers, solutions to many practical issues remain to be found. Compilations of technological and social prerequisites for the successful instantiation of ubiquitous computing and smart spaces have been presented in various publications, including [Wei93] and [EGo1]. Challenges in the directions of user interaction, social aspects, or privacy are beyond the scope of this thesis, and thus not discussed in detail. In contrast, the fundamental computation and communication aspects determined in the aforementioned references are summarized in the following list:

- ▷ **LOW-POWER HARDWARE** Ubiquitous computing devices need to sustain operation for extended periods of time, often without the possibility of recharging their energy supply during operation. Low-power hardware components are thus necessary to achieve practical node runtimes, and in consequence widely employed on embedded sensing and actuation systems.
- ▷ **ENERGY-EFFICIENT COMMUNICATION** Wireless communication is a main characteristic of WSANs, but radio transceivers are generally leading the list of energy consumers on current node platforms (cf. Sec. 2.2.3). Energy-efficient communication between embedded devices thus needs to be addressed on all levels of the communication stack, ranging from low-power physical layer implementations and energy-efficient Medium Access Control (MAC) protocols to lightweight networking and transport protocols. On almost all current mote platforms, the operation of their Microcontroller Unit (MCU) to process data locally is often significantly less energy-consuming than the transmission of the same amount of data over the wireless link [DWBPo1, BAo3]. Prior to radio transmissions, local data processing to reduce the volume and/or number of transmissions is thus strongly favored.
- ▷ **EFFICIENT COMPUTATIONAL METHODS** The necessity of low-power hardware and energy-efficient communication also requires a critical examination of the applied data processing algorithms, with the overall design target being the optimally energy-efficient exploitation of the hardware capabilities. In order to attain energy savings and thus extend a node's lifetime, the amount of energy required to perform the data processing step must not exceed the amount required to wirelessly transfer the unprocessed data. The computational intensity of traditional data processing algorithms often disqualifies them from being applied on motes. The integration of adapted mechanisms to optimally process data prior to their transmission therefore represents a further requirement to the successful realization of long-lived smart environments.
- ▷ **IMPROMPTU INTEROPERABILITY** Mechanisms to create interoperability between heterogeneous nodes are required due to the inherent multimodal character of sensing and actuation devices in smart environments. Only by means of abstracting from the underlying hardware and the enablement of network-wide communication across platforms, the seamless interaction between heterogeneous devices and the integration of new devices into existing systems are enabled.

Many approaches towards the realization of subsets of these goals have been addressed and partly achieved in research already. The demand for embedded sensor platforms with wireless communication capabilities and low energy consumption has resulted in many prototypical hardware systems, ranging from tablets, pads, and pens devised at Xerox PARC [Wei93] to the current de facto standard Mica [HC02] and Telos [PSC05] platforms for embedded sensor networks research. Although still far from invisible, both platforms enable rapid WSAN prototyping by allowing developers to flexibly adapt both hard- and software to their needs. Secondly, a wide range of communication aspects have been intensely investigated, e.g. MAC protocols for fair and energy-efficient communication [LM10], or the applicability of the Internet Protocol (IP) in sensor networks [DAV04]. Simultaneously, efficient methods for local

and network-wide data processing [MFHH02, FRWZ07] and routing [IGE00] have emerged to maximize the network lifetime. Many researchers have also addressed the interoperability between heterogeneous and multimodal sensor and actuator nodes [RE07, HCo8]. Although each of these contributions has enriched the research on WSANs, none represents a holistic solution that encompasses all of the points in the list. Instead, the proposed systems and architectures generally focus on individual items from the above list only.

1.3 GOALS

The objective of this thesis is the creation of building blocks for the realization of smart spaces. In contrast to existing research, in which the above challenges are mostly regarded individually, this work is primarily motivated by their effective combination. As a result, the following goals are addressed in this thesis:

- ▷ The enablement of *interoperability* between embedded sensing and actuation systems based on a suitable abstraction of hardware functionalities.
- ▷ The *energy-efficient transport* of messages in WSANs by reducing the size of transmitted packets, and thus the time during which the radio transceiver needs to be active.
- ▷ Ensuring the *applicability* of the devised solutions on real mote hardware by taking the characteristics of low-power embedded systems into consideration.

1.4 CONTRIBUTIONS

The contributions of this thesis are motivated by the aforementioned challenges for the integration of wireless sensor and actuator nodes into smart environments, and can be summarized as follows.

1.4.1 *Lightweight Remote Procedure Calls*

Sensor-RPC (S-RPC) [RMS11], the first contribution of this thesis, is a lightweight implementation of the remote procedure call paradigm. It provides the possibility for interaction between multimodal embedded sensing and actuation systems by resolving platform dependencies. Specifically adapted to the demands of WSAN nodes, it is based on an abstraction layer which allows functionality to be consumed from remote devices independent of their actual realization. Key features of S-RPC include its modular data type serialization concept and the use of binary packet headers in order to minimize the required overhead. The realization of S-RPC is evaluated both in simulation and practical experiment in order to confirm its applicability on sensor and actuator nodes.

1.4.2 *Lossless Payload Compression*

In the second contribution of this thesis, application-independent algorithms for the lossless compression of packet payloads are devised and evaluated for real-world applicability. Energy-efficient operation is achieved by optimizing the tradeoff between the required computational efforts and the energy required for the radio transmissions. The following two major contributions to the area of packet payload compression in WSANs are presented:

- ▷ The Squeeze.KOM framework [RHS09] exploits temporal correlations of real-world sensor data and presents an approach for lossless compression of packet payloads. Through the transmission of encoded packet differences, message sizes are reduced while the original data can be reconstructed at the receiver side without loss of precision. Recently transmitted packets are stored in a local history buffer of variable size, which allows the developer to trade higher compression gains for increased memory consumption.

- ▷ Entropy coding of packet payloads is addressed in the presentation of the adaptive Huffman coding algorithm tailored to operation in WSNs [RCH⁺10]. Based on the observation that both the storage and traversal of Huffman trees have high resource demands, Huffman trees of reduced size are being used in the devised solution. Energy-efficient operation is ensured by choosing the tree size based on the energy demand for its maintenance and traversal.

In order to prove the applicability of the presented solutions, data sets collected from real-world sensor deployments are being used for all evaluations. Furthermore, the Testbed for a Wireless Network of Sensors (TWiNS.KOM) [RKHS08] has been set up as an enabler for the data collection and experimental validation of simulated results in a real indoor WSN deployment.

1.4.3 Stateful Header Compression

Besides the compression of packet payloads, a method for the stateful compression of packet headers in IEEE 802.15.4-based WSNs is presented. It exploits the fact that most header fields used in communication between two WSN nodes either remain completely static or only change in a deterministic manner. The proposed SFHC.KOM [RMKS10] approach maintains state information for each communication link at both participants in the form of compression contexts, which are referred to by unique connection identifiers. Once a context has been established, the headers of all successive packets are then replaced by its connection identifier.

1.5 OUTLINE

The remainder of this thesis is structured as follows:

Chapter 2 “*Foundations and Definitions*” presents a detailed introduction to the foundations and definitions in the field of WSNs. Besides highlighting the multimodality of available sensor and actuator devices, the chapter also focuses on the suppliers and the consumers of energy in sensor networks and motivates the need for energy-efficient operation. Finally, an introduction to the application domain of smart spaces is presented.

Chapter 3 “*Problem Statement and Related Work*” summarizes work related to the fields of unified interfacing of WSN nodes and energy-efficient data transfer. After assessing the applicability of the surveyed solutions in the application area of smart spaces, the problem statement, which represents the foundation for the contributions of this thesis, is refined.

Chapter 4 “*Lightweight Remote Procedure Calls*” presents the lightweight and extensible S-RPC framework, catering for platform-independent access to sensing and actuation functionality by abstracting from the underlying hardware platform. S-RPC only entails a small energy overhead, while its modular concept enables the integration of a wide range of platforms.

Chapter 5 “*Lossless Payload Compression*” introduces the modular Squeeze.KOM data compression framework for WSN nodes and its extensions for stream-oriented data compression based on differential coding as well as adaptive Huffman coding with reduced tree sizes. Both compression mechanisms target the improvement of the node’s energy efficiency by exploiting the linear relation between packet payload sizes and the energy demand of their transmission. Comprehensive studies with real-world data are performed in order to quantify the extent of achievable energy savings.

Chapter 6 “*Stateful Header Compression*” supplements the payload compression solutions by the SFHC.KOM scheme for stateful header compression in WSNs. The size of packet headers is reduced by the omission of constant fields and the efficient encoding of header fields that only change in a deterministic way.

Chapter 7 “*Conclusions and Outlook*” concludes this thesis with a summary of the presented contributions and compares the attained solutions to the established research goals. Finally, an outlook outlines possible future extensions to this work.

Knowing is not enough; we must apply.
Willing is not enough; we must do.

Johann Wolfgang von Goethe

This chapter presents an introduction to WSAWs to define the foundations on which the contributions of this thesis are based. First, a description of the general architecture of wireless sensor and actuator nodes is provided, and supplemented by a brief introduction about the networking concepts applied in WSAWs. Subsequently, energy sources and consumers as well as approaches to extend the battery life of WSAW nodes are identified, as energy is known to be a scarce resource on battery-operated nodes. Finally, the foundations of smart spaces and ambient intelligence, as described in Mark Weiser's vision [Wei91], are revisited in order to outline the scope in which the contributions of this thesis are placed.

2.1 WIRELESS SENSOR AND ACTUATOR NETWORKS

Wireless sensor and actuator nodes are embedded hardware platforms typically comprised of a MCU, a radio transceiver, sensor devices, and actuation components [ASSC02]. Motes are typically operated on a finite energy budget, and thus comprise components with low energy consumption to enable long-lasting operation. In contrast to wired sensor installations, the use of wireless sensor devices provides great installation flexibility and ease of deployment.

2.1.1 Platform Operation

A mote's general operation comprises the sampling and processing of sensor data, wireless networking to exchange processed data, and actuation based on local or remote sensor readings [AK04]. A typical data flow is indicated in Fig. 2, in which the following operations are executed:

1. **SENSOR DATA SAMPLING** Characteristics of the physical environment are captured using sensor devices. The current value of the sensed physical phenomenon is retrieved from either integrated or peripheral sensors and transferred to the MCU, on which it is stored in memory or forwarded to an external storage component. A broad range of sensors can be interfaced to current mote platforms thanks to the versatile expansion options of current MCUs.
2. **LOCAL DATA PROCESSING** Local processing is commonly applied in order to reduce the amount of data that needs to be transmitted over the wireless channel. Various operations can be performed on the sampled readings, ranging from simple aggregation tasks to reduce the number of individual data transmissions [KEW02] to more complex operations like digital filtering [BTZ06] or processing based upon the data that were received from other nodes in the network [CPR03].



Figure 2: Typical data flow in wireless sensor and actuator networks

3. **WIRELESS COMMUNICATION** If a packet transmission is still deemed necessary after the processing step, the generated packet payload is prepared for transmission by the network stack installed on the MCU. Subsequently, the data is forwarded to the radio transceiver which caters for the data transmission over the wireless channel. WSNs often feature dedicated sink nodes, which collect the sensor readings and generate actuation commands.
4. **ACTUATION** While purely sensor-based networks are confined to the three previous steps, motes with actuation devices provide the possibility to interact with the physical world. Actuation can either be based on rules which are locally applied to the sensor data, or on remote triggers which are received over the wireless communication channel.

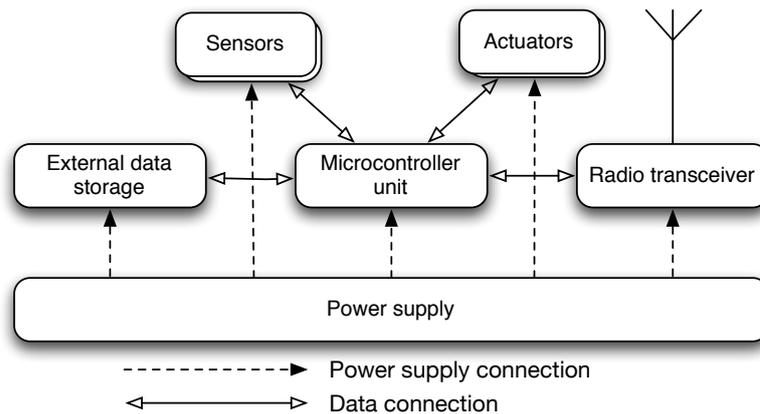


Figure 3: Common structure of a wireless sensor and actuator node

From a technological perspective, the realization of the presented logical flow of sensor information necessitates the hardware components depicted in Fig. 3. In general, a mote platform is composed of the following major components:

- ▷ **MICROCONTROLLER UNIT** The MCU represents the core element of a mote and features connections to all on-board peripherals, including sensors, actuators, data storage, and the radio transceiver. Due to the tight energy constraints present on mote platforms, low-power MCUs are almost exclusively used, operated at low clock speeds and confined to a limited set of resources.
- ▷ **RADIO TRANSCEIVER** As untethered operation is an essential characteristic of WSNs, data transfer between motes is generally realized over a wireless communication channel. Low-power radio transceivers operating at low data rates are prevalent on current motes in order to cater for the limited energy budget.
- ▷ **EXTERNAL DATA STORAGE** Besides the limited amount of Random Access Memory (RAM) present within the MCU, many motes comprise a secondary non-volatile storage device. Often significantly larger in size than the memory of the MCU, the external data storage can be used to collect larger amounts of data before performing processing tasks.
- ▷ **SENSORS** Sensors represent the primary source of raw data in WSNs and can be interfaced to the MCU either over analog or digital interfaces. Many sensors output analog voltages that need to be converted to the digital domain by means of Analog-to-Digital Converters (ADCs). Other sensors are supplied with digital interfaces, which allow for a straightforward connection to the MCU over on-board buses.
- ▷ **ACTUATORS** Actuators with binary states (i.e. on or off) can be directly controlled by digital output pins. Many current MCUs also permit the generation of frequency outputs, Pulse-width Modulation (PWM) signals, or analog output voltages. Further actuation signals may also be realized through the use of dedicated external hardware.

- ▷ **POWER SUPPLY** Local energy sources are employed in order to cater for the energy required for a mote's operation. Most often realized in the form of batteries, the energy supply is vital to all components on the mote. As most current motes lack dedicated support to recharge their battery during runtime, energy-efficient operation of both hardware and software is mandatory in WSNs.

In order to clarify the correlation between the application scenario and the hardware components, the high-level description of the logical data flow presented in Fig. 2 can be rephrased in technical terms as follows: A sensor with analog interface is connected to an ADC input of the MCU, from which it is being polled periodically. The latest readings are written to the non-volatile external data storage device to allow for later processing. Additionally, the latest reading is used to calculate statistics about the monitored phenomenon. A comparison of the statistics to pre-defined threshold values leads to the generation of an event message, which the MCU assembles into a radio packet and forwards to the radio transceiver. The packet is transmitted on the wireless communication channel to a control unit. As a response, a trigger message for actuation may be received from the controller station, upon which the corresponding actuation output is enabled.

2.1.2 Networking Concepts

Although the deployment of independent sensing devices to monitor environmental phenomena may be expedient, the major functionality gain of WSNs results from the wireless networking between motes. When multiple sensor and actuator nodes are deployed within the radio range of each other, data can be forwarded to its destination across multiple hops. In contrast to fixed sensor installations, the untethered character of wireless sensor nodes provides the flexibility to place sensors close to the origin of the monitored phenomenon. The distributed processing capabilities allow the network to autonomously establish routes for radio traffic, perform in-network processing, and cater for self-configuration. To illustrate this, a typical WSN topology, adapted from the PermaSense sensor network deployment in the Swiss Alps [BGH⁺09], is depicted in Fig. 4. The numbers within the nodes represent the physical

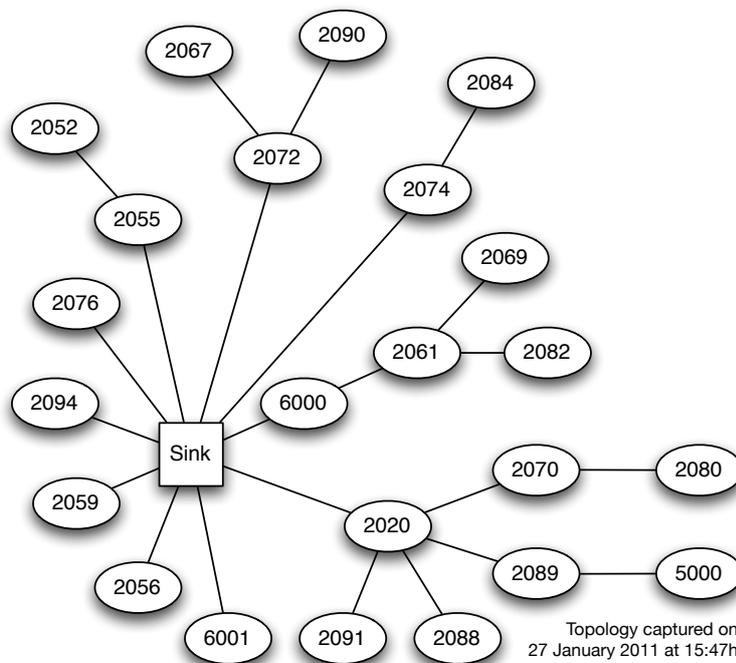


Figure 4: Topology of the PermaSense deployment on the Matterhorn

device addresses of the underlying hardware platforms. The WSN is composed of a number of sensor nodes which collect and transmit the sensor readings, and a single *sink* node to which all data is forwarded. Intermediate nodes on the route between sources and the sink primarily cater for the relaying of information, but can also be configured to aggregate and process incoming sensor data before their retransmission. The flow of data in the depicted sensor network follows a unidirectional many-to-one communication pattern, frequently referred to as *convergecast* [AGSo3], in which all data transmissions are directed and converging towards the sink. Tree-based routing protocols, such as the Collection Tree Protocol (CTP) [GFJ+09], are prevalently used in convergecast sensor networks, as sensor nodes need to store few routing information only. The spanning tree established and maintained in CTP is based on a periodic evaluation of the link quality to neighbor nodes. Packets are routed to the sink node by relaying them to the node's parent until they have reached their destination. The resulting converging character of the topology also allows for efficient in-network processing, i.e. processing both locally sampled and received data before retransmitting a packet to the sink.

The versatility of WSNs represents one of the main drivers for their widespread application in many different application domains. First real deployments of static sensor networks were realized to monitor environmental phenomena, such as animal populations [MCP+02], volcanic eruptions [WAJR+05], or the characteristics of glaciers [MOHo4] and permafrost areas [BGH+09]. Already in 2006, more than 50 environmental monitoring experiments were deployed in real-world settings [HMo6]. WSNs are also increasingly replacing wired installations in industrial monitoring applications [KAB+05, SBR10]. Suitable security concepts and mechanisms to provide Quality of Service (QoS) guarantees are applied, enabling the deployment of wireless sensors with high availability and reliability [Mei09]. In recent projects, motes have also been deployed within transportation containers [JML10], where they provide real-time tracking of goods and inventory management.

Besides the monitoring of environmental phenomena by statically deployed wireless sensors, research has also been conducted on the monitoring of mobile entities, e.g. by attaching sensors to animals like in the ZebraNet project [JOW+02]. Location traces and physiological parameters are being captured and relayed by other nodes on their path to the sink. The field of participatory sensing [CEL+06], albeit based on using mobile phones as sensing devices, is a second prominent application domain for mobile sensor systems. Sensors are also widely used in the fields of Ambient Intelligence (AmI), Ambient Assisted Living (AAL), and smart homes and buildings, where an extensive set of data about the current context of people and objects is used [LTS07, GHR+05]. Especially within the vision of the Internet of Things (IoT) [Int05b], ubiquitous wireless sensing and actuation devices are envisaged to be found in everyday objects, where they provide current status information as well as capabilities to interact with the physical environment.

2.2 ENERGY CONSIDERATIONS

Current mote platforms are typically fitted with low-power components to enable long-lasting operation on a finite energy supply [PK00], commonly present in the form of batteries. Additionally, the wireless nature of motes disallows the use of tethered power supplies, and both size and weight of the energy supply are additionally often restricted by the application scenario. A general constraint for algorithms and mechanisms is thus the necessity for resource-efficient operation, both with regard to the required energy and the demand for memory and computational power. To put the power consumption of components on mote platforms into perspective, the available energy budgets and the prevalent technological realizations of the components identified in Sec. 2.1 are discussed. Before dissecting a mote into its components, we however provide a brief recapitulation of the definitions of electrical energy and power.

2.2.1 Definitions Revisited

For a better understanding of the following analysis, let us briefly revisit some definitions according to the foundations of physics to avoid ambiguities and cater for a consistent terminology throughout this thesis. All definitions are based on the assumption of Direct Current (DC) operation, which is the general case in WSNs.

- ▷ **POWER** Electric power (denoted by the letter P) is generally measured in watts (W). The power consumption of a device is calculated as $P = U \cdot I$, where U is the potential difference (i.e. the voltage drop, measured in volts) across the device and I is the electric current (i.e. the rate of flow of electric charge, measured in amperes) through the device.
- ▷ **ENERGY** Electric energy (denoted by E) is measured in joules (J) and defines the amount of power consumed; applying the given definition of power above, the equation $E = P \cdot t$ defines the consumed (or generated) energy. Energy thus represents the available budget that can be consumed, while power defines the rate at which it is being consumed.
- ▷ **CHARGE** The amount of energy that can be generated by a battery is generally expressed through the stored electric charge (Q), which is often measured in ampere-hours (Ah), eliminating any dependency on the voltage at which the energy is provided.

2.2.2 Energy Suppliers

In order to assess the impact of consumers of electrical energy, let us first compare the energy budgets available on current mote platforms. A thorough discussion of the large number of mote platforms that has resulted from WSN research would exceed the scope of this thesis, hence the analyzed platforms are confined to a selection of widely used motes, concisely the BTnode rev3 [ETH05], Cricket [MEM05], EyesIFX v2.1 [HLo5], Imote2 [MEM06a], Mica [HCo2], Mica2dot [Cro02], MicaZ [MEM07], Shimmer [Shi10], SunSPOT [Sun08], TelosB [PSC05], and TinyNode [DFMFMo6] platforms. For a comprehensive overview of mote platforms used in research, the reader is referred to the Sensor Network Museum [TIK11].

In Table 1, the nominal power supply options for each of the mote platforms are listed and compared. The table, in which all figures have been taken from the manufacturer data sheets, confirms the omnipresence of battery-driven mote operation. The predominant battery types are primary cells in AA and AAA form factors as well as lithium coin cells (indicated by the prefix CR), with corresponding energy budgets ranging from 3,500 to 20,000 joules. With availability in large numbers, standardized sizes and voltages, primary cells are prominent and widely available sources of electrical energy. Suffering from very low self-discharge only, they are well suited for long-term applications. Based on a non-reversible chemical reaction however, primary cells cannot be recharged, and the node lifetime is thus inherently limited

Table 1: Comparison of selected mote battery types and nominally carried charge and energy

MOTE PLATFORM(S)	BATTERY TYPE(S)	VOLTAGE	STORED CHARGE	STORED ENERGY
Mica, MicaZ, TelosB, BTnode rev3, Cricket	2x AA	3.0 volts	~2000 mAh	~20,000 joules
Imote2	3x AAA	4.5 volts	~900 mAh	~15,000 joules
EyesIFX v2.1	1x CR2477	3.0 volts	1000 mAh	~10,000 joules
TinyNode	1x CR14335	3.6 volts	800 mAh	~10,000 joules
SunSPOT	Li-Ion rechargeable	3.7 volts	720 mAh	~9,500 joules
Mica2dot	1x CR2354	3.0 volts	560 mAh	~6,000 joules
Shimmer	Li-Ion rechargeable	3.7 volts	280 mAh	~3,500 joules

by the charge available on the battery. Approaches to overcome these limitations have made use of battery-friendly discharge patterns, which have shown slight improvements to the node runtime [MWW11]. Various alternative options to provide motes with energy have been explored recently, although none of these energy sources has yet evolved to a sufficient extent to replace batteries on current motes. The following list briefly summarizes these alternative energy sources.

RECHARGEABLE BATTERIES Although rechargeables (secondary cells) present an alternative to the use of primary cells, they lack applicability in many WSNs, as current motes rarely provide dedicated support for recharging the batteries during their deployment. When rechargeable batteries cannot be charged during a WSN deployment, they can be considered analog to conventional primary batteries. Rechargeable batteries are readily available in various kinds, but often suffer from measurable self-discharge ranging from 5% to 30% per month as well as conversion losses in the charging process [LR01]. Secondary cells are already employed on a small range of WSN nodes to store harvested energy, including [PC04, RKH⁺05].

ENERGY HARVESTING The extraction of electric energy from the environment, termed energy harvesting, is an alternative option to provide electrical energy to a WSN node. The basic assumption behind energy harvesting is that energy is present in versatile forms in the environment (such as sunlight or wind), and can be extracted using converters. Current technological trends in the field of energy harvesting are briefly presented in the following list:

- ▷ **VIBRATIONS** Vibration harvesters have been constructed in various projects to extract energy from the vibration of machinery and power embedded systems, such as [BTK⁺04, LYLL03]. Depending on the vibration frequency, different harvester technologies are being used, exploiting inductive, electrostatic, or piezoelectric effects. Vibration harvesters can provide electrical power from $20\mu\text{W}/\text{cm}^3$ to $400\mu\text{W}/\text{cm}^3$ [AC98]. Besides vibrating infrastructure, piezoelectric harvesting can also be applied in the field of human kinetics [Sta96, KKPG98], offering the capability to power a body area sensor using energy generated from piezo elements. Up to $1300\mu\text{W}$ of output power were measured for a shoe sole inlay of 5cm by 5cm in size [SP01].
- ▷ **SOLAR RADIATION** As outdoor deployments represent a prevalent class of application scenarios for WSNs, the use of solar panels to harvest energy becomes feasible. When compared to other means of energy harvesting, solar cells provide high energy outputs of up to $15\text{mW}/\text{cm}^3$ at noon when directly aimed towards the sun [Rou03]. However, the harvestable energy reduces to less than $10\mu\text{W}/\text{cm}^3$ for the same panel size in an indoor setting with indirect office lighting. Solar panels are widely available, and researchers have developed sensor nodes that utilize solar power in conjunction with energy storages, e.g. the PUMA [PC04] and Helimote [RKH⁺05] platforms. Practical deployments of solar-powered sensor nodes include ZebraNet [ZSLM04], where nodes were powered from a set of solar panels mounted to collars worn by zebras, and the Trio testbed, which uses 557 solar-powered nodes [DHJ⁺06].
- ▷ **TEMPERATURE DIFFERENCES** Thermal energy harvesting is the extraction of electrical energy from materials through temperature changes or differences. Pyroelectric materials generate a voltage when their temperature changes, i.e. when they are heated or cooled. In contrast, thermoelectric harvesters apply the Seebeck effect, converting temperature differences between two dissimilar conductors into an electric voltage. This renders them applicable in locations where constant temperature differences are present, e.g. exhaust vents or industrial machinery [RMW⁺97]. The efficiency of thermal harvesters strongly depends on the temperature difference, yielding output powers between a few microwatts and tens of milliwatts. In practice, thermal energy harvesters are mounted to the fuselage of aircrafts to provide information about the current structural health [BKS⁺08]. For devices with very small energy demand, such as motes in a body area sensor network, body warmth is also being used to generate electric energy for their operation [HTCP09].

ENERGY STORAGES When the amount of energy extracted or harvested from the environment exceeds the demand of the sensor node, energy storages can be employed to buffer the excess energy. Later, when only an insufficient amount of energy can be harvested from the environment, these storage elements can provide their buffered charge to the attached node to sustain its operation. As the output of many harvesting sources can rarely be predicted, storage devices are widely employed, ranging from small capacitors that keep a very low charge to power devices for only tiny fractions of a second, up to large batteries that enable operation for long periods of time.

Although energy harvesting provides viable means to supply motes with electric energy, only a small number of prototypical mote realizations rely on the use of harvesters in conjunction with storage devices. The majority of current motes, as already determined in Table 1, is powered by batteries, which has proven to be a viable approach for sensor network prototyping, because they enable long lifetimes and high energy densities at comparably low cost. Although the use of primary cells entails maintenance costs for changing the batteries upon their depletion, their reliability and convenience still manifests them as the de facto energy supply for motes.

2.2.3 Energy Consumers

As shown in the overall structure of a WSN node in Fig. 3, every component of the mote requires a connection to the power supply. In order to quantify the power consumption of the devices employed on current sensor nodes, the components present on the node platforms analyzed in Table 1 are discussed in detail. All components are presented with special regard to their power consumption, and techniques to reduce their power demand are outlined.

MICROCONTROLLER UNIT The central device on any mote is its MCU. In contrast to general purpose microprocessors, which offer high computational power, MCUs on WSN nodes are designed with a focus on low energy consumption. They hence integrate RAM and Flash storage to store the application software locally and reduce the amount of external components required by comprising further hardware components, such as freely configurable General Purpose Input/Output (GPIO) pins or dedicated Universal Synchronous/Asynchronous Receiver Transmitters (USARTs) for different protocols. Additionally, some MCUs feature integrated ADCs to convert external analog sensor readings to the digital domain.

Table 2 compares the power consumptions of the MCUs for the selected mote platforms in different operation modes. Current motes primarily feature the Atmel ATmega128L and Texas Instruments MSP430 series microcontrollers. The widely used Texas Instruments MSP430F1611 microcontroller provides 48kB of program memory, 10kB of RAM, and 48 GPIO pins [Tex09]. Additionally, the device features five power-saving modes, Direct Memory Access (DMA) capabilities, analog-to-digital and digital-to-analog converters, and two on-board USARTs

Table 2: Power consumption comparison of MCUs currently employed on motes

MOTE PLATFORM(S)	MCU TYPE	POWER CONSUMPTION		
		ACTIVE MODE	IDLE MODE	SLEEP MODE
EyesIFX v2.1, TelosB, TinyNode, Shimmer	Texas Instruments MSP430	1.5 mW	225 μ W	0.6 μ W
MicaZ, Mica2dot, BTnode rev3, Cricket	Atmel ATmega128L	15 mW	6 mW	45 μ W
Mica	Atmel ATmega103L	16.5 mW	4.8 mW	3 μ W
SunSPOT	Atmel ARM920T	57 mW	27 mW	3 mW
Imote2	Marvell PXA271	170 mW	138 mW	17.6 mW

supporting diverse serial communication protocols. Similarly, Atmel’s ATmega128L has 128kB of application memory, 4kB of RAM and 53 GPIO pins [Atm10]. It is provided with six sleep modes, several analog input and output channels, and two USARTs. Both are designed for an operating voltage of 3.3 volts. From the table, it becomes clear that the prevalent MCUs types are primarily selected by their power consumption. Based on the stored charge listed in Table 1, the MCU on the TelosB can operate up to 4,000 hours in active mode and 10 million hours in sleep. The more computationally powerful SunSPOT can operate for 45 hours when its MCU is active, and last up to 900 hours in sleep mode. With special consideration on the low current consumption during operation and especially during sleep mode, MCUs are often clocked at low frequencies. In simplified form, the power dissipation of a MCU is proportional to the four factors shown in Equation 1, where α represents the activity factor (i.e. the fraction of transistors that change states), C defines the load capacitance, f the clock frequency, and U the supply voltage [CSB92].

$$P(f, U) = \alpha \cdot C \cdot f \cdot U^2 \quad (1)$$

The power demand of a MCU can thus be minimized by reducing each of the factors in the equation:

- ▷ By reducing the number of transistors that need to switch states, the activity factor α can be lowered. In [HM99], a gray code (i.e. a code in which only a single bit changes for each value increment) is used on the address bus, effectively leading to 25–30% less switching activity and overall energy savings of 10%.
- ▷ Because the parasitic capacitances of transistors depend on the size of the active area, the use of smaller transistors with lower switching capacitances has been shown to be a viable approach to reduce the power demand [CSB92].
- ▷ Although the number of instructions executed per time unit linearly depends on the clock frequency, energy can be preserved by operating the device at low clock frequencies [APS⁺09], or making use of dynamic frequency scaling which adapts the operating frequency to the current load [MSG⁺05].
- ▷ As the operating voltage U has quadratic impact on the power demand, its reduction is a vital step towards reducing the device’s total consumption. Techniques like dynamic voltage scaling [BPSB00] and sub-threshold operation [WCK02] are viable steps towards reducing the impact of the operating voltage on the power consumption.
- ▷ Finally, also combinations of the aforementioned methods are viable, such as frequency and voltage scaling combined in [HKH01].

There is also ongoing research on MCUs with ultra-low power dissipation [SHL⁺08, KRV⁺08, WP04], although no products are readily available on the markets today. These devices dedicatedly lack extended hardware features and aim for very small operating frequencies to enable operation on low currents. However, their applicability in sensor network prototyping is very limited due to their lack of extensibility through GPIO pins.

SENSING DEVICES Sensing functionality represents a core component in WSNs, and consequently sensing devices are present on virtually all mote platforms. Devices range from simple temperature or light intensity sensors to sophisticated Micro-Electro-Mechanical System (MEMS) devices which can e.g. capture acceleration or rotational forces.

Sensors with analog signal outputs exist for a variety of sensing modalities. To process their data, the conversion of the analog reading into the digital domain is necessary. This task can be easily implemented on most current MCUs through integrated ADC inputs. Different technological realizations of sensors with analog interfaces are viable, including resistive sensors, discrete semiconductor-based sensors, and sensors with active electronics which convert the sensor reading to an analog voltage or current output. On many current motes,

sensors with analog interfaces are present. The National Instruments LM61 temperature sensor, which outputs a voltage linear to the temperature, is based on a thermistor and present on the eyesIFXv2.1 mote, on which it consumes $400\mu\text{W}$ in normal operation mode. The Shimmer node employs a Freescale MMA7260Q triple axis accelerometer with analog outputs of the voltage, which consumes 16.5mW in active and $10\mu\text{W}$ in sleep mode. Similarly, the SunSPOT makes use of the ST Micro LIS3Lo2AQ triple axis accelerometer with three analog outputs, consuming 3.3mW in the active mode. Additionally, it comprises a Toshiba TPS851 photosensor which generates a current proportional to the light intensity at a power consumption of 2mW .

An increasing number of sensors is already equipped with digital interfaces, which reduce the impact of electromagnetic noise emitted by digital components within proximity to the signal transmission lines. Additionally, the data conversion within the sensor device makes additional ADCs unnecessary. Sensors with digital interface are provided with an integrated logic that manages the exchange of readings with the MCU, and caters for extended features such as sleep modes in which the physical sensor hardware is powered down. A prominent example for a sensor device with digital interface is the Sensirion SHT11, which can sample temperature and humidity. Due to the incorporated digital logic circuitry, devices with digital interfaces have a static power consumption, similar to the driving circuit of analog sensors. For the SHT11, the power consumption ranges from $2\mu\text{W}$ in sleep to 3mW in measurement mode. More sophisticated sensors with digital interfaces are found on extension boards to the Imote2. The ITS400 board comprises a three-axis accelerometer, two digital temperature sensors, a humidity sensor, and two light sensors. Access to all of the components is provided over Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I2C) digital interfaces.

The authors of [RGS06] thus analyze energy-aware sensing techniques to achieve feasible lifetimes for nodes with power-hungry sensors. Further work on energy-efficient sensing is presented for various deployment scenarios, such as snow fields [AAG⁺07], biochemical sensing applications [DCSH08], or the analysis of seismic and acoustic waves [SBC⁺05]. Energy efficiency is especially vital when active transducers like ultrasound, sonar, radar, or laser ranging units are being used, which require a high amount of energy to operate [DE06].

RADIO TRANSCEIVER In order to fulfill the core functionality of wireless data exchange, a dedicated transceiver (composed of *transmitter* and *receiver*) is present on all mote platforms. For global royalty-free operation, these transceivers generally operate in the Industrial, Scientific, and Medical (ISM) frequency bands, as defined by the ITU-R in [Int08]. WSN node often employ lightweight communication mechanisms, such as the IEEE 802.15.4 standard [IEE06] for low-power short-range communication at low data rates. The decision for IEEE 802.15.4 is mainly based on the low energy consumption of according transceiver hardware, such that low-power operation of nodes for extended periods of time is possible. In order to highlight this discrepancy, the power consumptions of transceivers for both IEEE 802.11 (Wireless LAN) and IEEE 802.15.4 are compared in Table 3. Even though the Intel PRO/Wireless 2200 chipset analyzed in the table has not been specifically designed for energy-efficient operation, transmit power consumptions in excess of several hundred milliwatts are inherent to almost any IEEE 802.11 chipset [EBW02, CIB11]. In summary, the numbers shown in the table clearly confirm the significantly greater power demand of IEEE 802.11 over the low-power IEEE 802.15.4 standard.

Table 3: Power consumption of IEEE 802.11 and IEEE 802.15.4 radio transceivers

STANDARD	DEVICE	POWER CONSUMPTION		
		SLEEP	RECEIVE	TRANSMIT
IEEE 802.15.4	Microchip MRF24J40 [Mico7]	$6.6\mu\text{W}$	62.7mW	75.9mW (0dBm)
IEEE 802.11	Intel PRO/Wireless 2200 [Int05a]	60mW	850mW	$1,450\text{mW}$ (15dBm)

Table 4: Power consumption comparison of radio transceivers deployed on current motes

MOTE	RADIO DEVICE	FREQUENCY	POWER CONSUMPTION			
			SLEEP	IDLE	RECEIVE	TRANSMIT
TelosB	TI CC2420	2.4GHz	60 μ W	1.3mW	52.2mW	56.4mW (0dBm)
TinyNode	Semtech XE1205	915MHz	66 μ W	2.8mW	46.2mW	109mW (5dBm)
EyesIFX	Infineon TDA5250	868MHz	25nW	2.6mW	27mW	35.7mW (9dBm)
Mica2	TI CC1000	868MHz	347 μ W	2.8mW	31.7mW	54.5mW (0dBm)

The power consumption of current radio transceivers are compared in Table 4, from which power consumption values between 27mW and 109mW in active mode can be observed. Current radio transceivers thereby exceed the power demand of MCUs (as compared in Table 2) significantly, e.g. by a factor of 37 on the TelosB mote. As the operation of the radio transceiver represents the primary energy consumer on a mote, approaches to maximize node lifetimes often address suitable means to reduce the activity of the transceiver, e.g. by means of duty cycling. In any case, applications need to balance the trade-off between quality and timeliness aspects of the radio transmission and the energy required for operation. Versatile software-based solutions to reduce the power consumption of radio transceivers exist, including energy-efficient medium access control [EHD04], or energy-aware routing [AY05]. Another hardware-based way to reduce the energy consumption of the radio transceiver is the use of secondary low-power radio receivers which are used to signal upcoming radio transmissions to the MCU, such that the primary radio transceiver can be activated on demand [vdDKL09].

ACTUATION DEVICES To interact with the physical world, WSN nodes are fitted with actuation capabilities [AK04]. A broad range of actuators can be attached to current MCUs, as binary outputs (high/low logic level), pulse-width modulated signal outputs, and various on-board bus protocols are supported by their on-board USARTs. The most trivial form of actuation, present on most current mote platforms, is realized through Light Emitting Diodes (LEDs), which emit visible light when an electrical current is applied. Average power consumptions of current LED technologies range from a few milliwatts for low-current LEDs up to several watts for high power LEDs. Instead of manufacturing motes with hard-wired actuation devices, current motes are mostly confined to a set of GPIO pins and according software extension options to interface actuator devices in both hard- and software easily.

EXTERNAL DATA STORAGE Local data storage represents an alternative to the immediate forwarding of sensor readings over the radio. Current MCUs only provide a small amount of volatile RAM which is additionally shared with the implemented application, and even less non-volatile data storage, typically in the range of a few hundred bytes. To maintain logged sensor data even after a node has been reset, the use of external data storage is necessary. Current motes thus employ data storages with capacities ranging from a few megabits up to several gigabits, interfaced via on-board digital interconnections. For example, the eyesIFXv2.1 mote uses an Atmel AT45DB041B Flash memory with 4.125Mbits of storage, and a Numonyx M25P80 with 8Mbits of storage capacity is present on the TelosB.

Storage elements allow a large amount of sensor data to be buffered locally and processed in bulk. With regard to their energy consumption however, it must be accounted that read and write operations need to be performed in a page-wise manner, with common page sizes of 256 bytes (M25P80) or 264 bytes (AT45DB041B), respectively. While the Flash storage devices consume less than 7 μ W in standby, around 13mW are required for read operations, and 50mW to write or erase a page of data. In order to improve the energy efficiency of accessing data on the local data storage, special Flash file systems for motes have been developed, such as [DNH04, TDHV09], which coordinate the access to the storage.

2.2.4 Summary

The dissection of current mote platforms to their hardware components has shown that electrical energy is vital for the operation of all comprised components. However, it is generally only available to a limited extent due to the functional requirements of portability and untethered operation. Batteries are therefore prevalently used, as they combine the benefits of constant output voltages, high energy density, and dimensions suitable for portable devices. Although approaches to extract energy from the surrounding environment have been prototypically demonstrated, these energy harvesting devices are often limited in their output power and in consequence rarely being used in practical deployments today.

The conscious use of the available energy budget is vital for lasting WSN operation. A detailed analysis of the components present on current motes has shown that the two most significant energy consumers are the Flash memory device and the wireless radio transceiver, verifying the insights attained through practical experimentation by Nguyen et al. in [NFG11]. Many approaches have thus been developed in order to address the finite energy budget and optimize the node runtime, e.g. by duty-cycling energy-hungry components or scaling the MCU clock frequency according to the computational load. While runtimes are confined to a few days at most when all components are continuously operating, the application of energy management enables node runtimes up to several months or even more in practice. Based on the fact that local processing is often more energy-efficient than wireless communication, a plethora of energy-efficient algorithms (e.g. [SS01, LRW04, SM06]) have additionally been proposed, although their detailed discussion is beyond the scope of this thesis. In conclusion, energy management represents one of the essential tasks in WSN development to enable sustained operation of the network.

2.3 SMART SPACES

Besides the prevalent use of WSNs in environmental monitoring [HM06], the versatility of embedded sensing and actuation systems also enables them to be used in a variety of further application scenarios, e.g. surveillance and target tracking [HKS⁺06] or industrial process monitoring [CMH10]. Most relevant to the scope of this thesis, however, WSNs are also highly suited as enablers for ubiquitous and pervasive computing. In his visionary article dating back to the year 1991, Mark Weiser predicted that as a result of technological progress “[...] hundreds of computers will come to be invisible to common awareness. People will simply use them unconsciously to accomplish everyday tasks” [Wei91].

Current research prototypes have already reached a state in which embedded sensor systems are less than one cubic millimeter in size, the largest fraction of their volume being used for the energy supply. Inspired by the analogy between the small device sizes and dust particles, miniaturized sensor networks are also frequently termed *smart dust* [DWBP01, WLLP01]. Although current WSN nodes, such as the platforms compared throughout Sec. 2.2, still exceed the size design targets significantly, the notion of smart dust is still being used when addressing the seamless integration of embedded microcomputers into everyday objects, homes, and office spaces. Many novel application scenarios have arisen as a result of the emergence of smart dust, a selection of which are presented in the following list:

- ▷ INCREASING THE COMFORT OF LIVING By utilizing the available sensors to keep track of a user’s current context (cf. Sec. 2.3.2), the physical environment can be directly influenced according to the user’s preferences and wishes. Possible actions in a home environment include turning the lights on when it is both getting dark inside the room and a user is present, or adjusting the heating level according to the anticipated user presence in a room.
- ▷ ENERGY-EFFICIENT BUILDING CONTROL Large fractions of the power dissipated in households and industrial facilities can be attributed to electrically powered devices and Heating, Ventilation, and Air Conditioning (HVAC) equipment [Intoo]. As many

devices can only be switched manually, they are likely to be kept running continuously for convenience reasons. In many situations, sensors can provide information about the energy consumption of active devices. Besides creating awareness for the energy demand, smart buildings can also utilize WSANs to control both electrical devices and HVAC units, and thus preserve energy when no user is present [AFWH05].

- ▷ AMBIENT ASSISTED LIVING Sensor-enriched environments can also be used to monitor elderly people in their familiar environment instead of having to relocate them to retirement homes. Ambient Assisted Living (AAL) environments can be configured to notify relatives and medical personnel immediately when exceptional events are detected by the networked sensors [VWS⁺06, BLTS08].

Although the selected application possibilities only represent a subset of the possibilities enabled by ubiquitous computing, the selection already hints at the wide range of options offered by the combination of distributed sensing and actuation capabilities. Especially when considering the great number of everyday objects which can be manufactured or retrofitted with sensing and actuation technology (also termed *smart objects* or *things* [Int05b]), it becomes clear that an almost infinite number of logical connections between these devices can be established.

2.3.1 Smart Objects and the Internet of Things

Novel services emerge through the possible interaction between individual smart objects and users, e.g. through the visualization of locally sensed information or by acting as control units. However, by enriching smart things with networking capabilities and making their local resources and capabilities available to adjoining devices, the resulting synergy effect leads to an even greater amount of possible smart services. Everyday objects like books, electrical appliances, or even grocery items may be turned into smart objects, eventually becoming a commodity in all areas of daily life.

The presence of innumerable networked smart objects, invisibly connected to each other and heterogeneous in type, clearly necessitates sophisticated solutions to manage the communication between them. Based on the evolution of the Internet, which has succeeded in creating a world-embracing network, visions for global large-scale sensor networks have culminated in the definition of the Internet of Things (IoT) [Int05b]. By borrowing from the well-established concepts that globally connect millions of computers, the major goal of the IoT initiative is to interface smart objects to the Internet. Internetworking and application of the Internet Protocol in version 6 (IPv6) are expected to make the interaction between users and things seamless and intuitive, much like the experience of browsing the web today.

From a hardware perspective, the functional capabilities of WSAN nodes presented in Sec. 2.1 neatly fulfill the requirements to smart objects; they combine wireless communication with local data processing and integrate interaction capabilities with the real world. For less complex tasks like object identification or localization, Radio-Frequency Identification (RFID) technology embedded in smart products also represents an affordable way to integrate corresponding functionality. A mote's reprogrammable MCU enables the integration of the protocols required to participate in the IoT, effectively enabling the exchange of sensor readings and actuation commands with remote devices over the network.

Ongoing research on smart objects has yielded many different approaches for a practical integration of such devices into a global IoT, most prominently defined in the whitepapers provided by the IP for Smart Objects (IPSO) alliance [IPS11]. Founded in 2008, the alliance focuses on defining guidelines for the integration of motes by means of lightweight operating systems, the use of IPv6, low-power communication link layers, and security. However, neither the IPSO alliance nor other groups have yet managed to develop widely accepted standards that realize Mark Weiser's vision of invisible computers and the IoT. Many scientific aspects still remain to be investigated in depth before embedded computing systems may become a ubiquitous commodity.

2.3.2 Context Awareness and Ambient Intelligence

Through their integration into the IoT, motes are enabled to share resources with each other to achieve goals in a collaborative manner. This combination of distributed processing, sensing, and actuation represents one of the major enabling technologies for Ambient Intelligence (AmI) [DBS⁺01]. By configuring unobtrusively deployed motes to sense the environments of users and objects and to interact with these entities, novel application scenarios become possible, frequently referred to as context-aware computing or embedded intelligence. The notion of context defined as by Dey and Abowd represents a widely accepted foundation:

“Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.” [DA99]

Based on this definition, manyfold applications for context-aware computing have emerged in the recent past [MR03], including the use of location information to assist the user in selecting nearby objects [SAW94], automated profile selection on mobile phones [SAT⁺99], or application in telephone switchboards [Gör05]. Although not explicitly addressed in the definition, the available amount of sensor data has an important impact in order to correctly characterize situations. Current research prototypes for embedded sensing systems, as introduced in Sec. 2.2, often come populated with sensors for temperature, humidity, and light intensity. Even large numbers of these motes cannot provide accurate context information in all categories, e.g. about the user’s current location, identity, or activity. Besides the quantity of available sensors, their relevance (“Does the sensor contribute to the context of the given entity?”) and modality (“Can the sensor type provide data for a given context dimension?”) therefore also play a vital role to capture context in a holistic way. In addition to solely relying on embedded wireless sensing systems, many context-aware systems therefore also utilize heterogeneous context sources in the form of software-based data suppliers on computers and smartphones, or even building automation systems.

After the collection of context information from multi-modal sources, the transition from raw sensor data to a high-level information about a user’s state in a given context category must be made. Commonly, this processing step is realized by either composing rule sets which model the relation between sensors and context state [Nel98], or by applying supervised machine learning techniques (e.g. in [DKSo6]) to extract the relevant features and cater for the classification of sensor data. A comprehensive discussion of the sensor data evaluation component is however beyond the scope of this thesis, but discussed in further detail in related work, e.g. [Schog].

2.3.3 Smart Homes and Buildings

A concrete application scenario of AmI, which combines the need for ubiquitous and unobtrusive sensing and actuation devices, seamlessly networked in a web of things and collecting context information about users and objects, is the realization of smart home and office environments. In such smart spaces, a wide range of sensing and actuation modalities is invisibly integrated into a user’s environment, such as depicted in Fig. 5. By capturing the user’s context and his actions through the sensing infrastructure, and combining all context information into a generic model for the user’s wishes and preferences, the smart office can improve comfort, economic efficiency, and security. In contrast to conventional building automation systems, which mostly rely on timer clock-based approaches, smart spaces have clear benefits in many regards:

- ▷ They autonomously adapt to the user at low configuration overhead. While timer-based solutions for HVAC either need to be configured conservatively or require reconfiguration whenever a user’s presence times within a building change, smart environments

automatically determine the optimal heating schedule by taking user context (especially location and anticipated presence) into account.

- ▷ In times of rising energy prices, the conscious use of energy resources is highly desirable. Smart buildings provide the capability of metering electricity, heating, and water consumption using the deployed sensing devices [Clio8]. As they are part of the IoT, the sensor readings can be displayed to the user to create awareness for his consumption. Additionally, by analyzing the consumption values over longer periods of time and correlating them with context information, smart spaces may propose rules to the user in order to help in altering his daily routines and thereby reduce his energy footprint.
- ▷ By connecting presence information with the state of door and window locks, novel security features can be realized. As an example, discrepancies between the actual and the expected user activity in a smart environment can be directly used to trigger the transmission of notifications to the building owner and/or external security personnel.
- ▷ Besides acting upon the local measurement of environmental parameters (such as wind speed and rainfall), information from both forecasts and historical archives can also be regarded by the smart building. As a result, window blinds can be set to park positions before a storm hits, or reminders to close the windows be broadcast before heavy rainfalls. Similarly, when high amounts of sunlight are forecast, blinds and windows may be adjusted autonomously to maintain the desired temperature, air quality, and amount of heating required.
- ▷ Last, but not least, by laying the foundations for AAL, versatile assistance services can also be implemented on top of a smart building architecture. Combining the detection of abnormal events and global connectivity, notification of relatives and medical staff are possible.

Current implementations of smart buildings are often confined to specific functionalities, such as activating airport HVAC and lighting only for gates which are in use, or replacing conventional thermostats by digital equivalents to remotely control room temperatures in office buildings [Mito5]. On the way towards full-fledged smart environments, many practical challenges still remain unsolved. Besides catering for miniaturized sensor and actuator platforms, a universal solution for supplying them with electrical energy is required. Similarly, platform-independent networking protocols and mechanisms for seamless integration of new devices need to be devised and standardized to establish a common basis for communication. Many of the transferred data bear sensitive information, thus both safety and security aspects need to be dedicatedly regarded. Finally, scalable and extensible middleware solutions which are capable of incorporating the anticipated amount of sensor data are required.

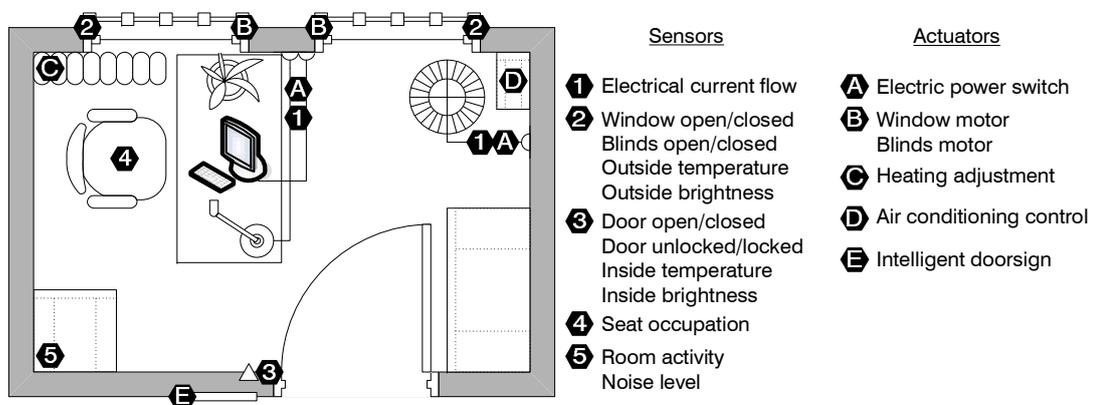


Figure 5: Example configuration for the deployment of physical sensors and actuators in a smart office

2.4 SUMMARY

Wireless sensor and actuator nodes are small hardware platforms which comprise sensing, actuation, processing, and wireless communication capabilities. Their versatility makes them highly suited for application in a broad range of domains, including environmental monitoring, industrial automation, or smart environments. As the nodes are generally operated on a finite energy supply, prevalently realized in the form of batteries, long-term WSN deployments require regular maintenance operation to recharge or replace the energy sources. Although solutions for harvesting energy from the environment exist and have been outlined in this section, their use on motes is still limited today. As a result, the energy-efficient design and optimization of soft- and hardware remains a common approach to extend a node's runtime and thus reduce the maintenance overhead.

Subsequently, the application domains of WSNs have been briefly summarized. A special emphasis was put on the use of sensor and actuator nodes in smart spaces, which represents the primary application domain for the contributions of this thesis. Following Mark Weiser's vision of ubiquitous computing, computers will become invisible and impossible to distinguish from the environment. At the size of a grain of dust, these highly embedded systems provide the basis for smart objects and the web of things. They integrate seamlessly with everyday objects and enable the creation of smart environments by combining sensing and actuation with context awareness. Wireless networking between nodes and the cooperation of sensors, actuators, and context evaluation systems have thus been highlighted as fundamental prerequisites to smart environments. For a successful deployment of smart spaces, universally applicable solutions to these key challenges need to be found.

We work to become, not to acquire.

Elbert Hubbard

In this chapter, we summarize existing work related to the challenges presented in Chapter 1. Although both energy-efficient operation and the creation of interoperability between heterogeneous nodes are major prerequisites for the successful realization of smart environments, the combination of both is rarely addressed in existing research. As a result of the unavailability of means to create interoperability in an energy-efficient way, work related to either of the fields is identified and summarized in this section. We first discuss existing means to bridge the heterogeneity between multimodal sensor and actuator nodes. Based on our observation that the radio transceiver represents the major power consumer on current mote platforms, we subsequently analyze solutions that address the reduction of its energy demand by applying data compression. Our summary of related research approaches is followed by an assessment of their applicability in the envisioned application area. A detailed problem statement follows the survey of existing solutions and our analysis of their applicability in smart spaces.

3.1 GENERIC INTERFACING AND UNIFIED ACCESS TO SENSORS AND ACTUATORS

In industrial and environmental monitoring scenarios, WSNs are commonly deployed with a clearly defined and invariant goal, e.g. the periodic monitoring of certain parameters or notifications on the occurrence of exceptional events. This knowledge of the deployment goal significantly impacts the design of such networks, and many aspects (in both hard- and software) are optimized according to the given constraints. As a result, WSN communication protocols traditionally define packet structures statically at compile time due to efficiency reasons, i.e. each of the fields in a radio message is hard-coded to contain a particular type of data [HSW⁺00]. The use of small packets with hard-coded content structures is highly efficient with regard to the computational complexity (and thus the demand for energy) of accessing the data in the packet. As its downside however, applications must be adapted and re-deployed on the nodes whenever new data types are introduced into the network. In other words, extending such WSNs by additional motes with different capabilities is virtually impossible due to the fixed packet structures.

A first step towards running multiple parallel applications within the same WSN has been the transition from a purely *address-centric* towards a *data-centric* system design. Instead of addressing specific nodes individually, the concepts behind data-centric approaches like directed diffusion [IGE00] or TinyDB [GHH⁺02] are mostly based on expressing interests in certain data types. By flooding the network with messages indicating these interests, gradient-based routes are established between sources and consumers of data. Likewise, the concept of semantic streams regards all sensors as event sources while dedicated processing modules in the network are used as inference engines [WZLo6]. Although enabling multiple connections between nodes in the network, both directed diffusion and semantic streams are purely data-driven approaches, and do not support address-oriented communication in the network. Due to the limited scalability of flooding-based approaches and the necessity of sophisticated data type descriptions, data-centric communication has however not been adopted in many WSANs.

Instead, the popular TinyOS [HSW⁺00] and Contiki [DGV04] operating systems for embedded sensing systems incorporate address-centric concepts, such as active messaging in TinyOS [LMG⁺04]. Active messages contain a type field, the so called *active message type*, which indicates the addressed functionality on the receiver side. Similar to the notion of ports

in TCP [Pos81] and UDP [Pos80], active message types are being used to differentiate between packet types and their corresponding internal structures. Although the use of active message types enables the address-centric communication between multiple disparate applications, the concurrent use of differently structured packets still requires the re-deployment of application software on all nodes when new functionality is added to the WSN, i.e. when new packet types are introduced in the network. In recent years, a trend towards shifting to the address-centric IPv6 can be observed for sensor and actuator networks [MKHC07, YD09, VD10, HC10, IPS11]. Besides simplifying the integration of sensor nodes into the globally prevalent IP-based networks, low-power implementations like 6LoWPAN (IPv6 over Low-power Wireless Personal Area Networks) provide network-layer connectivity for low-power embedded systems. 6LoWPAN and comparable approaches are thus seen as the enablers for cross-platform communication.

The vision of smart environments relies on a continuous deployment of sensor and actuator nodes, which are envisaged to be embedded in everyday objects. As a result, nodes might be disconnected from the network due to the depletion of their energy budget or their relocation out of the network's connectivity region. Meanwhile, new devices may be introduced during system runtime in order to extend the network and enrich its sensing and actuation functionality. The tight integration with the environment complicates the reprogramming and replacement of nodes. Integrating new devices into the network during runtime thus needs to be possible without the need to take each sensor offline and perform a manual reconfiguration step. However, despite creating the foundations for communication between WSN nodes, the presented network-layer protocols do not solve the challenge of unifying the access to sensing and actuation resources on heterogeneous node platforms.

3.1.1 Related Work

With the rise of computer networks and the Internet, the way has been paved for the usage of functionality through the invocation of methods and services on remote machines. The majority of approaches thereby follows the paradigm of Remote Procedure Calls (RPCs) as initially presented by Birrell and Nelson in [BN84]. Traditionally, the concept of RPCs is based on the provision of local functionality over the network, such that other nodes in the network can invoke the offered procedures remotely. The Common Object Request Broker Architecture (CORBA) [Vin97] was among the first widely adopted solutions to access functionality on remote hosts. Because CORBA is designed to be independent of the underlying platform and the used programming language, it represents a viable approach to cater for interoperability between heterogeneous systems.

Java Remote Method Invocation (RMI) has been specified in [Sun06], and represents a convenient solution to access objects and functions on remote Java virtual machines. By allowing to exchange any serializable Java object, and thus the majority of data structures, it provides a versatile solution to access objects and invoke methods on remote hosts. Java RMI however has the downside of being confined to the Java programming language. Similar restrictions exist for Microsoft's .NET Remoting [MWN02], which presents means of invoking remote methods focussed on the Microsoft .NET framework. Their tight binding to specific programming languages effectively hampers their use in networks comprised of heterogeneous node platforms. In an approach to address this issue, Remote-OSGi [RAR07] is composed of a dedicated platform-independent framework which handles the exchange of required object structures and has been designed to allow efficient communications between Open Services Gateway initiative (OSGi) frameworks. The operability of OSGi frameworks on nodes is however rarely possible, because current implementations are almost exclusively available in Java as of today, with a port based on the C programming language still in its infancy [Apa10].

In an approach to reduce the required efforts for remote service usage, the SOAP protocol [ML07] realizes the concept of invoking functionality on remote nodes. It makes use of eXtended Markup Language (XML) messages to achieve platform-independent access to remote services, and enables the usage of various different transport protocols. Analog to

CORBA, services are a priori described using specific languages [CCMW01]. The efficient encoding of messages is however not within primary focus of SOAP, and consequently, human-readable XML message representations have been preferred over efficient binary encodings. Since the introduction of the still computationally demanding SOAP, a couple of further cross-platform RPC approaches have been developed. These include Apache Etch [Apa09], Protocol Buffers [Goo09], or the Apache Thrift protocol [Apo08], which all present similar means to invoke remote services. Although targeting cross-platform interoperability, their design has been optimized for performance on desktop computer systems.

RESTful Web Services [RR07a] present a lightweight alternative to SOAP. Their prevalent operation over HTTP however disallows for binary query representations, which has a negative impact on the efficient encoding of messages. Like the SOAP protocol, XML-RPC [Use03] has been designed with flexibility in mind. While optimized for simplicity of use, the protocol also uses verbose XML message representations to execute remote services in a platform-independent way. Additionally, XML-RPC ships with a set of pre-defined data types only, and necessitates the use of additional components to define application-specific data types, such as the XStream serializer [WTVZ⁺05] for transferring complex objects.

Addressing the overhead introduced by the verbose character of XML, recent solutions provide means to alleviate the problem by using lightweight representations, such as the compact JavaScript Object Notation (JSON) in the SOAPjr implementation [Mano8]. kXML [Hau05] is an XML parser for resource-constrained devices, such as mobile phones, and provides support for the WAP Binary XML (WBXML) compression [Wiro01] to efficiently encode XML messages. Although the structure of the XML file is efficiently represented after the WBXML compression step, the contents of all data fields are treated as string representations and transferred uncompressed, thus the encoding is suboptimal in terms of size.

Specifically adapted to embedded sensing systems, the TinyRPC stack [MDDHo6] follows the concepts behind CORBA and allows for static remote procedure calls on sensor network nodes. It utilizes a compiler extension, which extracts the signatures of all methods offered for remote invocation at compile time. Both local and remote function calls are processed by the TinyRPC stack, and remote functions are transparently invoked over the network. Whitehouse et al. have developed Marionette, a version of Embedded RPC [WTT⁺06], which adds RPC functionalities to nodes in WSANs to increase the debugging capabilities of the network. Similar to TinyRPC, interfaces of all debugging methods are extracted during compile time, and made available for remote invocations. However, Marionette only provides means for offering invocable methods, but does not allow nodes to invoke remote methods themselves. SpartanRPC [CS10] presents a link-layer RPC mechanism which enables the invocation of functions on remote devices. Analog to the TinyRPC and Marionette, the set of provided interfaces is statically defined at compile time, and changes or additions to the set of interfaces are thus not supported. SpartanRPC mainly contributes security aspects to the implementation of RPCs on embedded systems.

The notion of Coroutines is presented by Cohen et al. in [CPRR07]. It introduces functionality that overcomes the traditional synchronous one-to-one communication flow of RPCs. Instead, an extension to the task scheduler of WSAN nodes is presented which allows for asynchronous RPC invocations and sustains the node's primary operation while waiting for the reception of a response message. Similarly, the Anycast-RPC concept presented by Bergstrom and Pandey in [BP07] introduces the idea of invocations of remote services based on asynchronous message passing. The impact of packet losses on the communication channels is effectively diminished by the asynchronous character of the solution. Additionally, the notion of one-way message passing is introduced, which can be used to invoke services without return types in a best-effort manner. TinySOA is a stack for service-oriented sensor-actuator networks [RE07], which comprises an event specification language based on the Event-Condition-Action (ECA) model and a dedicated event request routing module. The routing module is operating on each node in the WSAN, where it caters for the matching of event requests and service providers, and the routing of queries to suitable providers. Neither of the approaches however

dedicatedly focuses on the energy efficiency or the efficient encoding of messages, and all of them utilize proprietary message formats incompatible with each other.

Besides the aforementioned solutions that mainly address the feasibility of the RPCs in sensor and actuator networks and corresponding issues with regard to its practical implementation, the efficient serialization of parameters contained in RPC messages is covered in [PWH⁺07]. The presented microFibre serialization approach borrows concepts from the Abstract Syntax Notation One (ASN.1) Packed Encoding Rules (PER) [DF01], e.g. limiting the number of bits required to transfer an integer if its values range between previously known bounds. The serialization preprocesses radio message definitions at compile-time and achieves message size reductions of almost 30%. Addressing a similar field of research, Moritz et al. have presented means for encoding and compression of SOAP messages [MTSG10]. Analog to microFibre, the interface definition step must precede compilation to adapt to the message contents in their implementation. Tiny Web Services [PKGZ08] also rely on compressed SOAP messages which are transferred over TCP/IP connections.

In parallel to the application of the RPC paradigm, data-centric approaches to virtually share memory regions between nodes like Hood [WSBC04] and TeenyLIME [CMMP06] exist for WSNs. Although sensor readings can be written to the distributed memory region, and actuation commands be received from dedicated memory locations, the abstraction of virtual memory may not be as intuitive as the dedicated transmission of requests and actuation commands using RPCs. The use of mobile agents in WSNs is another orthogonal approach to distribute sensing, actuation, or processing tasks to other nodes in the network [CKYL06]. Instead of relying on the dedicated invocation of functionality on remote devices, executable machine code and the corresponding memory segments are encapsulated in so called agents. These agents then traverse the network in order to autonomously collect sensor readings, process them, and perform actuation [ODL09, BL11].

3.1.2 *Applicability of Existing Solutions in Smart Spaces*

The resource constraints and platform heterogeneity in the application scenario of smart environments clearly necessitate the use of adapted solutions that are both generic enough to integrate WSN nodes into large-scale sensor networks, but at the same time cater to their tight resource limitations. RPCs are prominent in the area of Web Services [ACKM03], where they enable loose coupling of components. However, mostly designed for computationally powerful desktop computers, common RPC stacks typically have corresponding computational requirements which mostly exceed the capabilities of motes. Besides the programming efforts required to offer local functionality over the network in the original CORBA system, sophisticated means to describe method interfaces [Obj02] and message encoding based on ASN.1 render CORBA heavyweight and thus only applicable on WSNs nodes to a limited extent [PWH⁺07]. Similarly, the tight binding of Java RMI and .NET Remoting to specific programming languages effectively hampers their use on the majority of available WSN node platforms, where predominantly low-level programming in C is prevalent.

Besides the computational overhead and the required development efforts for the integration of the aforementioned RPC stacks, the performance aspect (and thus inherently the energy efficiency) also need to be regarded. The initial design target of RPCs was transparency, i.e. to make remote calls indistinguishable from local calls. In the case of WSNs however, the computational overhead introduced by calling local functions through the RPCs framework has an immediate impact on both invocation delay and energy consumption, and should hence be avoided. Direct invocation of locally available functions, and thus the circumvention of the RPC stack for local calls, is therefore desirable.

Existing protocols like SOAP often rely on XML-based messages with semantic annotations. The applicability of such implementations on embedded systems is strongly limited, because WSN nodes can often not even transmit packets of more than 127 bytes at a time (assuming operation over IEEE 802.15.4 [IEE06], a standard widely used in WSNs). In order to allow access to sensing and actuation capabilities over an RPC interface, a lightweight message

format as well as an efficient serialization mechanism for messages is necessary. While a static allocation of payload fields in WSNs is the common way of interfacing with the functions offered by nodes, it strongly limits the extensibility of the network by new devices. To gain the flexibility of accessing sensing and actuation functionality offered by remote nodes and at the same time cater for the dynamic extensibility of the network, a generic approach without static field allocations is required.

In summary, existing solutions for desktop computers are generally designed to achieve optimum performance while the efficient use of the available resources is a subordinate requirement. Consequently, their applicability on resource-constrained embedded sensing systems is commonly not given. Instead, RPC frameworks need to specifically take the capabilities of WSN devices into account. In contrast to the striving for performance, the primary design criteria are low resource consumption and operability on motes. Efficient message formats are required in order to cater for the extensibility of the network by additional devices with new sensors or different actuator controls at very small overhead. At the same time, the generic nature of RPCs and the loose coupling of method providers and consumers needs to be maintained, i.e. the integration of new nodes and data types must be possible at any time. While the RPC solutions tailored for their use in WSNs provide valuable contributions towards a holistic architecture for smart spaces, a number of aspects have not been researched in complete detail. First and foremost, resource consumptions almost exclusively focus on the determination of MCU and memory utilization; sophisticated energy analyses still remain an open issue. Furthermore, close adherence to the original RPC concept (e.g. transparency and the synchronous character of invocations) has been shown to mismatch the requirements of WSNs [CPRR07]. Accordingly, adaptations of the concept to the specific application domain thus need to be made. Finally, neither the application on heterogeneous sensor and actuator hardware nor the efficient encoding of messages have been addressed in existing work on RPCs.

3.2 LOSSLESS DATA COMPRESSION IN WIRELESS SENSOR AND ACTUATOR NETWORKS

With the radio transceiver representing the predominant energy consumer on most node platforms, the majority of research contributions target the reduction of its activity periods. Through the reduction of size and amount of transmissions, node lifetimes can be extended significantly. Most prominently, the adaptation of MAC protocols has been addressed in numerous research contributions [LM10], based on the observation by Woo and Culler, who have shown that MAC protocols designed for traditional wireless networks are unsuited for application in WSNs in [WCo1]. Three major contributors to the radio energy consumption on motes are identified in [YHE01]. The occurrence of collisions and the corresponding need to retransmit messages constitutes one source of inefficient energy usage. Listening to the radio channel in order to detect preamble sequences (idle listening) constitutes a second contributor to the energy balance. Finally, overhearing packets addressed to different nodes also consumes energy without any benefit for the receiving node.

Directly addressing the aforementioned contributions to the energy consumption of wireless communications, a common property of all research results in the field of MAC protocols is the duty-cycled operation of the radio transceiver. Initially, the use of low-power listening [PHCo4] was proposed, which periodically activates the radio transceiver in order to check the radio channel for ongoing transmissions. The transmitting node however needs to be configured accordingly to prepend its transmissions by a long preamble, with a duration that is at least equal to the receiver's probing interval. Once the receiving node detects the preamble on the channel, the reception of the full packet is triggered. Extensions to the basic concept of low-power listening include the integration of more adaptation functionality into the MAC protocol [JBL07]. The underlying idea of shifting the energy consumption for transmitting a packet is also adopted in RI-MAC [SGJo8], in which the receiver indicates its availability while the burden of polling the channel periodically is shifted to the sender. The S-MAC protocol [YHE04] introduces the use of a length indicator field, which indicates the time

needed for the successive data transmission. Energy savings are achieved by putting the radio devices of uninvolved nodes into sleep modes during ongoing transmissions while the channel is occupied. An improvement to the S-MAC protocol is presented in [vDL03] and named T-MAC. It uses a synchronization process to ensure that all nodes activate their radio transceivers simultaneously. If no data is being received within an activity phase, the node puts the radio device into sleep mode again. Amongst many others, further contributions include WiseMAC, which takes knowledge about traffic characteristics into account [EHD04] and X-MAC [BYAH06], which uses a series of short packets with destination information before the actual transmission takes place in order to reduce unnecessary overhearing.

Besides controlling the activity periods of the radio device, energy can also be saved by reducing the number of packets which need to be transmitted, i.e. effectively trading reduced throughput for energy savings [SHB10]. Chou et al. present a solution that enriches distributed compression with adaptive prediction functionality in [CPR03]. The distributed compression component takes previous local readings and readings from other nodes into account for the decision how many bits need to be transferred to attain a complete and undistorted representation of the signals at the external sink. This way, the approach only transfers a compressed representation of the sensed value and thereby achieves energy savings ranging from 10% to 65%. The concept is similar to the notion of compressive sensing presented by Donoho in [Dono6]. By exploiting the typical correlation in the readings of neighboring sensor nodes, it allows for a close reconstruction of the original sensor data by taking only a reduced number of sensor readings into account. The application of compressive sensing to reduce both the amount and size of packets in WSNs has e.g. been shown in [RHSNo6, RHC10]. Furthermore, the overall number of packet transmissions in the network can be reduced by means of in-network processing [PK00, HSI⁺01]. In-network processing is especially well suited in application scenarios with clearly defined and invariant goals, e.g. the determination of minimum and maximum temperatures on a monitored field. When a message containing a minimum temperature value passes a node that has already reported a lower temperature boundary, the message is automatically discarded and does not traverse the routing topology any further.

A third class of approaches to preserve energy exclusively targets the reduction of packet sizes by means of data compression, also termed source coding. The local processing of sensor readings in order to reduce redundancies in the signal and increase its entropy caters for smaller packet payloads, and thus results in shorter transmission durations. Besides reducing the required transceiver activity and thus the corresponding energy demand, the risk of collisions (e.g. due to the hidden terminal problem [TK75]) is also reduced when smaller packets are used. Especially for messages that cannot be easily omitted from transmission (e.g. network management or beaconing messages), data compression is a viable step to reduce the required energy expenditure for transmission. Compression algorithms can be categorized into lossy and lossless solutions [Sal07]. While the former group of algorithms is well applicable for sensor readings for which small deviations are tolerable, packet headers and addressing fields generally need to be always transferred in a lossless manner. Similarly, network management messages (e.g. routing information or topology control messages) need to be transported in an accurate fashion to avoid ambiguities at the receiver side. Due to the fact that lossy data compression is highly specific to the underlying data type (intuitively, webcam images require a different type of data compression than temperature samples taken at hourly intervals), no universally applicable solutions exist. Optimized lossy compression solutions specifically tailored to the underlying sensor type and model are hence predominantly applied, because they result in better compression gains than the use of generic solutions [KL05]. On the contrary, a wide range of lossless data compression algorithms is available and used in manifold application scenarios today. Most compression algorithms have however been developed with high compression gains in mind, while their applicability on devices with tight resource limitations is typically not within primary focus.

3.2.1 *Related Work*

Research on reducing the size of executables and data files to save valuable space on costly storage devices has been conducted for many years. After Ziv and Lempel presented the LZ77 algorithm in 1977 [ZL77], many further approaches have been made towards compressing data in a lossless manner. Most lossless data compression algorithms are composed of three steps. In the initial preprocessing step, the input sequence is reversibly permuted in order to increase its compressibility. The most common preprocessing algorithms include Move-to-Front Coding (MTF) coding [BSTW86], and the Burrows-Wheeler Transform (BWT) [BW94], although a variety of modifications and alternatives are known (cf. [De002]). Subsequently, data compression takes place, reducing the size of the preprocessed data by eliminating redundancies. Run-Length Encoding (RLE) is frequently used to reduce multiple successive appearances of the same symbol within an input string by replacing its repetitions with an appearance count field. The more complex LZ77 [ZL77] and LZW [Wel84] algorithms compare sequences in the input data to elements contained in a sliding window or a dictionary, respectively, and replace matching elements by pointers to their previous occurrences. Finally, the main compression operation is followed by an entropy coding step that increases the entropy of the output data. Range Coding (RC) increases the information entropy by iteratively dividing a finite length number range proportionally to the occurrence count of all symbols within the input, and generates a single floating point number representing the entire input sequence [Mar79]. Alternatively, Huffman trees are widely used for the entropy coding step. They contain bit representations of used symbols, with their length approximately inversely proportional to the frequency of their occurrence within the input stream [Huf52]. Entropy decoders either need the used symbol dictionary in advance, or operate on a static dictionary with decreased coding gain. This limitation can however be overcome by using adaptive coding approaches, e.g. dynamic Huffman codes [Vit87].

The area of data compression in WSANs has been addressed in different ways in recent research. Pottie and Kaiser have determined in [PK00] that the energy demand to transfer one kilobyte of data over a distance of one hundred meters in a WSN is the same as required for executing three million MCU instructions. Barr and Asanović have determined in [BA03] that the energy consumed by the radio device to transmit one byte of data on the radio can also be used to perform up to a thousand MCU operations. Later, this observation was confirmed by Sadler and Martonosi, who determined that the one-hop transmission of a single byte consumes energy equivalent to performing several thousand instructions on a Texas Instruments MSP430 microcontroller [SM06]. These observations motivate the feasibility of data compression in WSANs, and in consequence, a plethora of compression means for both packet headers and payloads were proposed in sensor networks research.

Packet Payload Compression

The envisioned ubiquity of dense sensor network deployments has motivated Pradhan et al. to present the DISCUS framework in [PKR02]. By exploiting the presence of side information (i.e. the correlation between sensor readings), the amount of communication between nodes is minimized while the accuracy of the collected data is maintained. DISCUS leverages lossless Slepian-Wolf coding for an efficient transmission of the required fields. The applicability of lossless Slepian-Wolf and lossy Wyner-Ziv compression on video transmissions in sensor networks is discussed in [XLC04]. The authors highlight the need for cooperation between groups of nodes, because at least one node needs to transfer the full video stream in order to enable all others to only transfer their side information. Pattem et al. also describe a scheme for routing with compression of spatially correlated data in [PKG04]. They base their contribution on the observation that the combination of clustering and in-network processing leads to traffic reductions in densely deployed sensor networks. The determination of signal correlations is exclusively performed within the clusters, and only the aggregated and compressed data are being sent to the sink. The PINCO scheme for in-network compression also buffers compressed data at intermediate nodes on the route to the sink [AGAL03]. Through the collection of data

originating from several sensors, both temporal and spatial redundancies can be detected and efficiently compressed. A maximum latency bound triggers the transmission of compressed data to the sink in order to fulfill the desired delay requirements. The number of messages in the network can also be reduced by omitting events from transmission [KW08b, ZRMS10] if their relevance is below a given threshold. All of these approaches however operate on network level and do not take the local optimization of the energy budget into account.

Besides network-oriented approaches, many solutions that are executed locally on nodes have been presented in recent research. Schoellhammer et al. have determined the need for data compression to reduce the required storage and transmission efforts in WSNs [SGO⁺04]. In their approach, linear trends of the collected sensor readings are extracted from past readings. Current sensor readings are then fitted to the determined trends by means of slight alterations in order to maximize the compression gain. A comparison between a simple threshold-based compression technique and differential encoding is presented in [PGN⁺03]. The results indicate that for a set of collected Global Positioning System (GPS) traces, size reductions between 80% and 86% are achievable, resulting in overall energy savings from 22% to 43% on a mobile sensor network of Personal Digital Assistant (PDA) devices. The transmission of packet predictions, as presented by Blass et al. in [BTZ06], uses Kalman filters to predict readings. Instead of the actual sensor readings, only the relevant coefficients are transferred to the sink in order to enable the reconstruction of the original values. Its applicability in sensor networks is however strongly hampered by its computational complexity. Other compression approaches that analyze the characteristics of the data before the actual compression step are presented in [SGO⁺04, CS09, XD10]. Based on linear prediction models, the algorithms achieve good compression gains when sensor samples with slow variations are given. However, the approaches fail to compress fast varying sensor data.

A common attribute of the aforementioned approaches is their local evaluation of the signal characteristics prior to the compression step. The compression is however not always lossless due to the use of estimation or windowing techniques. Lossless predictive coding is presented in [HL07], which utilizes a linear prediction model in order to reduce the amount of transferred data. Similarly, [LP10] adopts a Laplacian distribution in order to represent the characteristics of sensor readings. When packet structures can be statically defined prior to node deployment and some fields are known to remain constant or only change incrementally, the EasiPC packet compression scheme by Ju and Cui [JC05] can also be used to transmit changed fields only. Its application is however limited to scenarios in which the bounds of the transferred values and their presence within packets can be determined before the actual deployment. The compression of messages in XML representation is presented in [HRN⁺10]. The incorporation of prior knowledge about the contained data structures thereby effectively contributes to the compression gains. In [TDV08], Tsiftes et al. have focussed on compressing firmware updates that are transferred over the radio, and designed the SBZIP algorithm, a derivative of BZIP2, adapted to the requirements present in sensor networks. However, the implementation of SBZIP on sensor nodes does not target to compress application-generated data. Instead, it only caters for the decompression of code updates, while the compression takes place external to the WSN.

In the aforementioned work by Sadler and Martonosi [SM06], the authors propose the RT-LZW (retransmission LZW) algorithm, a special adaptation of the LZW algorithm for resource-constrained nodes communicating over lossy channels. RT-LZW operates on aggregated data blocks of 528 bytes each, and achieves compression gains up to a factor of 2.5. It relies on retransmissions of lost packets in order to ensure that data required to construct the code dictionary is present at both parties, possibly resulting in energy expenses for these additional transmissions. The importance of lossless data collection in body-area networks is demonstrated in [MV08]. To still cater for an efficient transfer of physiological parameters, the presented solution encodes the signal differences and dynamically assigns Huffman codes to the differences. Its combination with dictionary approaches that map signal coefficients to compact binary representations is discussed in [MV09]. Guitton et al. have analyzed the applicability of adaptive data compression in WSNs in [GTH08], and proposed to extend

the Adaptive Huffman Coding (AHC) algorithm by fault-tolerant mechanisms to cater for typical channel characteristics in sensor networks. Their contributions include groupwise acknowledgements of encoded data and adaptation of the dictionaries to successfully received data only. An analysis of the achievable compression gains or the energy consumption of their algorithm is however not presented. Marcelloni and Vecchio present another approach towards lossless entropy coding (LEC) compression in [MV09], in which differential coding is combined with exponential Golomb codes in order to efficiently encode the difference between two successive sensor readings. Their LEC approach does not operate on packet level, but calculates the differential values for the readings of each sensor individually. In [vdBNW09], a small number of lossless and lossy compression algorithms are being compared with regard to their energy demand and the battery runtimes on commonly available sensor nodes. Finally, dedicated external hardware components for data compression are discussed in [CP09]. By employing Application-specific Integrated Circuits (ASICs) or reprogrammable hardware, significant energy savings can be achieved in comparison to software implementations of the compression algorithms. The integration of external components on mote platforms is however not feasible in many application scenarios, e.g. due to cost pressure or size constraints. In contrast to the presented approaches, which exploit temporal similarities of sensor data on a local scale, compressive sensing [CW08] reduces the amount of transferred data on network level by exploiting the spatial characteristics of the monitored physical phenomenon.

Packet Header Compression

Besides compressing the payload of radio packets, successive packet headers often bear resemblance and are thus target of further optimization approaches. The compression of packet headers in IP-based networks has been initially presented in 1984 by Farber et al. in [FDC84]. Specifically addressing the transmission of TCP/IP traffic over low speed serial connections, the Thinwire I protocol reduces Telnet packets (i.e. packets with a very small payload) from 41 bytes to 17 bytes each; its successor Thinwire II even reduces the compressed packets to 13 bytes at the cost of increased computational demands at the receiver side. Since then, many algorithms have been published to reduce the header size of IP-based traffic. In general, header compression schemes can be classified into *hop-by-hop* and *end-to-end* mechanisms. While hop-by-hop header compression mechanisms often operate on the lower layers of the network stack (cf. Fig. 6), mechanisms applying the end-to-end concept generally try to exploit similarities in the packet headers of upper layers.

A stateful hop-by-hop compression scheme, reducing the 40 byte TCP/IP header to sizes as small as 3 bytes, is presented in RFC 1144 [Jac90]. Operating in hop-by-hop manner, the mechanism makes several assumptions about the characteristics of TCP/IP headers, such as the presence of monotonically increasing sequence numbers. A noteworthy contribution is its concept of assigning connection numbers to individual streams, allowing to maintain the compression mode even when multiple connections are open. Degermark et al. present an IP header compression scheme for IPv4 and IPv6, compatible for the use with TCP and UDP, in RFC 2507 [DNP99]. Its application can reduce an entire packet header to 4 to 7 bytes, including the 2 byte checksum used in TCP and UDP. In practice, the scheme establishes an individual context for each TCP stream by sending an uncompressed header periodically. All intermediate packets refer to the last fully transmitted header and thus only transmit differential data. Robust Header Compression (ROHC) [SPJ10] uses multiple compression states and always operates in the best compression state it can confidently use for a connection. The state selection is based on various criteria relating to successful compression and decompression (e.g., ACKs and NACKs). However, ROHC relies on steady network topologies, where sender and receiver can assume a long-term stability of their link, which is not always the case in sensor networks.

In contrast to the presented mechanisms that establish a context for all further transmissions, Stateless Header Compression [WK05] builds on the observation that the required stream management generates overhead in terms of both computation and memory. Instead, IP headers are compressed by creating shorthand aliases for addresses in the same subnetwork and replac-

ing all address fields in the headers. By assigning different levels of aggregation capabilities, further header fields are also transferred in a compressed, but stateless manner. Specifically crafted for WSNs, the IPv6 over Low-power Wireless Personal Area Networks (6LoWPAN) standard [MKHC07] targets to make IPv6 applicable in sensor networks. Inherently considering the payload size restriction present in IEEE 802.15.4 [IEE06], it also defines a header compression scheme that only makes use of stateless header compression to keep the memory consumption low. As devices in the same WSN are assumed to already share a common state, some fields (such as the used IP version or the common subnetwork prefix) present in common IPv6 headers are consistent throughout a 6LoWPAN sensor network and are thus omitted. The use of Dynamic Link Labels [KSS02] addresses the length of MAC addresses on the data link level of WSNs. As often only a limited number of communication partners are present within a node's radio range, the approach reduces the header length by assigning each node a short unique identifier.

In the presented hop-by-hop mechanisms, the existence of long routes leads to additional computational overhead for the recompression of headers at all intermediate nodes. The reduction of this overhead is addressed in SEEHOC [KSo4], which caters for end-to-end header compression. Each connection is assigned a unique connection identifier, which is composed of the MAC addresses of both sender and receiver. Management of the soft states along the path is maintained by periodic uncompressed update messages. Finally, besides hop-by-hop and end-to-end solutions, hybrid approaches exist that combine features from both worlds. Both Westphal [Weso6] and Arango et al. [APA⁺05] hereby propose to compress all headers including network layer headers on a hop-by-hop basis, while all transport layer and above streams are additionally compressed in an end-to-end fashion.

3.2.2 *Applicability of Existing Solutions in Smart Spaces*

On the whole, the field of data compression in sensor and actuator networks has been addressed in a comparably small number of publications so far. Furthermore, of all references presented above, the majority of approaches has been specifically tailored to a given sensor network architecture with a fixed data type of interest. Their applicability in generic scenarios, in which different sensor data characteristics or other network topologies are present, is rarely analyzed. However, application-independent data compression is needed in order to provide developers with a simple and efficient tool to achieve energy savings in their WSN applications. Many steps remain on the way towards a holistic compression framework that can be easily integrated into WSN applications, and automatically adapts to the characteristics of the underlying data.

The representative character of the data traces used for the evaluation of the presented solutions is also debatable, as they were commonly present in the form of continuous temperature or GPS samples. Thanks to the flexible architecture of motes, however, they can easily be extended by various sensing modalities, as outlined in the list compiled in Sec. 2.2.3. While temperature and GPS traces represent valid modalities which can be sensed using motes, sensor traces with less coherent characteristics (e.g. human acceleration patterns or noise level recordings in streets) are likely to expose different properties. An analysis of the correlation between the underlying data sets and the compression gains achieved by the proposed algorithms still remains an open issue. Only Liang et al. perform a thorough analysis of the used data sets in [LP10]. The proposed compression step algorithm is then optimized by taking a priori gained information about the symbol distributions into account. A reference corpus of sensor data — although its compilation is beyond the scope of this thesis — would provide a useful benchmark for the comparison of compression algorithms.

Especially in the domain of WSNs, practical experiences have shown that simulation results are likely to differ from real-world observations [LBV06, BISV08, HB09]. Still, the majority of presented approaches do not validate the attained simulation results in practical experiments in order to prove their function in real-world settings. Also, energy expenditure values of nodes in different operation modes are often estimated, and do not necessarily reflect the

consumption of the real hardware. Especially with regard to wireless connectivity, various parameters that cannot be reflected in simulation environments are given [KGKS05]. Practical validation in hardware emulators or real-world experiments should hence be always conducted supplementary in order to ensure the applicability of the devised solutions.

With regard to header compression, it can be observed that the majority of existing approaches have been designed for IP-based network protocols. However, although being strongly driven by industrial consortia, IPv6 has not yet evolved to the de facto standard network protocol employed in WSNs. Based on the use of IPv6, the 6LoWPAN proposal [MKHC07] addresses the compression of both network and transport layer headers (i.e. TCP and UDP over IPv6). Still, the efficient encoding of the MAC header of the underlying IEEE 802.15.4 standard has not received much attention, although it is almost exclusively being used in WSNs today. Header compression mechanisms from other application domains (e.g. the Internet) appear suitable for application on the MAC headers of WSNs, although a practical verification of their viability has not been shown in existing literature.

In summary, the major shortcoming of existing payload compression solutions is their lack of generic applicability. Thorough compressibility analyses with realistic data sets are however required in order to determine the practical use of data compression algorithms. In the case of header compression, most existing approaches focus on network and transport layer functionality, while the widely used IEEE 802.15.4 MAC header has not received significant attention. Finally, the lack of supplementary energy measurements disallows the judgement about an algorithm's energy efficiency. Positive energy savings achieved through data compression are easily changed to the opposite when high computational efforts are required to perform the compression step. A thorough evaluation of the incurred energy expenditure is hence essential in order to assess compression algorithms.

3.3 SUMMARY AND PROBLEM STATEMENT

Numerous WSN research contributions have been proposed to the lower three layers of the ISO-OSI reference model [Zim80] depicted in Fig. 6. As a result, physical connectivity between motes, i.e. the possibility for data exchange, is given. In case of platform heterogeneity with regard to the equipped sensor and actuator components however, pure network-level connectivity is insufficient to cater for the addition of new nodes and their respective functionalities into the network. Instead, for a seamless integration of new devices, mechanisms on higher layers of the ISO-OSI reference model, i.e. on the session, presentation, or application layer, are required. Adequate abstractions from the underlying node hardware must therefore be made in order to provide unified access to the offered functionality over the network [CDE⁺05, HPH⁺05].

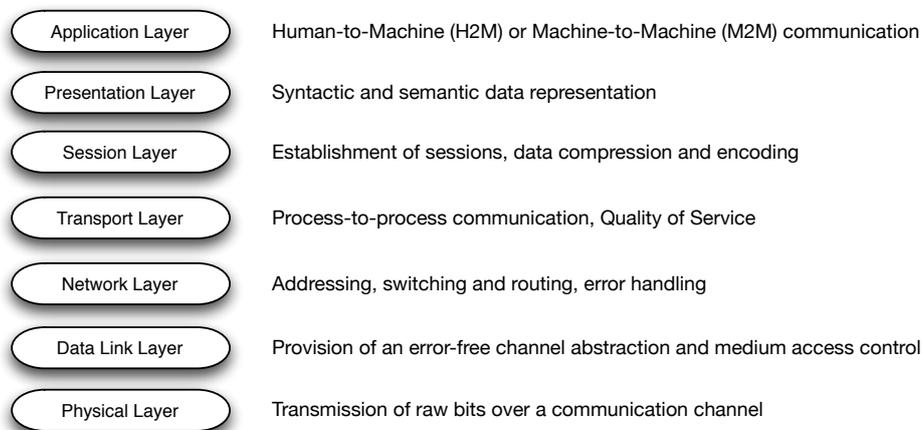


Figure 6: The seven layers of the ISO-OSI reference model with annotations partially taken from [SNo4]

Two major paradigms for the access to remote functionality have been identified in related work. While data-centric approaches rely on semantic annotations of the functionality available in the network, address-centric solutions identify remotely invocable functions based on the address of their providing node. The application of either solution creates interoperability between WSN nodes, however optimized for different networking concepts. Data-centric solutions are mostly based on the assumption of a distributed generation and consumption of data, i.e. networks without a central entity that coordinates actuation based on its collected sensor readings. In contrast, address-centric systems often rely on a central directory that maintains descriptions of all services in the network. As a major asset of address-centric solutions, the knowledge about the addresses of sensing and actuation providers effectively renders the discovery of functionality, e.g. by flooding [IGE00] or random walks [BE02], unnecessary.

Existing address-centric approaches to create interoperability between heterogeneous systems are often based on the application of the RPC paradigm [BN84], which has evolved to a well-established solution on the Internet. As opposed to hosts on the Internet however, the limited energy budget of motes necessitates the design of energy-efficient solutions. Despite this fact, our survey of interoperability mechanisms for WSNs has shown that energy efficiency has rarely been considered during their design. In fact, most approaches make use of verbose packet structures and extensive local computation, which impairs their generic applicability on resource-constrained sensor and actuator nodes. Only a small fraction of existing RPC schemes for WSNs have been designed to use compact message representations, albeit the resulting energy savings are rarely quantified. Across all surveyed solutions, none of the RPC schemes has been combined with specific means to efficiently compress the resulting messages prior to their transmission.

In our analysis of the major contributors to a mote's energy demand (cf. Sec. 2.2.4), we have observed that the radio transceiver is among the predominant energy consumers. Energy-efficient wireless communication, e.g. by the application of data compression, is thus important during the design of any sensor network application, including mechanisms to cater for node interoperability. Our survey of related work in the field of data compression has however shown that existing algorithms are generally tailored to specific sensor network applications and highly adapted to the characteristics of the underlying data. In contrast to these specialized solutions, generic and universally applicable concepts are required to cater for the extensible nature of WSNs in smart environments, where new devices are envisioned to join existing networks at any time. While both lossless and lossy data compression algorithms have been presented in related work, lossy algorithms disqualify themselves due to their lacking applicability to previously unknown data structures. As a result, generic and application-agnostic packet payload compression solutions can only operate in a lossless manner. In addition to the compression of packet payloads, mechanisms for the lossless compression of packet headers have also been proposed in order to reduce packet sizes and thus preserve energy. However, while packet header compression has a long history in the Internet, only a single stateless approach [MKHC07] is known in the domain of WSNs, and its energy demand has not been quantified.

A drawback of almost all surveyed solutions, both in the domains of interoperability and data compression, is their lack of dedicated energy analyses. Many thousand MCU operations can be performed for the same amount of energy that is required for the transmission of a single byte over the radio [BA03]. Still, savings are only possible when a solution consumes less energy for the processing and transmission of compressed packets than for the uncompressed data transmission. The optimum tradeoff between local computation and wireless communication thus needs to be found in order to maximize the resulting energy gains. To summarize our findings, address-centric solutions to provide interoperability between heterogeneous nodes have been proposed, but rarely tailored to their resource-constrained architectures. Similarly, various data compression algorithms have been adapted for WSNs, but their energy demands have not been assessed in order to quantify the achievable runtime gains. Finally, a holistic data compression framework that encompasses both header and

payload compression has not been presented in existing literature to the best of our knowledge. Based on the observations from our analysis of related research, the contributions of this thesis are motivated by the following problem statement:

Cross-platform interoperability is a major prerequisite for the realization of smart spaces. The tight resource constraints of embedded systems however necessitate solutions to be especially lightweight, both in terms of their demand for energy and MCU resources. At the same time, generic operation across diverse hardware platforms is required in order to enable the inclusion of multimodal sensing and actuation devices into the smart environment. In order to cater for long node lifetimes, the energy overhead incurred by the interoperability scheme needs to be minimal. As the radio transceiver has been determined to be the predominant consumer of electric energy on current mote hardware, the lossless compression of packet headers and payloads represents a suitable solution to this end. The tradeoff between the energy expenditure for computation and wireless communication needs to be carefully optimized in order to maximize the energy gains. As a result, energy analyses for all proposed solutions are necessary in order to validate their applicability on mote platforms and to quantify the achievable energy gains.

Make sure you don't use an overly complex package to address what may be your rather simple needs.

Brett McLaughlin

Interoperability between heterogeneous and multimodal sensor and actuator nodes represents one of the major prerequisites for the realization of smart spaces. The applicability of existing Internet standards in WSAWs is however limited due to their design for computationally powerful devices, between which virtually no bandwidth restrictions exist. We encounter the limited applicability of existing solutions by presenting the S-RPC framework [RMS11]. In a first step, we summarize the requirements to interoperability mechanisms for networks composed of small and low-power embedded systems. Based on the findings, we motivate the design decisions behind our lightweight realization of the RPC paradigm for WSAWs. After describing the concepts of S-RPC in detail, we prove its applicability in WSAWs by evaluations of the used message sizes, the incurred delays, and its energy overhead. A summary of the benefits of using S-RPC concludes this chapter.

4.1 REQUIREMENTS AND DESIGN DECISIONS

Traditionally, WSAW communication protocols define packet structures statically at compile time due to efficiency reasons, i.e. each of the fields in a radio message is hard-coded to contain a particular type of data [HSW⁺00]. While this is efficient with regard to the computational complexity (and thus the demand for energy) of accessing the data in the packet, applications must be adapted and re-deployed on the nodes whenever new sensor or actuator types are introduced into the network. Especially for long-lived deployments of sensors and actuators at large scale, the simple addition of a new sensor or actuator node results in a significant maintenance overhead.

4.1.1 Requirements

In order to overcome the aforementioned limitation, a generic approach to invoke functionality of other devices in the network is required. More specifically, an abstraction from the underlying hardware platforms is necessary in order to enable a unified access to their sensing and actuation resources. The use of remotely invocable embedded Web Services has already become the prevalent solution to administer electrical equipment by means of Machine-to-Machine (M2M) communications [Bee08]. However, this solution has not yet evolved to the standard way of interaction in the field of WSAWs [She10], because of its demand for computational resources. In fact, implementations of Web Services are almost exclusively based on Internet standards [ACKM03], such as the comparably verbose SOAP protocol or the eXtended Markup Language (XML). The application of these standards in WSAWs, e.g. in the implementations of TinySOA [RE07] or Tiny Web Services [PKGZ08], entails packet sizes that exceed that maximum transmission unit of 127 bytes (assuming operation over the prevalently employed IEEE 802.15.4 standard for wireless communication [IEE06]). The cited approaches thus necessitate the fragmentation of messages, which entails an additional memory demand. Furthermore, a measurable amount of mote resources is required for the interpretation of the XML-encoded messages. In summary, even though the implementations are based on the well-established concept of Web Services, their application on embedded sensing systems is limited due to their resource demands. An adaptation of the interoperability scheme to the capabilities of the underlying mote hardware is thus inevitable.

The transmission of sensor readings to a central instance, where it is evaluated with regard to user-specific models, has been determined as the regular mode of operation in smart spaces (cf. Sec. 2.3). Functionalities with temporal constraints, e.g. the coupling of light switches and actuation relays, however require fast responses independent of the context evaluation server. As a result, both the central entity and other nodes in the network are potential consumers of provided functionality, and should thus be able to invoke remote methods. In other words, the resulting interoperability scheme must be sufficiently lightweight such that both the provider and the consumer components can be instantiated on motes at the same time. Existing approaches like [MTSG10] disqualify themselves in this aspect, because they exploit the computational capabilities of the gateway node that converts IEEE 802.15.4-based traffic into standards understood by current computers (e.g. IEEE 802.3 or IEEE 802.11). Instead of transmitting complete function invocations to the embedded sensing systems, a reduced protocol is being used in the WSA. Although this approach results in compact message representations, it is not suitable for application in smart environments because it disallows direct interactions between motes. Furthermore, the necessity of a protocol translating gateway thwarts the idea of seamless integration of embedded devices into the Internet.

Since long before the emergence of Web Services, the paradigm of RPCs [Whi75, BN84] has been well established to access functions on remote hosts on the Internet. As opposed to the encapsulation of offered functionality into Web Services, RPCs are not inherently bound to the use of Web technologies. Instead, the generic concept behind RPCs does not define particular message transport mechanisms and representations. As a result, RPCs can be adapted to various application scenarios. Its implementations for the Internet mostly rely on TCP- and UDP-based transport combined with message representations based on e.g. the eXternal Data Representation (XDR) [Sun87]. Adaptations of the paradigm to resource-constrained WSA nodes are however possible, and also required in order to make RPCs applicable on motes. The possibility to define efficient data representation solutions is of great importance in WSAs due to the prevalence of primitive data types (e.g. boolean values or signed and unsigned integers of different sizes).

The energy limitations of mote platforms necessitate the application of energy-efficient interoperability solutions in order to maximize a node's operational time. The lightweight character of implementations that provide unified access to a mote's resources is thus highly necessary. Furthermore, MAC protocols that operate the radio transceiver in duty cycles are commonly used in order to maximize a node's operational time. As a result, packet transmissions in sensor networks may encounter measurable delays on their route from the source to the destination. A further requirement to interoperability schemes for WSAs is thus their ability to cope with significant packet transmission delays.

4.1.2 *Design decisions*

While sharing the same goal as embedded Web Services, i.e. the facilitation of access to functionality on remote devices, our S-RPC framework specifically takes the previously established requirements of WSAs into account. As a result, the major difference between S-RPC and related work is its realization of RPC functionality under the constraint that its resource demand (especially in the dimensions of computation, memory, message sizes, and energy) needs to be minimized in order to cater for its applicability on mote hardware. Instead of adapting an existing implementation of Web Services to the demands of WSAs, we have thus decided to create a new method to invoke node functionality remotely in order to meet the demand of low-power operation. Interoperability within the WSA is inherently given by following this approach, and in case connectivity to external services on the Internet is required, dedicated protocol translation devices can still be employed. Based on this reduction of the solution space, the task at hand simplifies to the creation of a suitable abstraction of sensing, computing, and actuation capabilities from the underlying hardware platform. Instead of being bound to the integration of a complete existing protocol stack, the resulting solution can be specifically tailored to the capabilities of embedded sensing devices. The

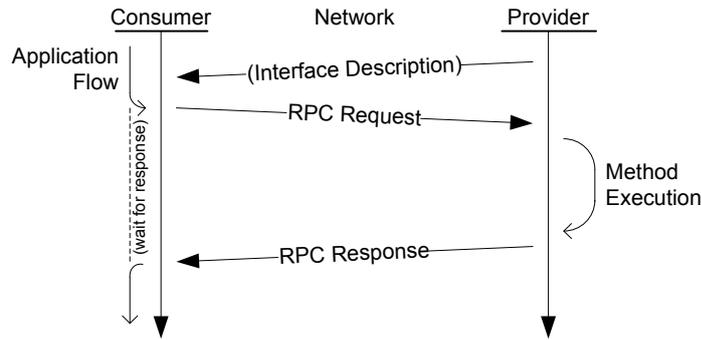


Figure 7: Simplified flow of a remote procedure call

proposed approach dedicatedly regards the limited resources of current mote platforms, which necessitate lightweight implementations.

The RPC paradigm has been used all over the Internet for decades to invoke functions on remote machines. The typical mode of operation is depicted in Fig. 7, including the optional step of describing the interfaces of the provided methods to the consumer. The interface description step can be omitted in systems where interfaces are known a priori, or when additional means for obtaining information about these interfaces are employed, e.g. service discovery [MPSHHo8] or introspection [Heno7]. Once the application flow encounters the need to invoke a remote procedure, a call is made to the local RPC stack, in which the name of the remote method as well as the invocation parameters are contained. Upon reception of the local call, the RPC stack prepares a request message, which contains an identifier of the function to invoke and the parameters, which are serialized into streams of bytes. The RPC stack then forwards the invocation message to the network layer, which caters for its transmission to the provider. Upon its reception at the provider side, the provider's network stack forwards the RPC message to the local RPC stack, which deserializes the request and invokes the defined procedure locally with the given parameter set. After its completion, the RPC stack serializes the return value of the executed function into an RPC response message, which is forwarded to the network layer in order to be transmitted to the consumer. When the consumer has received the RPC response, its RPC stack extracts the result of the invoked function from the response message and returns it to the invoking application. Due to the synchronous operation of the RPC mechanism, the consumer's application flow is interrupted until a response or error message has been received.

In contrast to the static allocation of message types or ports, the use of RPCs allows to extend a WSN by new functions at any given time because the format of exchanged messages is unified. As soon as a node has published the description of a new procedure interface, any other node in the network can invoke the method through the RPC stack. Extensibility is ensured through the definition of a generic message format, which encapsulates invocation and response messages and caters for a defined interpretation of the contents. The major design decisions that differ from the traditional definitions of RPCs are the following:

- ▷ **ASYNCHRONICITY** Traditionally, RPC calls exhibit synchronous behavior, i.e. they block the invoking application until a result has been obtained. Due to the delays incurred by energy-efficient MAC protocols and the limited processing power of motes, excessive delays might be encountered by the consumers of RPC functionality. S-RPC has thus been designed to operate in an asynchronous manner, i.e. a node's local program flow is not interrupted until a response message has been received. In order to cope with provider and connection failures during asynchronous invocations, RPC error semantics (e.g. at-least-once or at-most-once execution) are used in S-RPC. While being irrelevant for idempotent methods (operations that do not incur a state change, e.g. polling a sensor in WSNs), these semantics become relevant when actuators are controlled.

- ▷ **CONSIDERATION OF RESOURCE CONSTRAINTS** In order to ensure the applicability of the devised solution on WSAN nodes, where only small amounts of memory and limited computational resources are available, knowledge about the constraints of the underlying hardware platforms is specifically taken into account during the design phase in order to facilitate the efficient use of the available resources. More specifically, S-RPC is designed such that nodes are not limited to be either providers or consumers of functionality (such as realized in Marionette [WTT⁺06]), but provide both features without the need for reprogramming.
- ▷ **EFFICIENT DATA SERIALIZATION** Before transporting RPC messages over the network, all parameters need to be serialized into a sequence of bytes. While the serialization process of primitive data types often equals the concatenation of the individual data bytes, the representation of complex data structures and objects as a series of bytes is often confined to specific platforms and programming languages, or comparably verbose. Providing the network with generic data serializers and thus overprovisioning the data representation component clearly caters for the generic extensibility of the scheme, but it also strongly affects the required resources. Finding a trade-off between the supported data types and the resource demand (both in terms of transmission sizes and computational efforts for encoding and decoding) is thus essential.
- ▷ **EXCLUSION OF LOCAL INVOCATIONS** Instead of passing local function calls through the RPC stack in order to cater for their execution, the limitations in the available energy budget motivate the deviation from the traditional RPC definition. As a result, local calls are not processed by the RPC implementation, and local functions are thus not inherently available for remote invocations. Each local method (e.g. access to sensor readings, processing, or actuation capabilities) can however be registered to the RPC framework during runtime in order to make it available for remote invocations.

In conclusion, the concept behind the RPCs paradigm is slightly adapted in order to cater for the constraints present in WSANs. The general idea of abstracting a node's functionality into individually invocable services is however maintained, and interoperability is created through the enablement of seamless communication between embedded sensing and actuation systems.

4.2 LIGHTWEIGHT REMOTE PROCEDURE CALLS

The realization of all of the design criteria discussed above, the S-RPC framework has been designed following the structural diagram depicted in Fig. 8. A typical data flow sequence is indicated by the sequence numbers next to the arrows. Remote methods can be invoked by the consumer node through a dedicated interface offered by the S-RPC framework. Invocations are encoded by the S-RPC Data Representation Layer (DRL). All parameters are serialized by dedicated modules specifically designed to efficiently encode the given data type. The encoded message is transmitted to the method provider, where the DRL caters for invoking the necessary deserializer modules, invocation of the local method, and generation of the S-RPC response. Upon reception of a reply, the return value is deserialized and returned to the consumer application.

We have chosen to uncouple data serialization and representation to allow for a flexible configuration of the data types supported by each participating node. Each device can be fitted with the data types its own procedures consume or generate instead of enforcing support for a network-wide selection of relevant data types. The general use of complex serialization engines would disqualify S-RPC from its application on resource-constrained systems. As a result, S-RPC relies on serialization modules, allowing more powerful platforms to support a larger set of serializers while small systems can be confined to just the minimum set. The core components of S-RPC are discussed in depth in the following subsections.

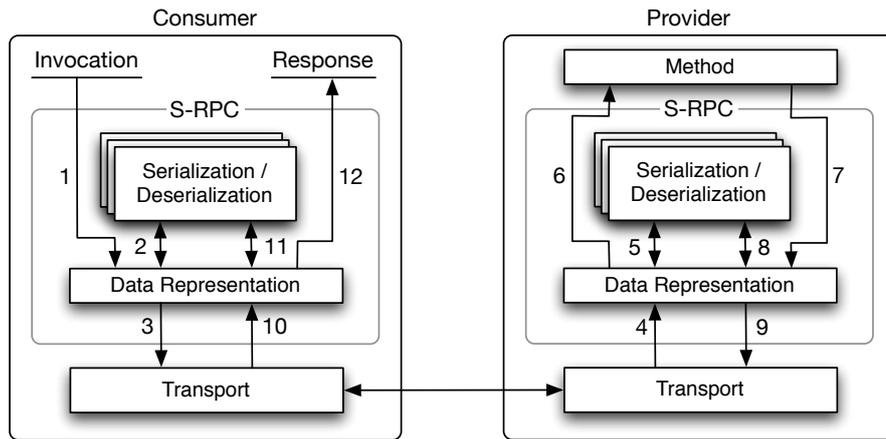


Figure 8: Internal structure of the S-RPC framework

4.2.1 Data Representation Layer

The DRL is the core component of S-RPC and fulfills the task of creating S-RPC messages from invocation calls as well as returning the responses to the requester. For methods on remote hosts that require parameters, the DRL checks the set of available serializers for their support of the given data type. If present, the corresponding serializer is invoked, returning the serialized values as well as a unique data type identifier. Otherwise, an error code is returned immediately to the invoking application. Once all parameters have been serialized, the resulting byte sequences are concatenated by the DRL and prefixed by the S-RPC header.

The header follows the structure shown in the upper four bytes of Fig. 9 for a message with three parameters, and its fields are used as follows:

- ▷ **MESSAGE TYPE** The message type field is two bit wide and specifies if the following message represents a request (binary value '00'), a response ('01'), or an error ('10'). The binary value of '11' has not been defined in the current implementation of S-RPC, and leaves room for future extensions.
- ▷ **NUMBER OF PARAMETERS** In the parameter count field, the number of data type identifiers following the second header byte is defined. Being six bits in size, between zero and 63 parameters are supported. Although significantly less than 63 parameters are used in the majority of applications that use RPCs, the field has been allocated six bits wide in order to make the header field byte-aligned.
- ▷ **INVOCATION SEQUENCE NUMBER** The invocation sequence number is eight bits in size and used to distinguish multiple calls between identical consumer and provider nodes due to the asynchronous character of S-RPC. It thereby enables the implementation of at-most-once execution semantics. An internal ring buffer for recently encountered sequence number ensures that duplicate invocations are detected.
- ▷ **DATA TYPE IDENTIFIERS** The data type identifiers are four bits in size and indicate the types of the data fields that follow the header. If their number is odd, four padding bits are inserted. For RPC request messages, the first parameter contains a byte array indicating the identifier of the function to call. Replies are generally identified through the invocation sequence number, but can optionally also carry the name of the invoked function.

The design decision to specify the data types in the header has been made to allow verification of the availability of the required deserializer module before trying to deserialize the data. Knowledge about the packet structure also allows invoked methods to verify if the provided parameters match the required parameter set before executing the call. Additionally,

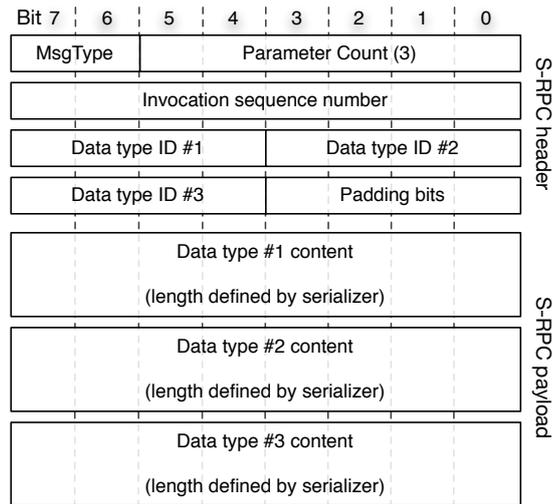


Figure 9: S-RPC representation of a message with three parameters

using the structure of contained data types as the similarity metric for stream-oriented data compression, presented in detail in Sec. 5.2, becomes a viable way to allow packet size reductions and thus preserve energy. An analysis of the compression gains achievable when S-RPC is combined with means for lossless payload compression is presented in Sec. 5.4.5.

4.2.2 *Serialization Modules*

While full RPC implementations are designed to support serialization of a broad variety of data types, the actual number of different data types in a WSN is usually limited. Especially the absence of complex objects leads to a small number of data types, e.g. signed and unsigned integers of different sizes, strings, and arrays of primitive data types. Providing the RPC implementation with generic data serializers would thus lead to increased resource consumption, although the implemented functionality is never being used. We have hence decided to use dedicated serialization modules for each data type anticipated to be present in the network. Instead of utilizing a single serializer/deserializer component, S-RPC employs a modular serialization concept, which allows the developer to fit nodes with serializer components on demand. Each serializer bears a unique identifier for the data type it represents, allowing the decoder to interpret the corresponding content correctly. As the identifier fields in our reference implementation have been defined to be four bits wide, up to 16 data types can be represented in total.

Each serializer is responsible for performing efficient conversion of its input data, i.e. means towards lossless data reduction (e.g. adaptive Huffman coding as introduced in Sec. 5.3) amongst array elements, or the application of efficient string encodings. If further data types, such lists of key-value pairs, need to be transferred between applications, corresponding serializer/deserializer components can be included. In our reference implementation, we have defined data type assignments as shown in Table 5. Some type identifiers have been intentionally left unassigned to allow for the later adaptation to the intended application. As shown in Fig. 9, the length of serialized values is determined by the data type itself and is either fixed or stored in the serialized implementation. Each deserializer thus returns both the value stored in the S-RPC data type field as well as the pointer to the beginning of the subsequent entry in the sequence. Obviously, when deserializer modules are unavailable, the correct offset cannot be returned, possibly leading to incorrect deserialization. However, if data types are unknown to a node, it can generally be assumed that no local method makes use of them, and thus that an error message can be returned immediately.

Table 5: Data type assignments used in the S-RPC implementation

ID	DATA TYPE	SIZE	REMARKS
0	null	0 bits	indicates void results
1	boolean value	8 bits	uses 7 padding bits to stay byte-aligned
2	8bit unsigned integer	8 bits	
3	8bit signed integer	8 bits	
4	16bit unsigned integer	16 bits	
5	16bit signed integer	16 bits	
6	32bit signed integer	32 bits	
7	byte array (string)	variable	output prefixed by length
8	array of a single type	variable	number of elements defined in a subheader
9	array of mixed types	variable	element types defined in a subheader
14	Java serialized object	variable	payload defined by the Java serializer
15	n/a	n/a	reserved for future extensions

Besides the presence of serialization modules that encapsulate the contained values in a serialized form of fixed length (such as boolean and integer values), array types indicate the number and types of contained elements in a specific subheader structure which prefixes the serialized representation. The subheaders for data types ‘8’ and ‘9’ (i.e. arrays of a single or mixed data types) are shown in Fig. 10. For an array that contains elements of a single data type only, the subheader is visualized in Fig. 10a. The first byte of the header contains the identifier of the data type and the number of elements in the array, and is followed by the serialized representations of the data. In case more than 15 elements need to be stored in the array, the bit value of “1111” is used as an escape sequence, which indicates the presence of an additional length field of four bytes in size that directly follows the header byte. In contrast, Fig. 10b shows the header for an array of different types, in which the number of entries is stored in the first byte, followed by the data type identifiers for each entry. Similarly, when more than 256 types are required, the bit value of “1111 1111” is used as an escape sequence to indicate that the first array subheader byte is followed by a four byte length field.

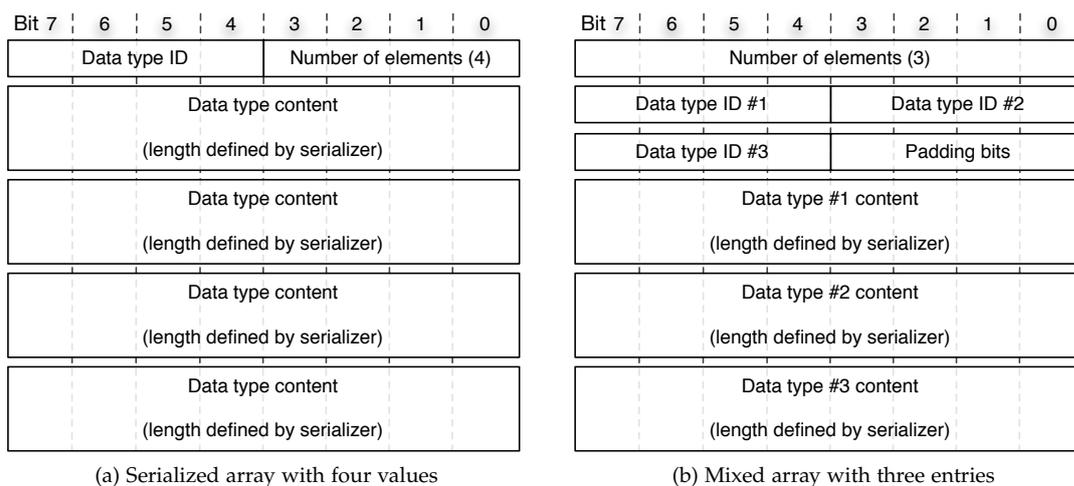


Figure 10: Subheaders for arrays of identical and mixed types

4.2.3 Message Transport and Error Handling

Due to the fact that energy-efficient MAC and data transport protocols are frequently employed in order to achieve long node lifetimes, S-RPC has been designed to work in an asynchronous manner. As a result, sequence numbers have been added to all S-RPC messages in order to link response and request messages. Each mote thus maintains a buffer, in which the originating node's address and the used sequence number are entered after an invocation has been processed. In order to permit the reuse of sequence numbers, buffer entries are purged after a timeout period. As a result of its asynchronous character and the use of sequence numbers however, S-RPC is confined to the realization of best-effort execution and at-most-once invocation semantics.

Even though the realization of transport functionality is beyond the scope of an RPC implementation, it has an impact (e.g. unsuccessful packet transmissions) on the operation of the RPC scheme. In order to recover from remote node failures, packet losses, or excessive delays, timeout mechanisms are integrated into S-RPC. If no response has been received within a specific time frame, the call is considered as *maybe* successful, i.e. a possible actuation command may have been received and executed. In case the provider node gets disconnected from the network, an error message can be generated at intermediate nodes to provide the caller with details about the problem and thus terminate the call.

4.3 EVALUATION

Verifying that a WSN application works as expected is commonly not a straightforward task. The formal verification of algorithms has been applied successfully [OT09] in order to prove the correct operation of the application prior to its deployment. However, due to the resource constraints present on current motes (cf. Sec. 2.1 for details), the application complexity is commonly rather low. In consequence, formal verification of applications on a node-local scale is used less extensively. Instead, network simulation approaches are prevalently employed, because wireless networking represents one of the major characteristics of WSNs. Finally, the correct operation of applications for sensor and actuator networks can also be validated through practical experimentation. This practical functionality assessment has proven to be a valuable tool in the development of WSN applications, because algorithms proposed for sensor and actuator networks have shown different, sometimes even contradictory, results between simulation and practical deployments [LBV06]. The observed discrepancies typically originate from several factors, e.g. the lack of realism in the radio channel or sensor models. The simulations presented in this section are thus supplemented by a practical evaluation of the solution under real-world channel conditions.

An implementation of the S-RPC stack for TelosB motes [MEM06b] has been written in the C programming language for the widely used TinyOS operating system [HSW⁺00], which represents the prevalent operating system in sensor networks research. In order to quantify the overhead introduced by the S-RPC stack, a second application with identical functionality that does not make use of the S-RPC abstraction has also been created. The resulting resource requirements are then compared by regarding the difference in the resource consumption in terms of both program memory and RAM. After having verified the applicability of the solution through an analysis of its resource consumption, the implementations are instantiated in the COOJA/MSPsim simulation suite [EOF⁺09b]. The suite is comprised of the COOJA network-level simulation environment and instruction-level emulators for mote platforms based on MCUs manufactured by Atmel or Texas Instruments. The underlying TelosB platform is based on a Texas Instruments MSP430 series MCU, and its emulation is hence fully supported in MSPsim. As a result, the platform's energy demand can be precisely determined [EOF⁺09a]. In order to fully prove the operation of S-RPC on mote hardware, real-world experiments are conducted. In our experiments, a simple topology with a single consumer and a single provider node has been set up. Besides catering for the validation of the results achieved through simulation, experimentation is inherently prone to the physical

Table 6: The remotely invocable functions used for the evaluation of S-RPC

METHOD INTERFACE	DESCRIPTION
byte[] ping(void)	returns the byte array pong
byte[] cat(byte[] a, byte[] b)	concatenates the two input arrays to a single array
void ledsOn(void)	turns the node's LEDs on
bool xor(bool a, bool b)	returns the value of $a \oplus b$
int add(short a, short b)	returns the value of $a + b$
int diff(short a, short b)	returns the value of $a - b$
int sum(short[] a)	returns the value of $\sum a$
int[] echo(int[] a)	echoes the input array of 32 bit integer values

characteristics of the real environment. The impact of real-world conditions on the operation of S-RPC is thus assessed in our real-world experiment.

4.3.1 Resource Utilization

In contrast to smartphones or laptop computers, WSA nodes are based on embedded systems which are generally operated on a sources with a finite energy budget. A solution to perform RPCs must hence be sufficiently lightweight to be supported even on the smallest participating platform. In order to assess their resource demand, both the reference implementation as well its extension by the S-RPC stack were first implemented without any remotely invocable methods. In a second variant, the eight functions described in Table 6 were added to the implementations and thus made available for remote invocation. The resulting resource consumptions in terms of program memory and RAM are shown in Table 7. When incorporating the S-RPC stack without any functions offered, a mere size increase of 656 bytes application ROM (equalling 1.33% of the MCU's total program memory) as well as 138 bytes of additional memory (representing 1.35% of the available amount) are occupied. Even when all eight functions are included, only 1.8 kilobytes of program memory (3.6% of the available resources on the TelosB platform) and only 204 additional bytes of RAM (representing 2% of the available memory on the TelosB) are required. This resource demand of less than 4% of both application memory and RAM matches the requirement of applicability on motes well, and strongly contributes to the practicality of S-RPC.

The S-RPC framework caters for the communication and transparent serialization and deserialization of messages. The design decision to decouple the local invocation of methods from their remote use has led to the necessity of defining which methods are available for remote invocation. Methods can be made available for remote invocation by registering them to the framework at any given time, and similarly, methods can be deregistered from the framework, e.g. when the available energy budget disallows offering energy-intensive services. As a result, the application and operating system can dynamically adjust which functionality is offered to remote nodes.

Table 7: Resource requirements of the TinyOS implementation of S-RPC

	NO INVOCABLE METHODS		EIGHT INVOCABLE METHODS	
	ROM	RAM	ROM	RAM
Reference (without S-RPC)	16720 bytes	1210 bytes	17372 bytes	1374 bytes
Implementation of S-RPC	17376 bytes	1348 bytes	19256 bytes	1578 bytes
Difference	656 bytes	138 bytes	1884 bytes	204 bytes

Listing 1: Implementation and metadata definition of the add(int a, int b) function

```

/* Method implementation */
char* add(char* i1, char* i2) {
    return int2srpc(srpc2int(i1) + srpc2int(i2));
}

/* Metadata definition */
const struct methodStruct add_func = {
    .name = "add";
    .paramCount = 2;
    .parama = IntegerDataType;
    .paramb = IntegerDataType;
    .returnType = IntegerDataType;
    .function = &add;
};

/* Registration to the S-RPC framework */
add_srpc_method(add_func);

/* Unregistration from the S-RPC framework */
add_remove_method(add_func);

```

In order to make a function available for remote invocations, the corresponding function must be programmed, and its metadata must be registered to the framework. The basic implementation of a method to add two integer numbers is shown in Listing 1, wherein the helper functions to convert integer values to and from their serialized form (e.g. `srpc2int`) are provided by the S-RPC framework. The `methodStruct` prototype is provided by the framework, and must be instantiated for every function available for remote access. It defines the name of the invocable method, the numbers and types of parameters that need to be passed along with the invocation message, and the data type returned by the function. Finally, a function pointer is being used to refer to the actual implementation of the functionality. In order to register the function to the framework, the `add_srpc_method(methodStruct)` method needs to be called with the metadata structure as the parameter. Similarly, functionality can be unregistered during runtime (e.g. due to a lack of energy) using the `remove_srpc_method(methodStruct)` function.

4.3.2 Message Size

Packet size limitations and low channel bandwidths require the use of small packets in WSANs. Considering that packet sizes in IEEE 802.15.4-based networks are confined to a maximum of 127 bytes, of which 11 bytes are required for the IEEE 802.15.4 header, at most 116 bytes are available for the packet payload. For packet payloads that exceed this size limit, fragmentation is necessary and an increased memory demand is incurred. Avoiding the need for fragmentation through the use of small message sizes is thus a desirable feature in WSANs. The message sizes to invoke the functions on remote devices are compared to both the aforementioned reference implementation without RPC features and the TinyRPC approach [MDDH06], which is based on the generation of method stubs at both consumer and provider before the RPC functionality is available for use. In the reference implementation, a field of one byte in size has been added to indicate the method to be executed in the incoming invocations. In the TinyRPC implementation, a header of seven bytes size indicates which method to execute. In contrast, all S-RPC invocations carry the full name of the called method, as well as all parameters including their types. In order to compare the invocation sizes to existing and widely used Web Service standards, the corresponding SOAP [ML07] and JSON-RPC [JSO10] message sizes (without consideration of the HTTP headers) are also

Listing 2: SOAP representation of the invocation of `cat("foo", "bar");`

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:cat xmlns:m="http://www.example.org/rpc">
      <m:param1>foo</m:param1>
      <m:param2>bar</m:param2>
    </m:cat>
  </soap:Body>
</soap:Envelope>
```

Listing 3: JSON-RPC representation of the invocation of `cat("foo", "bar");`

```
{ "jsonrpc": "2.0",
  "method": "cat",
  "params": ["foo", "bar"],
  "id": 1 }
```

regarded in our analysis. For reference, the used SOAP and JSON-RPC packet structures are indicated in Listings 2 and 3, respectively.

The results are presented in Table 8 and show that the reference implementation is capable of invoking parameter-less methods using only a single byte of payload (i.e. the identifier of the method to execute). In contrast, the XML-based SOAP protocol requires more than 200 bytes to invoke the same method. As this exceeds the packet size limits in IEEE 802.15.4-compliant sensor networks, the transfer of SOAP messages in WSNs would make fragmentation and the resulting increased memory demand for buffering and parsing necessary. In contrast, both RPC implementations for WSNs show almost comparable results, with TinyRPC using slightly smaller packets in some cases. This difference can however be accounted to the fact that both the reference implementation and TinyRPC reduce the full method name to a one byte identifier, whereas S-RPC carries the complete method name in its parameter list. S-RPC only introduces $\lceil 2 + \frac{\#parameters}{2} \rceil$ bytes of overhead per packet for the realization of the RPC functionality. With invocation message sizes of 7 to 21 bytes in size for the given set of methods, S-RPC compares well to existing RPC approaches for sensor and actuator networks while it requires significantly smaller message sizes than solutions from the area of Web Services.

Table 8: Comparison of request message sizes in different RPC implementations

METHOD CALL	REFERENCE	S-RPC	TINYRPC	JSON-RPC	SOAP
<code>ping();</code>	1 byte	7 bytes	7 bytes	59 bytes	220 bytes
<code>echo({1, 2, 3});</code>	13 bytes	21 bytes	19 bytes	66 bytes	320 bytes
<code>cat("foo", "bar");</code>	9 bytes	15 bytes	15 bytes	70 bytes	293 bytes
<code>ledsOn();</code>	1 byte	9 bytes	7 bytes	61 bytes	222 bytes
<code>xor(true, false);</code>	3 bytes	9 bytes	9 bytes	69 bytes	296 bytes
<code>add(1024, 2148);</code>	5 bytes	11 bytes	11 bytes	68 bytes	295 bytes
<code>diff(728, 8210);</code>	5 bytes	12 bytes	11 bytes	68 bytes	296 bytes
<code>sum({3, 2, 1});</code>	7 bytes	20 bytes	19 bytes	65 bytes	318 bytes

4.3.3 *Invocation Delay*

Because of the confined energy budgets available to WSN nodes, an application’s energy consumption is a crucial contributing factor to its applicability in real deployments. In a first step, we thus prove the feasibility of our design decision to exclude local invocations through an analysis of the energy and delays introduced by the use of the S-RPC framework on a node-local scale. Subsequently, the analysis is extended to assess the effect of invocations over the wireless network on the energy consumption and delay.

To monitor the time required to process invocations, we have implemented a function to repeatedly invoke each of the eight methods shown in Table 8 locally at the provider node. The resulting application has been deployed on a TelosB node, which was additionally configured to indicate the start and end of the measurement period by toggling the state of one of its GPIO pins. The local invocation delay was then determined by measuring the time during which the pin was active. The average invocation delays for local execution of each method are stated in Table 9. The encountered delays of these local invocations show a measurable discrepancy on the TelosB mote: Less than $20\mu\text{s}$ are required to execute the actual processing, while the time required to deserialize the S-RPC data, invoke the contained method, and serialize the result into a response message ranges from $360\text{--}375\mu\text{s}$, depending on the complexity of the used data types. As a result, the decision to deviate from the original definition of RPCs and exclude local calls from invocation through the S-RPC stack has proven to be feasible, because the measured invocation delays indicate that significant processing overhead is omitted by using direct function invocations.

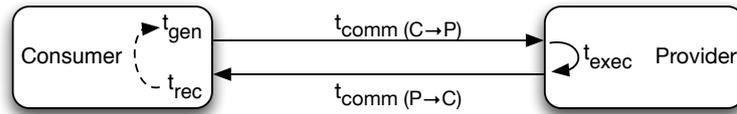


Figure 11: Topology for the experimental evaluation of S-RPC

In a second experiment, an analysis of the delays on the network level has been conducted, for which the topology depicted in Fig. 11 has been used. The consumer node was configured to continuously emit S-RPC invocations whenever a response for the previous call had been received. An additional watchdog timer has been integrated to reduce the impact of packet losses. It has been set to a timeout interval of one second, i.e. if no response is received within the timeout interval, a packet loss is assumed and the next S-RPC invocation message is emitted. The second node was set up as the S-RPC service provider, with its only task being the processing of RPCs. As opposed to local invocations, several elements contribute to the overall delay when networking aspects are considered. For the analysis of the invocation delay, we have thus dissected the total delay into its contributing elements. The total delay t_{total} of a communication sequence is thus shown in Eq. 2, where t_{gen} is the time required for the generation of the invocation, t_{comm} represents the delay of the wireless communication, t_{exec} denotes the time required to execute the method at the provider side and generate the S-RPC response, and t_{rec} is the delay of the process that receives and validates the response at the consumer node.

$$t_{\text{total}} = t_{\text{gen}} + t_{\text{comm}(C \rightarrow P)} + t_{\text{exec}} + t_{\text{comm}(P \rightarrow C)} + t_{\text{rec}} \quad (2)$$

In order to eliminate the impact of the MAC protocol on the node’s energy consumption in the simulations, we have configured COOJA to use the NullMAC protocol implementation. In contrast to other MAC protocols, NullMAC does not employ duty-cycling of the radio transceiver, but instead keeps it active at all times. Thus, it effectively eliminates communication delays due to the impact of the MAC protocol and allows for a better assessment of S-RPC’s impact on the invocation delay. Both the provider and the consumer node have been configured to use NullMAC in the experiment.

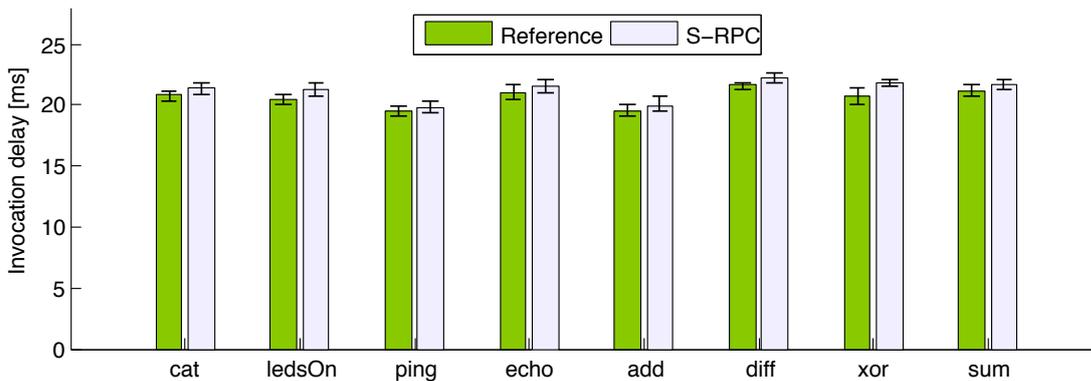


Figure 12: Experienced delays for the invocation of methods

By analyzing more than 3,200 invocations to the `add(1024, 2148)` method in the simulation environment, the following average delay values have been determined: $t_{gen} = 7.52\text{ms}$, $t_{comm(C \rightarrow P)} = 3.01\text{ms}$, $t_{exec} = 1.83\text{ms}$, $t_{comm(P \rightarrow C)} = 6.52\text{ms}$, and $t_{rec} = 1.57\text{ms}$. The resulting overall invocation delay therefore sums up to $t_{total} = 20.45\text{ms}$, with the execution time t_{exec} contributing a fraction of less than 9%. In an supplementary experimental evaluation, 5,120 invocations have been sent by the consumer to the provider for each of the eight methods. The resulting average delays measured for remote invocations as well as the minimum and maximum values are depicted in Fig. 12. In contrast to local method calls, it becomes clear that the processing duration becomes almost negligible when compared to delay incurred by the radio communication. As expected from the aforementioned simulation of a remote invocation, the durations of processing at the consumer side (t_{gen} and t_{rec}) and communication between the nodes ($t_{comm(C \rightarrow P)}$ and $t_{comm(P \rightarrow C)}$) dominate the delay. The execution of the S-RPC stack introduces less than a millisecond of delay in all invocations.

4.3.4 Energy Consumption

After the analysis of its resource demand, message sizes, and invocation delays, we have analyzed the energy overhead entailed by the application of S-RPC. We have conducted two simulations in order to assess the energy overhead of S-RPC in contrast to the use of proprietary means to invoke remote procedures. In a first step, we have quantified the additional energy expenditure when invocations of local procedures are processed by the S-RPC stack. Subsequently, we have conducted an analysis of the relative energy overhead incurred when S-RPC is employed to cater for the invocation of remote functionality.

In order to analyze the energy demand of local invocations, a single node has been configured to repeatedly invoke the `ping()` function locally without any wireless communications taking place. According to the COOJA/MSPsim environment, each the local invocation and processing of the function entails a demand for 2.56mJ of energy. The energy consumed to process a single call on the MCU increases to 2.81mJ when S-RPC is being used for local function invocations. In other words, an additional energy demand of 9.4% is required when S-RPC is applied to local invocations. This observation clearly confirms our aforementioned decision to exclude local invocations from being managed by the S-RPC stack.

Table 9: Invocation delays for local invocations with and without the S-RPC stack

	PING	ECHO	CAT	LEDS ON	XOR	ADD	DIFF	SUM
Direct invocation (μs)	11.0	5.0	4.9	4.9	12.9	4.8	4.8	18.3
S-RPC invocation (μs)	366.6	370.5	373.1	364.0	364.0	367.1	366.7	371.6

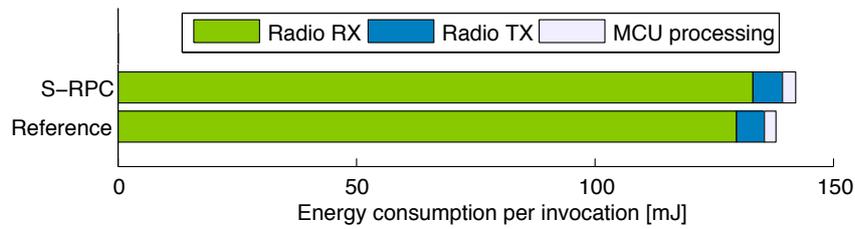


Figure 13: Experienced energy consumption for remote method invocations

For the analysis of the remote invocation overhead, we have again assumed the topology depicted in Fig. 11, i.e. a single provider and a single consumer node within each other's communication ranges. Analog to the previous experiment, we have configured TinyOS to use the NullMAC implementation in order to eliminate the impact of the MAC layer on the energy demand of S-RPC. In order to eliminate the energy demand introduced by the execution of a procedure's computation we have configured the consumer node to emit queries to a procedure that does not perform any computations, but instead returns a success message in any case. A total of 5,120 RPC requests were emitted at maximum rate, i.e. an invocation was sent whenever the response to the previous invocation had been received. The simulation results of the S-RPC provider node's average energy demand for a complete invocation sequence is depicted in Fig. 13. It can be seen in the figure that the additional expenditure to process S-RPC requests is comparably small; it only represents 0.11% of the total energy consumption. In contrast, the overall energy demand is dominated by the consumption of the radio transceiver in both S-RPC and the reference implementation.

As a result, the energy analysis has shown that the use of S-RPC on average requires 0.25mJ of energy. While this represents a significant fraction of the energy demand of average local invocations, its impact is strongly diminished when invocations are received over the radio. S-RPC thus fulfills the requirement of a low energy overhead by confining the additional energy requirement for the processing of S-RPC messages to a very small amount of just 0.11% when compared to the total energy demand of a remote invocation.

4.4 SUMMARY

We have addressed the need for seamless interoperability between motes in smart spaces by presenting the concept and design considerations of the S-RPC framework in this chapter. S-RPC has been specifically adapted to its use in WSANs by considering the requirements of embedded sensing and actuation systems during its design. In contrast to the prevalent approach of adapting existing approaches from the field of Web Services, we have taken the decision to devise a new solution due to the significant computational overhead of existing means to invoke functionality on remote hosts. In summary, two major design decisions have been taken for the successful application of the RPC paradigm in WSANs for smart environments:

1. We have exploited the strong prevalence of primitive data types in such networks to optimize the performance under the imposed resource constraints. The resulting modular serialization concept enables motes to selectively provide support for certain data types and thus reduce the resource demand of their S-RPC implementation. Motes only need to be provisioned with serialization modules for data types they either consume or generate.
2. We have defined a compact data representation scheme to limit the additional overhead. Despite its compact size, the resulting binary header structure introduces flexibility with regard to the contained payload fields, and thus enables the use of RPCs to create interoperability between heterogeneous nodes.

By having established a method for the generic integration of heterogeneous sensor and actuator nodes into a wireless network, the foundation for the realization of Ambient Intelligence and smart environments has been laid. Our analysis of its applicability on embedded sensing and actuation systems has however shown that the achieved interoperability comes at a price; a measurable energy demand has been determined when invocations need to be processed by the S-RPC stack. Because energy-efficient operation is a major requirement for battery-operated sensor and actuator nodes, suitable means to reduce the overall node's energy demand are required. We directly address this requirement by presenting solutions for the energy-efficient transfer of data over wireless communication channels in the following chapters.

Perfection is achieved, not when there is nothing left to add, but when there is nothing left to take away.

Antoine de Saint-Exupery

In this chapter, we discuss the benefits of lossless payload compression in WSNs, and analyze its impact on the runtime of nodes with a fixed energy budget. Data compression is a viable step for the preprocessing of sensor data prior to their wireless transmission in order to reduce their size and thus their transmission duration. Energy-efficient operation and thus extended node lifetimes are however only achieved when less energy is required for the processing step and the transmission of the shorter packet than when sending the data uncompressed. In a first step, we present an initial analysis of the compressibility of sensor data, and quantify the corresponding achievable energy savings. Based on the findings from the analysis, we highlight the implications of applying data compression algorithms in sensor and actuator networks. Subsequently, we analyze the achievable compression gains when differential coding is applied to compress differences between successive outgoing packets and devise the Squeeze.KOM compression layer, which caters for a transparent compression of payload data. In addition to the compression of packet differences, we present an adaptation of the Adaptive Huffman Coding (AHC) algorithm, which we have specifically tailored to its application in WSNs. By conducting an analysis of the energy expenditure resulting from the application of our AHC implementation, we highlight the parameter selection necessary for the successful application of entropy coding. For the validation of the devised data compression solutions, we describe our experimental data collection setup in an office environment and present characteristics of data traces collected in the domains of body-area sensing, environmental monitoring, and building surveillance. We conclude this chapter with a summary of the results achievable by applying lossless data compression in sensor and actuator networks.

5.1 FEASIBILITY OF DATA COMPRESSION IN SENSOR NETWORKS

Before developing data compression algorithms for WSNs, we analyze the feasibility of payload compression in a preliminary simulation study [RCHS09]. In the feasibility study, we use representative data sets from the Porcupine sensor platform, which performs person activity tracking through readings of on-board accelerometers [VGM06]. The goal of the simulation study is the assessment of the achievable energy gains when data compression is applied prior to packet transmissions. Our simulations specifically focus on the sensor node's MCU and radio energy consumptions, because data compression in WSNs targets to minimize the energy required for a data transmission at the cost of an increased energy demand for computation. Due to the fact that computation is less expensive in terms of energy than radio communication (cf. Sec. 2.2.3), data compression appears as a viable way to reduce a platform's overall energy consumption.

5.1.1 *Simulation Setup*

In the application scenario of smart environments, data collected in the underlying WSN often needs to be transferred to other motes or an external base station periodically with low latency. Similarly, in case remote procedure calls as introduced in Sec. 4.2 are being used, responses to invocation messages are required in a timely manner, and cannot be delayed

for extended periods of time, e.g. to perform data processing like aggregation [KEW02] or network coding [LYCo3, Mog10]. Although in a regular state of operation, successive readings of the same sensor might not significantly differ from each other, we base our analysis on the assumption that all values need to be transferred in a lossless and timely manner. In the simulation, we hence analyze an ideal (i.e. lossless) one-hop communication channel, over which a sensor node periodically delivers data to a sink.

In our feasibility study, we have used the RLE algorithm [Gol66], which replaces multiple occurrences of an input symbol by a repetition count field. Although a broad variety of compression algorithms exist, the RLE algorithm has been specifically selected because of its low computational complexity and the corresponding low resource consumption. The algorithm has been configured to use a threshold value of two, i.e. whenever two or more successive identical symbols are encountered in the input sequence, all subsequent occurrences of the symbol are represented by a field containing the length of the symbol run. We have added a one byte status field preceding the packet payload in our implementations, which indicates whether the following payload contains compressed or uncompressed data. Although the compression step is always performed, compressed data is only sent if its size is smaller than the uncompressed data. In case of incompressible data, the packet size to be transmitted thus increases by one byte due to the status field. With RLE being a comparably simple approach to reduce the size of packet payloads, its implementation on the TelosB node requires only 654 bytes of program memory and 44 bytes of RAM (a resource utilization of less than 1.5% in both regards).

Our simulation study is comprised of two parts. Simulations are initially performed with synthetic data in order to evaluate the achievable best and worst case results. In a subsequent step, we have used traces from the Porcupine [VGM06] deployment, in which the wearer’s current activity is inferred from acceleration readings taken from an accelerometer sensor worn on the wrist. Through the use of excerpts collected by a Porcupine node, we assess the real-world viability of data compression in WSANs. In both simulations, we have used the COOJA/MSPsim simulation environment with hardware emulation capabilities [EOF+09b] in order to assess the energy demand of the data compression algorithm.

5.1.2 Results for Synthetic Data

Since the fundamental operation of RLE is based on the presence of runs of successive identical symbols within the input sequence, we analyze the impact of the length of these runs in a first step. A packet payload size of 21 bytes has been chosen for our analysis, identical to the payloads used on the Porcupines (cf. Sec. 5.1.3). In order to assess the impact of the number of successive symbols, we have generated 21 synthetic sensor traces, which are composed of k identical bytes (which will be efficiently encoded using RLE) followed by m differing values. In case the sum $k + m$ is less than 21, the sequence is repeated. An exemplary payload for $k=9$ and $m=12$ is shown in Fig. 14.

We have conducted analyses of the energy consumptions of MCU and radio transceiver for all 21 possible values of k (with $m=21-k$), and show the results in Fig. 15. All simulations are based on the TelosB platform, which has been completely emulated in order to attain representative energy figures. In the figure, the total energy demand is composed of the weighted sums of MCU and radio consumptions. The radio transceiver has a higher impact on the total energy demand because of its larger operating power (cf. Sec. 2.2.3). Similar to the previous evaluation of S-RPC, the NullMAC protocol has been assumed for the medium

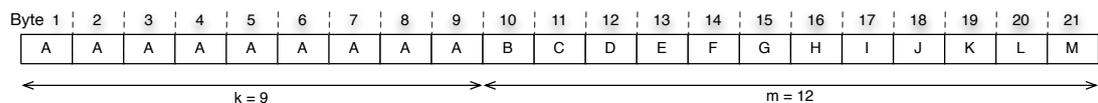


Figure 14: Exemplary synthetic payload with $k=9$ identical symbols followed by $m=12$ differing ones

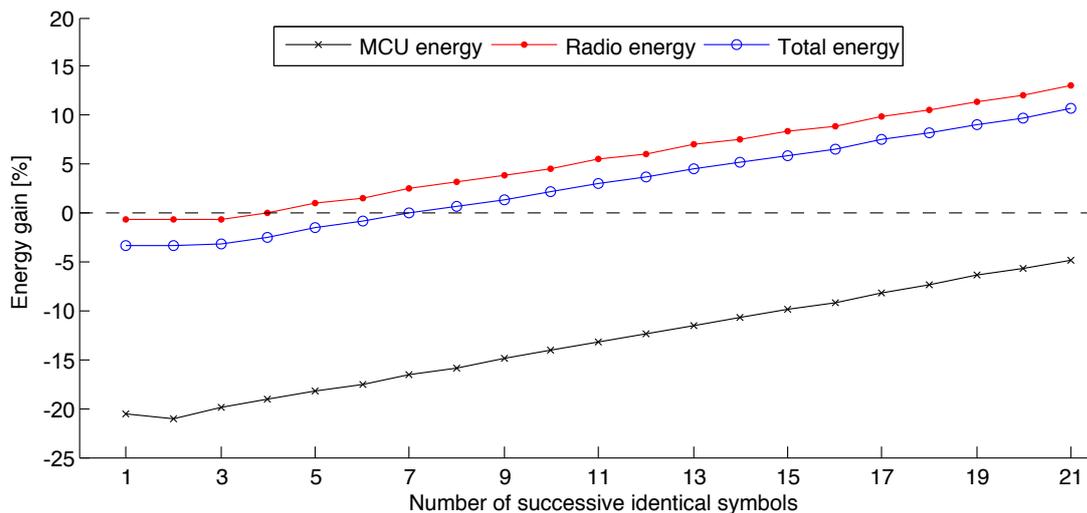


Figure 15: Energy gains for data with k of successive identical symbols and $m=21-k$

access in order to eliminate the influence of the MAC protocol on our energy analysis. In consequence, the radio transceiver is activated only when a packet needs to be transmitted, and put into sleep mode otherwise in order to preserve energy. From the figure, it becomes apparent that the MCU experiences energy losses of more than 20% when it needs to compress an input sequence composed of incompressible data (i.e. the value of k ranging between 1 and 3). Similarly, the radio energy gain is slightly negative (at -0.6%) due to the introduced one byte overhead that is required to discriminate between compressed and uncompressed data. By summing the weighted contributions of MCU and radio up, an energy loss of slightly more than 3% in comparison to the reference implementation is observed in the case of incompressible data.

As soon as four or more identical symbols are present in the input sequence, their replacement by two repetitions of the encoded symbol, followed by the length of the run, results in an output size of only three bytes. This size reduction of the packet payload directly results in shorter transmission duration and, as a result, in energy savings. In the figure, this transition is observed from a value of $k=5$, because we have also taken the transmission of the status byte into consideration. From $k=3$ on, a positive trend of the energy gain can be observed for increasing values of k , both in terms of the radio and the MCU energy. In case of the radio transceiver, the transmission of shorter packets leads to a faster return into the low-power sleep mode. In contrast, the MCU requires a constant amount of energy to perform the RLE operation. Its energy gains can however be attributed to the shorter time period required to transfer the packet to the radio transceiver over the on-board interconnection. Overall, positive energy gains can be achieved for any sequence of 7 or more consecutive identical symbols, and energy savings up to 10.7% can be attained in the best case of 21 identical symbols in the input sequence.

For the analysis of the worst case energy loss, the input sequence needs to be composed of multiple symbol runs of the same length as the threshold. We have hence conducted two supplementary analyses using the packet payloads visualized in Fig. 16. For the payload composition shown in Fig. 16a, the application of the RLE algorithm would lead to an increase of the output size by a factor of $4/3$, i.e. an increase from 21 to 28 bytes for the given packet, while the payload shown in Fig. 16b would even result in an output size of 31 bytes, i.e. a compressed sequence that is almost 50% larger than the input. Because our implementation does not transmit packets that would be larger than the original payload, it inherently limits the maximum energy losses on the radio. The results for both payloads depicted in Fig. 16 have shown energy losses identical to the case where $k=2$ and $m=19$, i.e. 21.1% of MCU energy losses and 0.6% of additionally required radio energy. In summary, the compression of

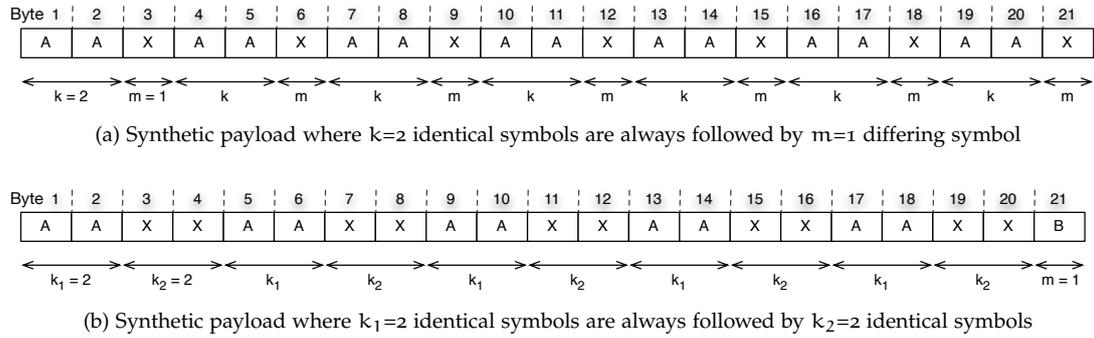


Figure 16: Data sets used for a worst case analysis

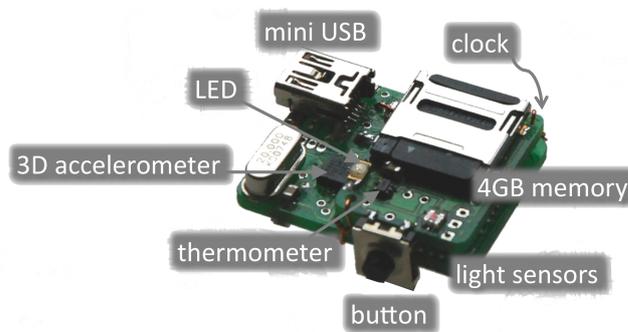
incompressible data leads to overall platform energy losses of up to 3.3%. In contrast, while up to 10.7% of energy can be saved in case only a single symbol is present in the input sequence.

5.1.3 Results for Real-World Data

As highlighted in Sec. 3.2.2, the use of synthetic data traces does not cater for real-world conditions. After having analyzed the boundaries of operation for synthetic data, we thus analyze the performance of the RLE algorithm with realistic data in this section. The wearable Porcupine nodes (illustrated in Fig. 17) have been designed to determine a person's activity from accelerometer readings. An exemplary plot of the acceleration data is given in Fig. 18, showing a 10 minute excerpt of the acceleration readings, in which a phase of the user's activity is interrupted by an interval without motion. In the figure, all acceleration readings are shown as signed byte values. The payloads of all transmitted packets consist of seven accelerometer readings in X, Y, and Z dimension of one byte each. Acceleration readings are collected every 60ms, i.e. a packet transmission is triggered every 420ms.

In order to analyze the energy gains under realistic conditions, we have selected three representative sequences of accelerometer readings from the Porcupine project. The Porcupine data set has been specifically selected because of the fact that it is composed of discrete samples from a continuous physical phenomenon. Due to this property, it bears the possibility for multiple sequential occurrences of the same symbol, which can be encoded well using the RLE algorithm.

Besides the trace shown in Fig. 18, a second trace with user activity as well as a data set taken while the user was asleep have been used. Each of the three traces that have been used contained 30,316 acceleration readings for each dimension, equalling a number of 90,948 individual sensor readings. In order to visualize the relation between data variation and energy gain, we have segmented the traces into 26 chunks of 1,166 readings per axis each, corresponding to 166 transmitted packets per chunk. For each of the chunks, the average

Figure 17: The Porcupine platform (Source: <http://www.ess.tu-darmstadt.de/projects/porcupine>)

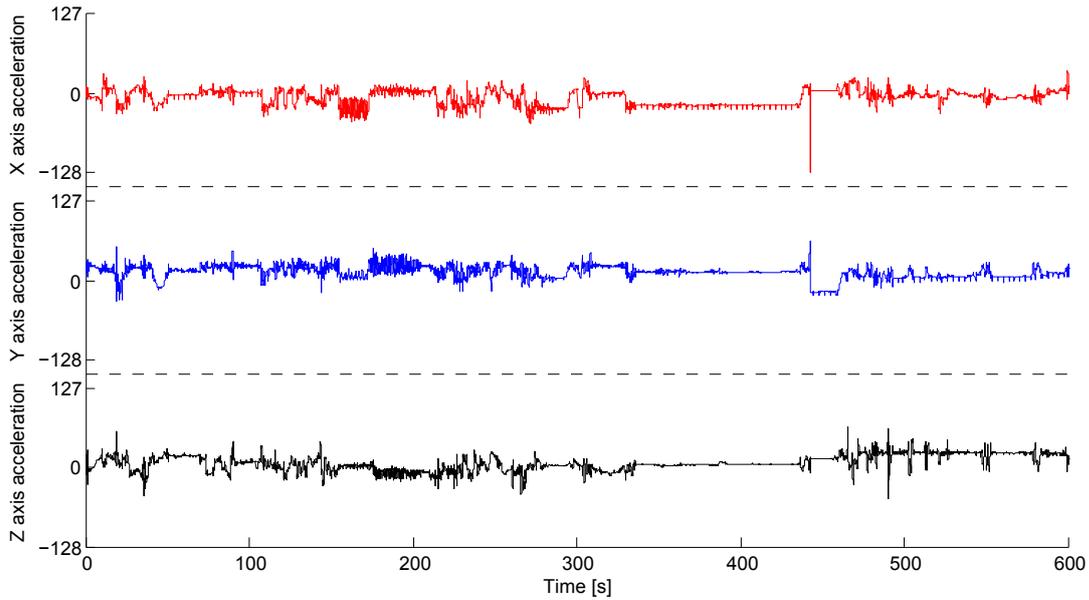


Figure 18: Exemplary Porcupine accelerometer data trace

energy demand has been calculated individually. The actual acceleration values are plotted in the top of each figure to allow for an easy comparison between the data characteristics and the corresponding energy gains.

In a first simulation, we have analyzed the energy gains for each of the chunks in the data trace presented above, for which the results are shown in Fig. 19. A clear correlation between the activity of the user and the data compressibility can be observed. During the phase of user activity (i.e. before $t=330s$ and after $t=450s$ on the time axis), only slight energy savings can be achieved; the compression of some chunks even leads to energy losses. In contrast, during the user’s inactivity period between the two activities, measurable energy gains of up to 5.9% are observed. Over the total trace duration, an average energy gain of 1.1% is achieved, which

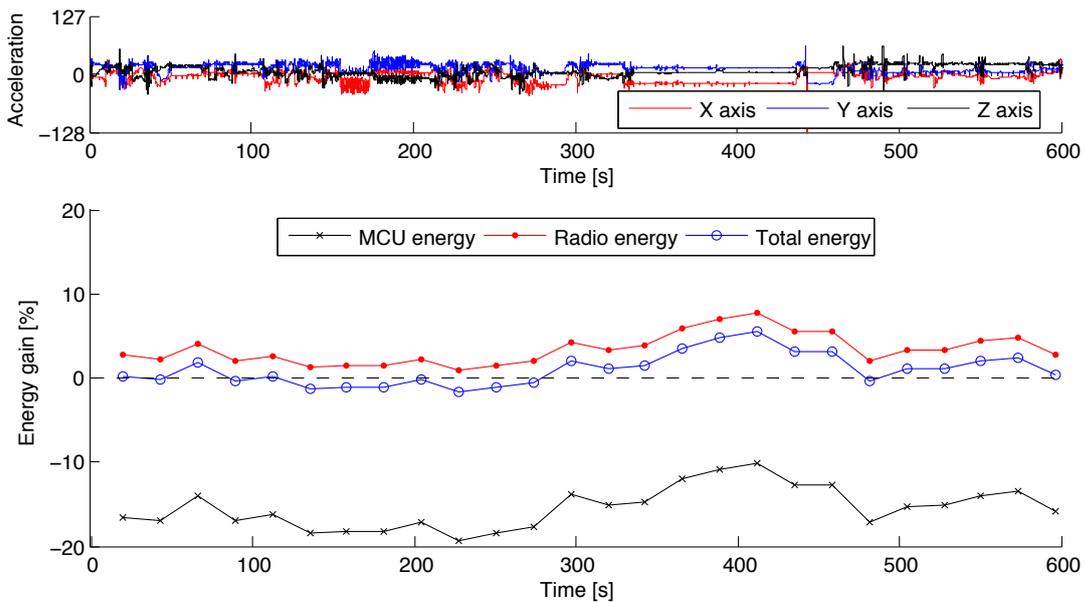


Figure 19: Results of the energy simulation of the first activity period trace

is composed from the weighted sums of 3.3% radio energy savings and a demand for 14.1% additional MCU energy.

The result for the second trace captured during the user's activity are presented in Fig. 20. Although the trace has also been captured while the user has been active, the characteristics of the trace are different from the previous data set. In contrast to the first data set, the trace shows stronger excitations during the user's activity phases, while less variations are present in the data between activities. The results shown in the figure reflect these characteristics, and show total energy gains ranging from -1.6% to 5.5%, with an average energy gain of 1.0% over the whole data set.

Finally, the third data set (shown in Fig. 21) represents readings from a phase when the wearer was asleep. The acceleration readings thus show a great number of consecutive identical symbols, which represents an ideal basis for the application of RLE. It becomes obvious from the figure that positive radio energy gains of 8.3% on average are encountered during the entire time period. Although an additional 9.1% of MCU energy is required, an average overall energy gain of 5.9% has been determined for the full duration of the trace.

5.1.4 Summary

In this feasibility study, we have analyzed the achievable energy gains when the RLE algorithm is applied on WSN nodes. Through the use of pre-defined data sequences, we have assessed best and worst case energy gains. A packet length of 21 bytes has been used in our simulation study to allow for the direct comparison between the Porcupine data sets and the synthetic data. Our simulations have shown that the TelosB platform can save up to 10.7% of energy when a payload of 21 identical symbols is given. This perfectly compressible payload is reduced to only three bytes by the RLE algorithm, representing an output ratio of $1/7$ of the original payload size. Even when taking the status flag into consideration, which indicates whether the payload is compressed, the compression gain is in excess of 80%. In contrast, for the case of incompressible payloads, platform energy losses of at most 3.3% have been determined.

Based on the observation that energy savings are possible when synthetic data traces are used, we have used three representative data traces collected by the Porcupine platform worn on a person's wrist. For a data trace collected while the user was asleep, 5.9% platform energy

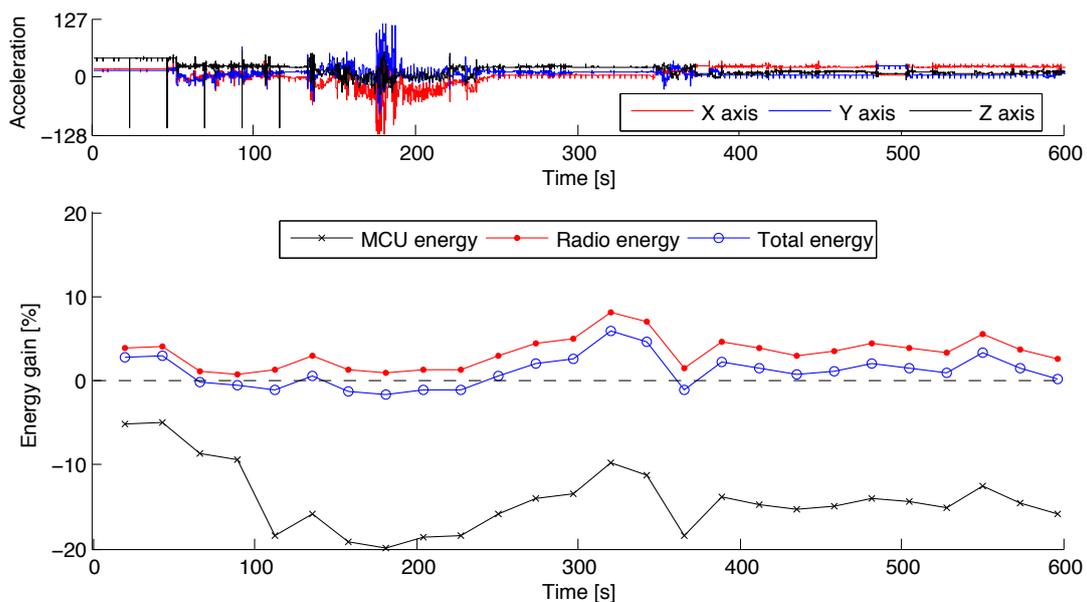


Figure 20: Results of the energy simulation of the second activity period trace

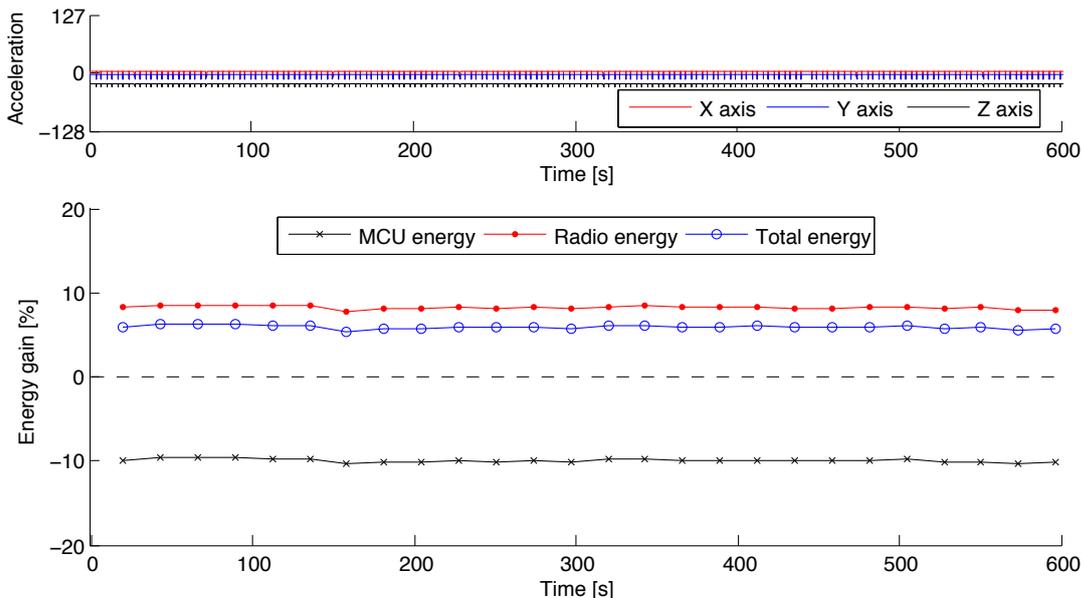


Figure 21: Results of the energy simulation of the sleep period trace

could be saved on average when applying the RLE algorithm. Both data traces collected during the user's activity have resulted in smaller energy savings of around 1.1% only, yet no overall energy losses have been encountered in any of the three simulations. It has thus been shown that the reduced transmission durations of compressed packets have effectively preserved more energy than required by the MCU to perform the RLE algorithm. In summary, the general feasibility of data compression in WSNs in order to attain energy savings has been shown. The computational demand of data compression algorithms must however be carefully analyzed to ensure that data compression actually leads to energy savings.

5.2 THE SQUEEZE.KOM COMPRESSION LAYER

For our demonstration of the feasibility of data compression in WSNs, we have used the simple RLE algorithm due to its low implementation complexity and resource demand. Although the application of the RLE algorithm has shown measurable energy savings in the simulated best case, only small overall energy savings were achieved in the average case because the specific properties of sensor network data streams have not been taken into account. We address this shortcoming by presenting the Squeeze.KOM data compression layer in this section [RHS09]. It specifically exploits the fact that data streams in sensor and actuator networks often show similarities between successive packets. We begin with an analysis of typical data characteristics and the compressibility of data packets in sensor networks. Based on our observations, we motivate our design decision to efficiently encode differences between successive packets instead of applying data compression on a per-packet basis. The proposed Squeeze.KOM layer integrates with the mote's operating system and processes all packets prior to their wireless transmission. Squeeze.KOM maintains a history of previously transmitted packets, against which outgoing packets are compared. When an element with sufficient resemblance has been found in the history, the difference between the two elements is compressed prior to its transmission. In the remainder of this section, we analyze the compressibility of sensor network data streams, followed by the presentation of the Squeeze.KOM compression layer. Finally, we assess its applicability in WSNs by conducting an analysis of its resource consumption and energy demand.

5.2.1 Compressibility of Sensor Network Data Streams

A wide range of WSN application areas have been identified by Estrin et al. in [EGH00] and [EGPS01], e.g. environmental and physiological monitoring, precision agriculture, smart spaces, or inventory tracking. In most scenarios, sensor readings are transferred to an external sink node, where they are centrally collected and analyzed. Besides the collection of data at a sink node, networks can also operate in ad hoc fashion by exchanging sensor readings between neighboring nodes and performing data processing on a local scale. In both cases, sensor samples taken from the physical environment surrounding the node comprise the predominant content of wireless traffic. Although network management messages (e.g. for routing or topology control) are required for the proper operation of a WSN, a low network management overhead is a major prerequisite for the efficient use of the available energy budget. As a result, we disregard management messages and only focus on the characteristics of packets containing actual sensor data in the following analysis.

In order to assess the properties of real sensor network traffic and confirm our assumption that many successive packets bear temporal resemblance, we have originally intended to analyze the data sets of representative WSN deployments. However, during our search for available traces, we have determined that the major hindrance to the use of sensor network data sets is their unavailability. To substantiate this claim, we have analyzed the project web pages for all environmental monitoring deployments listed in [HMo6], disregarding the projects for which the websites were no longer reachable. In our analysis, we have determined that the data sets for only two deployments were readily available for download from the project web page. A further two projects provided query masks to retrieve data, but the returned data set was always empty. In fifteen deployments, screenshots of the data traces could be downloaded, however of varying quality and often only for limited time periods. On three out of the fifteen project websites, a registration form was provided in order to attain access to data. Finally, eight projects did not provide any collected data, neither in visualized nor in raw form. As a result, we have only taken a closer look at the data sets for which at least information about the structure was available, and present the determined characteristics in the following list:

- ▷ **GREAT DUCK ISLAND** In the Great Duck Island project [MCP⁺02], the natural habitat of snow petrels has been monitored by deploying sensors deployed inside their burrows. Data packets were collected every five minutes and composed of a 25 byte payload containing temperature, humidity, barometric pressure, and light level readings.
- ▷ **GLACSWEB** In Glacsweb [MOHo4], probes with temperature, pressure, orientation, external conductivity, and strain gauge sensors were deployed into a glacier. The data packets used to transmit the sensor readings to the base station were 52 bytes in size, and transmitted six times per day.
- ▷ **PERMASENSE** The PermaSense deployment in the Swiss alps [BGH⁺09] has been installed in order to analyze the environmental conditions in permafrost regions. The deployment uses packets of 30 bytes size, which are transmitted every 30 minutes. The packets are composed of a sequence number, multiple timestamps, and readings from analog sensors that capture various sensing modalities, such as cleft dilatation, water pressure, or ice strain.
- ▷ **ZEBRANET** The social behavior of zebras was tracked in ZebraNet [JOW⁺02], where the GPS coordinates of the animals' current locations as well as a brightness reading (indicating whether the animal was located in the sun or the shade) were transferred to a base station in packets of 64 bytes each once every two hours.
- ▷ **SATIRE** The SATIRE body area network targets to trace human activity patterns and keep track of the user's current location [GJASo6]. In order to cover rapid movements, it collects and transmits sensor readings at a rate of 55 packets of 68 bytes payload size each second.

Table 10: Maximum achievable compression ratio for PermaSense data

COMPRESSION ALGORITHM	OUTPUT SIZE	AVERAGE PACKET SIZE	COMPRESSION RATIO
Uncompressed	591,900 bytes	30 bytes	1.0
GZIP	263,746 bytes	13.37 bytes	0.45
BZIP2	297,340 bytes	15.07 bytes	0.50
7-Zip LZMA	149,078 bytes	7.55 bytes	0.25
7-Zip PPMd	284,232 bytes	14.40 bytes	0.48

These observations also confirm the expectation that due to the inherent packet size limitation of the prevalently used IEEE 802.15.4 standard, packets of less than 127 bytes size are commonly used in order to avoid the need for packet fragmentation. An additional observation from all presented projects has been the fact that — as already outlined in Sec. 3.1 — statically defined packet payload structures are being used for the transmission of sensor data. Having observed that WSN packets are commonly only tens of bytes in size, we analyze their compressibility in two regards:

1. We establish the compression gain when a full data set is compiled into a single file and completely available to the compressor at the time of encoding. We compare the size reductions achievable when standard desktop computer compression tools are used.
2. We analyze the compressibility of individual sensor data packets. In order to eliminate the impact of headers or metadata appended by standard compression tools, we rely on specific implementations of the underlying mechanisms.

We have conducted our analysis using data traces from the PermaSense [BGH⁺09] environmental monitoring project, because PermaSense is amongst the few projects that make their data available for download from the project website. Furthermore, the largest coherent data set could be downloaded from the PermaSense website. In the following evaluation, we have used the data captured by node 2036 from 15 November through 15 December 2008, a data set composed of 19,730 packets. In a first step, we have conducted an analysis of the overall compressibility of collected sensor data using standard desktop computer compression applications. In our analysis, we have used the GZIP [Deu96], bzip2 [Sew11], and 7-Zip [Pav11] (using both the PPM and the LZMA compression algorithms) file archivers. The compression of the complete data sets has been performed on a desktop computer, because the algorithms' demands for computational power and memory exceed the capabilities of current mote platforms. The results are shown in Table 10, and indicate that size savings of up to 75% can be achieved when compressing the data in its entirety using adaptive statistical compression techniques.

In a second step of our analysis, we have compressed the packets individually by a number of different combinations of the preprocessing, compression, and entropy coding algorithms, according to the common sequence of data processing steps shown in Fig. 22. We have analyzed the use of the Move-to-Front [BSTW86] and Burrows-Wheeler [BW94] transforms for preprocessing. While the former solution maintains a list of frequently occurring symbols and allocates lower values to symbols that appear more frequently, the BWT sorts all rotated versions of the input string and outputs the last symbols of each of the entries in the sorted list. Both are widely used as preprocessing steps because they often increase the redundancy of the

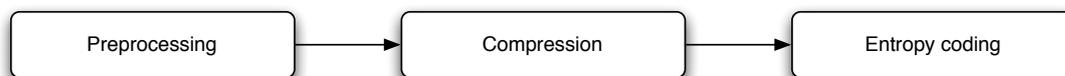


Figure 22: Flow of processing steps in typical data compression algorithms

Table 11: Output size and compression ratio of PermaSense data

OPERATION SEQUENCE	OUTPUT SIZE OF ENTROPY ENCODER		
	NONE	ADAPTIVE HUFFMAN	RANGE CODING
<i>none</i>	30 bytes (1.0)	47.4 bytes (1.58)	28.7 bytes (0.96)
MTF	30 bytes (1.0)	51.4 bytes (1.71)	28.9 bytes (0.96)
BWT	34 bytes (1.13)	50.6 bytes (1.69)	31.9 bytes (1.06)
RLE	31 bytes (1.03)	47.9 bytes (1.6)	29.5 bytes (0.98)
LZ77	49.4 bytes (1.65)	60.6 bytes (2.02)	42.2 bytes (1.4)
LZW	32.6 bytes (1.09)	31.8 bytes (1.06)	31.8 bytes (1.06)
MTF → RLE	30 bytes (1.0)	52.2 bytes (1.74)	29 bytes (0.97)
BWT → RLE	37.5 bytes (1.25)	52.1 bytes (1.74)	34.5 bytes (1.15)
MTF → LZ77	49.4 bytes (1.65)	60.6 bytes (2.02)	42.2 bytes (1.4)
BWT → LZ77	49.4 bytes (1.65)	60.6 bytes (2.02)	42.2 bytes (1.4)
MTF → LZW	32.6 bytes (1.09)	58.6 bytes (1.95)	31.8 bytes (1.06)
BWT → LZW	32.6 bytes (1.09)	58.6 bytes (1.95)	31.8 bytes (1.06)

sequence and thus increase its compressibility. Furthermore, we have regarded RLE [Gol66], LZ77 [ZL77], and LZ78 [ZL78] for the actual compression step. While RLE replaces multiple occurrences of the same symbol by a repetition count field, both Lempel-Ziv approaches are based on the use of dictionaries and replace recurring phrases in the input sequence by pointers to the dictionary entry. In the final entropy coding step, both adaptive Huffman coding [Vit87] and range coding [Mar79] have been used in our analysis.

The average resulting output sizes are shown in Table 11. It is obvious from the results that only some sequences employing range coding provide results that are slightly smaller than the corresponding input data. However, in the worst case, the output was twice as large as the input, clearly disqualifying these compression mechanisms on a per-packet basis for sensor network traffic. Packet compression can assist in maximizing the lifetime of WSNs by reducing the energy consumed by radio transmissions. Our analysis of the compressibility of sensor data has however made clear that compression algorithms operating on individual packets cannot achieve high compression gains due to the limited correlation between its contents. While we have shown that the correlation of data within one packet is often too small to effectively apply data compression, we exploit the characteristic feature that packets sent in a sequence or stream often expose a high similarity of their contents.

5.2.2 Compression of Packet Differences

We have shown that full knowledge of the data set allows for compression gains of up to 75%, while our compressibility analysis of individual packets has mostly resulted in size increases. Considering that an additional energy expenditure is required for the processing step, the node's total energy demand increases in all cases where packets are compressed independently of each other. In contrast, the collection of large data sets on a local scale has been shown to provide a good basis for high compression gains. However, the collection and handling of large data sets on embedded sensing devices has several major drawbacks:

- ▷ **LOSS OF TIMELINESS** In most sensor network deployments, data needs to be delivered to the base station in certain time intervals. Depending on the required reporting interval, the possible extent of local aggregation might be confined to a small number of sensor readings only, resulting in small compression gains.

- ▷ **ALGORITHMIC COMPLEXITY** The tight resource constraints of current motes mostly disallow the execution of complex compression algorithms. For example, the 7-Zip LZMA algorithm used above significantly exceeds the amount of RAM available on current motes, and their application would require excessive computation time.
- ▷ **FRAGMENTATION** Once the compressed packet exceeds the maximum frame size of the underlying communication standard, packet fragmentation must be applied. In order to recover from packet losses, all resulting fragments must be buffered even after their transmission in order to ensure that the compressed data can be successfully decoded at the receiver.
- ▷ **ENERGY** We have outlined in Sec. 2.2.3 that local data storage is a major consumer of electric energy. This additional energy demand for both read and write operations to the local data storage must thus be taken into consideration when large amounts of sensor data need to be stored prior to their compression.

Due to the continuous nature of many monitored physical phenomena and the prevalence of static payload structures, packets in sensor networks often bear high temporal resemblance to each other. Instead of considering packets in isolation, we thus deliberately exploit the correlations between packets in order to confine transmissions to the differences between consecutive packets only. We therefore store previously transmitted packets by creating a local history buffer with a limited number of entries. After the successful transmission of a packet, a copy of the packet can be stored in the local history for later reference. Outgoing packets can then be either transmitted as a whole, or the difference to an entry in the local history can be calculated and efficiently encoded prior to its transmission.

In the remainder of this section, we term packets that were completely transmitted as *index* packets, denoting their payload as I_n , where n is the number of the entry within the local history. A separate transmission history of index packets is maintained for each receiver node. It stores the index n , the corresponding payload I_n , and a hash value $\text{Hash}(I_n)$ of the data, which is eight bits in size. Furthermore, each entry is annotated by the time of its last usage, in order to enable Squeeze.KOM to replace the least recently used entry.

As opposed to index packets, which contain a full payload, packets that only contain the differences between a payload P and an index packet stored in the local history, are termed *differential* packets, and denoted as Δ_n , where n is the index of the referred index packet. The hash value of the referenced index packet is inserted in every differential packet in order to ensure that sender and receiver are synchronized and refer to the same index packet. In order to generate the differential of the index packet I_n and the payload of the next packet P , we compare the following three differentiating methods:

1. Byte-wise arithmetic subtraction ($P - I_n$)
2. Bit-wise XOR operation on the payloads ($P \oplus I_n$)
3. Preceding the XOR operation in the previous method by a conversion of the payloads to Gray Code [Gra53] (an encoding in which only a single bit changes for successive values), resulting in $\text{GC}(P) \oplus \text{GC}(I_n)$

In a first evaluation step, we assess the performance of the differentiating algorithms with regard to the properties of the resulting output sequence. The ideal output of any differentiation algorithm would be comprised of a long run of identical symbols, which can then be encoded efficiently. In our first analysis, we have set the size of the local history to one element only and regarded both the very first transmitted payload P_0 and the payload P_{i-1} which has been transmitted immediately before P_i . We have again used the same PermaSense data set that has already been discussed in the previous section. The results of our analysis are shown in Table 12, indicating the average and maximum bit run lengths.

For the difference between a packet P_i and the initial packet P_0 , all mechanisms increase the length of the longest run of '0' bits by a factor of at least 2.7. However, the mean lengths of bit

Table 14: Payload output sizes of the presented compression algorithms

INPUT	PAYLOAD OUTPUT SIZE AFTER COMPRESSION/CODING		
	RUN-LENGTH ENCODING	GOLOMB-RICE	BINARY DISTANCE
P_i	31.05 bytes (1.04)	75.72 bytes (2.52)	44.98 bytes (1.5)
$P_i - P_0$	29.49 bytes (0.98)	72.89 bytes (2.43)	43.05 bytes (1.44)
$P_i \oplus P_0$	29.45 bytes (0.98)	64.66 bytes (2.16)	42.18 bytes (1.41)
$GC(P_i) \oplus GC(P_0)$	29.82 bytes (0.99)	45.71 bytes (1.52)	42.10 bytes (1.4)
$P_i - P_{i-1}$	21.40 bytes (0.71)	34.40 bytes (1.15)	15.17 bytes (0.51)
$P_i \oplus P_{i-1}$	21.39 bytes (0.71)	20.71 bytes (0.69)	13.39 bytes (0.45)
$GC(P_i) \oplus GC(P_{i-1})$	21.38 bytes (0.71)	16.97 bytes (0.57)	12.98 bytes (0.43)

number of ‘0’ bits exceeds 71 and is therefore calculated as the sum of 71 and the value of the succeeding code.

The resulting output sizes of the data compression step are compared in Table 14. The compressibility of P_i itself shows that the compression of full index packets is not meaningful. However, when packet differences are used as the input to the compression stage, size reductions can be observed. While comparably small in the case when the reference to the initial packet P_0 is being made, size reductions of 57% are observed on average when the binary distance coding step is applied to the difference between two successive packets. Because the observed small benefit of applying Gray coding to the packets is faced by an increased decoding effort, we have disregarded its use in our further design. Instead, we rely on the sole use of the (computationally inexpensive) XOR operation in combination with binary distance coding for our further design of the Squeeze.KOM compression layer.

5.2.3 The Squeeze.KOM Compression Layer

We have observed that payload compression in sensor networks is only feasible when information about preceding packets is available. Due to the temporal continuity of many physical phenomena, such as temperature or humidity, the transmission of differential readings has been presented in many implementations like [SGO⁺04, MV08]. However, the drawback of exclusive transmissions of packet differences is the loss of all further data when packet losses occur on the wireless communication channel (loss propagation). Similarly, when two or more applications that use the same transmission intervals are executed on the same mote, the calculation of packet differences must be realized in each application individually. Otherwise, significant differences between two consecutive packets would be determined because the packets result from different applications with different packet contents.

Instead of imposing the burden of detecting and compressing similarities between outgoing packets on the application developer, we have decided to create a dedicated compression layer in the communication stack of the mote’s operating system. The resulting Squeeze.KOM implementation can be seamlessly integrated with existing node platforms and applications. By replicating the interfaces provided by the network layer, only small modifications to application code are required; existing applications can hence be adapted to make use of the new layer easily. An overview of the elements in the compression layer is depicted in Fig. 23. Squeeze.KOM operates on unidirectional streams of data and creates a packet history for each data stream, both in reception (RX) and transmission (TX) direction. As a result, two history instances are required for a bidirectional communication link, one catering for the correct decoding of received differential packets, and one for the calculation and transmission of packet differences.

Packets originating from the application are separated in header and payload fields and forwarded to the central component of the layer, the compression framework. The core

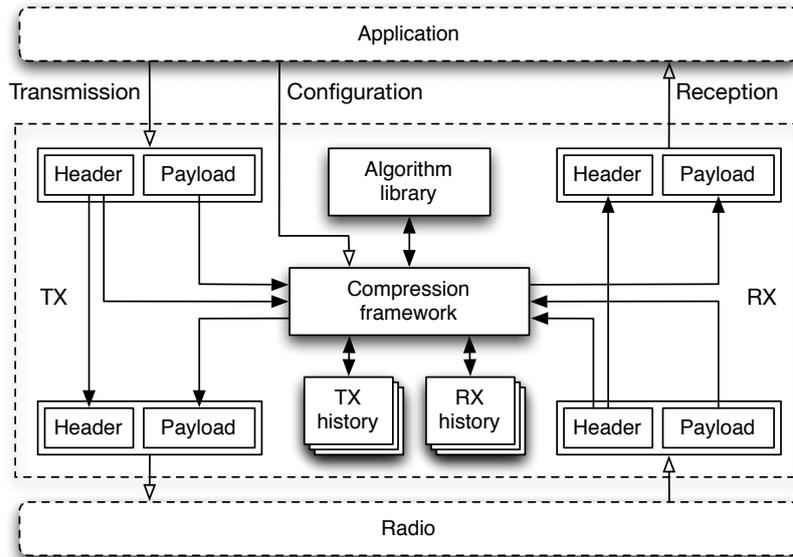


Figure 23: Internal structure of the Squeeze.KOM compression layer

functionality of the compression framework is the analysis of outbound packets for their compressibility with the help of packets stored in a local history. A comparison of the packet's payload to the set of previously sent packets stored in the local transmission history allows the framework to determine whether transmitting a differential packet is feasible. If this is the case, the payload is replaced by a reference to the most similar history element, its hash value, as well as the encoded difference between both sequences. After performing the selected compression algorithm on the data, a status field is added to the payload, notifying the receiver about the sequence of performed mechanisms. The receiver operates in reverse to the data encoding at the sender. Received packets are forwarded to the local compression framework, where compressed contents are first decompressed. When packet differences have been transferred only, the original sequence is restored by combining the received difference data with the local history, while full packets are directly forwarded to the application after inserting their payload into the reception history.

DATA ENCODING Once the application requests a packet transmission, Squeeze.KOM checks the corresponding payload P for similarities with the elements in its history table for the given destination node. This procedure is indicated in Fig. 24. A complete reference of all used abbreviations is shown in Appendix A.1. For each packet payload I_m stored in the history list,

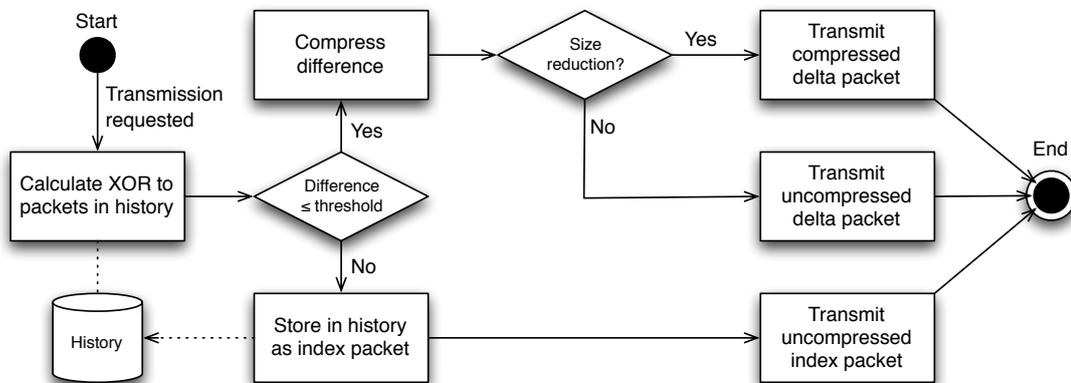


Figure 24: Data encoding flow of the Squeeze.KOM compression layer

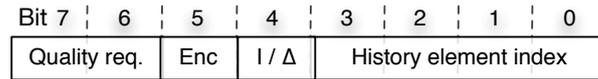


Figure 25: Bit structure of the Squeeze.KOM status field

the number of set bits b_m after the bitwise XOR operation between I_m and P is calculated. In order to maximize the compression gain, only the history entry I_n for which $b_n = \min(b_m)$ is regarded for all further operations.

Before the history entry is however used for the differentiation of the outbound packet, b_n is compared to a previously determined similarity threshold th (cf. Sec. 5.2.4 for an analysis of suitable values for th). Only when $b_n \leq th$, the previously calculated packet difference between I_n and P is used, and the resulting Δ_n is analyzed with regard to its compressibility. Should both packet payloads be identical, an empty Δ_n is transmitted along with the hash value of I_n . Otherwise, Δ_n is compressed using the binary distance coding algorithm presented above, and its output size compared to the uncompressed size. In case the compression of the differential packet results in a size reduction, the compressed difference $\Delta_{n,enc}$ is transmitted. If the application of binary distance coding does not reduce the packet size, Δ_n is sent uncompressed.

Should no entries with sufficiently high similarity be present in the node's history, a new entry is created in the history table, and assigned the next available index. In case all indices are already taken, the existing index element which has not been used for the longest time is replaced with the new index packet and the corresponding hash value is calculated. The packet is then transferred as a full index packet in order to ensure that the history lists of sender and receiver are in sync. Based on our above analysis, which has shown that the compression of individual packets does not lead to significant energy savings, index packets are always transferred uncompressed.

DATA TRANSMISSION Squeeze.KOM precedes outgoing packets by a status field in order to transfer required information to the receiver. The status field is shown in Fig. 25 and composed of a quality requirement field (bits 7 and 6) that indicates the reliability requirement for the transmission. The quality indicator is followed by two binary flags that indicate whether the payload is compressed (bit 5) and if it represents an index or a differential packet (bit 4). Finally, the index n of the element in the history table is stored in bits 3 to 0. The status field is limited to a single byte, such that only a minimum of additional overhead is introduced in cases of incompressible payload data, while the four bit wide history element index allows for referencing up to 16 history elements.

On radio links suffering from packet loss, transferring differential messages can also lead to situations where a differential packet is received, but the referenced index packet is not present (error propagation). Due to the fact that such packets cannot be decoded correctly, they are normally silently discarded. Squeeze.KOM however offers an additional interface for the configuration of quality requirements. Through the interface, the stack's compression feature can be deactivated to eliminate the risk of hash value mismatches and error propagation. Similarly, quality requirements can be defined, e.g. to retransmit packets several times until an acknowledgement is received. Finally, the configuration interface can also be used to limit the memory consumption of the compression layer by reducing the size of the history lists.

DATA DECODING In order to restore the packet payload contents from Δ_n , a local copy of the referred index packet I_n is required. Receivers thus have to keep track of incoming index data in the form of a history list with information about the set of received indices n and the corresponding index packets I_n . The history list is thus identical to the table present at the sender side. The hash value does not need to be transferred with the index packet; instead, it is calculated locally on reception of an index packet. Only incoming differential packets Δ_n need to contain the hash value of I_n used at the sender side, such that the Squeeze.KOM

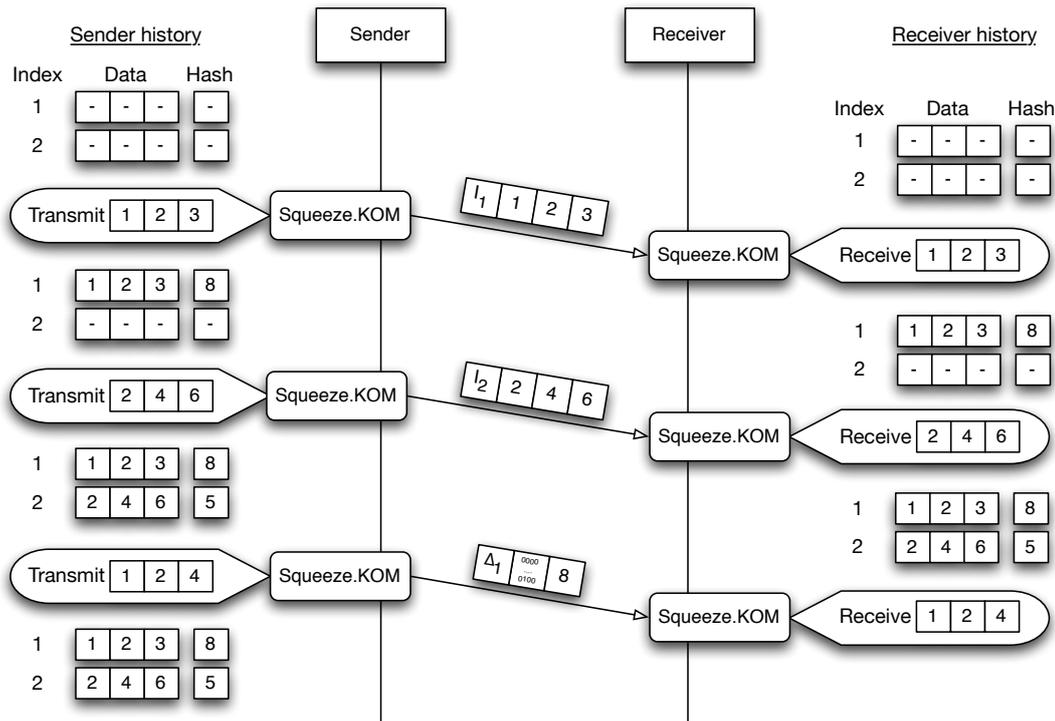


Figure 26: Exemplary communication sequence between two nodes using Squeeze.KOM

layer can ensure that the identical history element is referred to. This check is essential to detect lost index packets and make sure that nodes rejoining the network do not operate on outdated index data. In order to counteract mismatches between the hash values, which would normally result in the incoming packet being discarded, Squeeze.KOM can also be configured to request a retransmission of the packet as an index packet if its relevance is high.

Operation on the receiver side works in reverse to encoding the data at the sender. If the compression flag indicates that the data is encoded, decompression is performed in a first step. Index packets are added to the history table, possibly replacing existing elements, while differential packets are decoded using I_n from the local history. By performing an XOR operation on the data and referred index packets, the entire packet can be restored in a lossless way. By default, packets are transferred in a best-effort manner, where the compression layer is not required to provide any latency or delivery guarantees.

SYSTEM OPERATION An exemplary data flow between two nodes is depicted in Fig. 26, where a history size of two elements is used. In the figure, the sender first transmits two index packets I_1 and I_2 to the receiver node. Both are entered into the receiver's history upon their reception, along with their hash value, which is calculated locally at the receiver side. The third packet is sent as a differential packet, because the Squeeze.KOM layer has determined a similarity to history entry I_1 . As a result, a differential packet is compiled, comprised of the index of the history entry (1), the packet difference compressed using the bitwise XOR operation followed by the application of the binary distance coding algorithm, and the hash of the referred index packet (8). From the packet status field presented in Fig. 25, only the information whether an index or a differential packet has been transferred, and the history element index, are visualized. The impact of transmission errors can also be easily deduced from the figure. In case the first transmission is lost, the third packet cannot be properly decoded, and is discarded by the receiver's Squeeze.KOM instance. Similarly, if an updated index packet had been sent, but not received, the hash value mismatch would be detected by Squeeze.KOM, and the data packet discarded as a result.

Table 15: RAM demand of the Squeeze.KOM encoder for different history and packet sizes

PAYLOAD SIZE	HISTORY SIZE				
	1 ENTRY	2 ENTRIES	4 ENTRIES	8 ENTRIES	16 ENTRIES
16 bytes	40 bytes	62 bytes	106 bytes	194 bytes	370 bytes
32 bytes	88 bytes	126 bytes	202 bytes	354 bytes	658 bytes
48 bytes	136 bytes	190 bytes	298 bytes	514 bytes	946 bytes
64 bytes	184 bytes	254 bytes	394 bytes	674 bytes	1,234 bytes

Squeeze.KOM has been designed to realize an end-to-end compression, i.e. intermediate nodes do not need to decode the packets in order to forward them to their destination. As a result, energy is also preserved at intermediate nodes, which only need to forward smaller packets. The originating node’s address must however be retained throughout the packet transfer to indicate to the receiver which history list, i.e. which index packet set, to use. While the end-to-end compression leads to better energy efficiency at intermediate nodes, high channel loss rates may incur a large overhead for retransmissions of lost packets. In order to encounter such cases, the configuration interface can be used to deactivate the compression of important messages. Squeeze.KOM could however also be easily extended to use compressed data streams in a hop-by-hop manner or buffer messages on intermediate nodes until their receipt has been acknowledged when reliable transmissions are required.

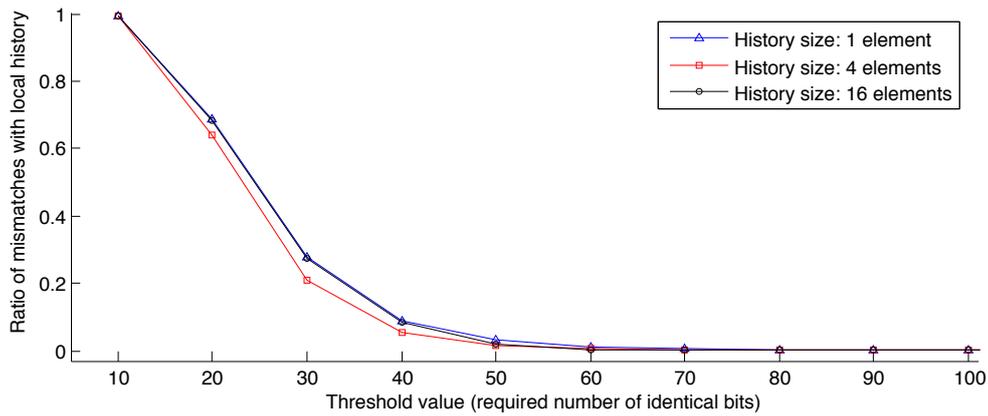
RESOURCE DEMAND We have implemented the presented Squeeze.KOM compression layer in the TinyOS operating system and evaluated its resource consumption on the TelosB platform. In order to allow for a fair comparison, we have also created a reference version of the implementation in which the complete Squeeze.KOM compression layer is deactivated. When comparing both to each other, an additional demand for 1644 bytes of program memory (equivalent to 3.4% of the TelosB platform’s total program memory) is required for Squeeze.KOM.

We have additionally determined the memory demand of Squeeze.KOM for different combinations of packet and history sizes in Table 15. The memory consumption of the compression layer is dominated by the number of index packets stored for outgoing and incoming connections. It can thus be reduced by limiting the number of entries for outgoing data, as well as by notifying neighbor nodes to reduce the number of entries allocated for their transmissions. While the demand for program memory does not change when multiple connections are being used, the amount of memory shown in the table is required for each connection.

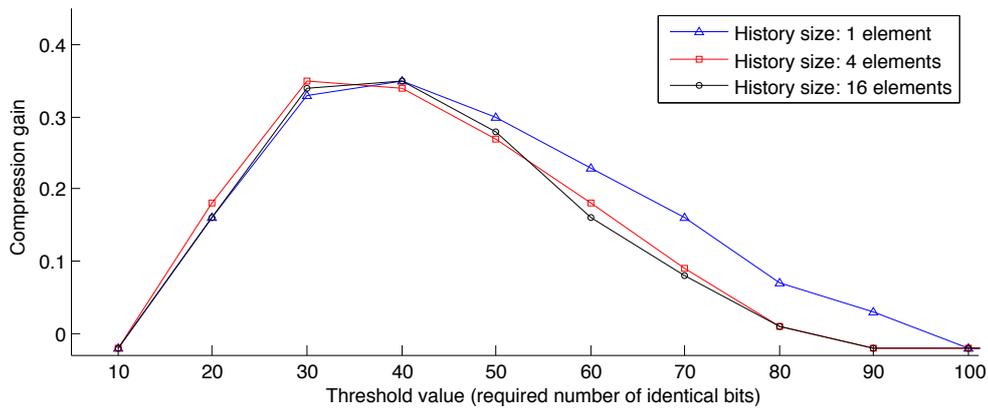
5.2.4 Evaluation

After having presented the design decisions and internal operation of Squeeze.KOM, we evaluate its performance in this section. In a first step, we analyze the identified tunable parameters and their impact on the overall performance of Squeeze.KOM. In the second step, the impact of packet losses on the rate of successful packet transmissions is analyzed, and a mechanism to recover from lost packets is presented. Finally, we verify the applicability of Squeeze.KOM on real hardware by presenting the results from a real-world experiment.

In all following experiments, we have set up a simple network topology with one sender and a single receiver node within communication range of each other. We have assumed an ideal communication channel without packet loss unless stated otherwise. In the evaluation, we have used all packets from the PermaSense data trace which has already been employed in the compressibility analyses. Based on our observations from the analysis of differentiation mechanisms in Sec. 5.2.2, we have selected the bitwise XOR operation to generate packet



(a) Relation between threshold value and ratio of history mismatches



(b) Relation between threshold value and compression

Figure 27: Analysis of the ratio of history mismatches and the compression gains for a single stream

differences between the outbound packet and the history elements. The compression step has been realized using binary distance coding. Only differential packets were compressed, while index packets have been transferred uncompressed. Instead of defining the values for history size and similarity threshold statically, we stepwise assess their impact on the compression gains.

IMPACT OF THE SIMILARITY THRESHOLD In order to generate a compressible differential packet, the payload to be compressed must exhibit sufficient similarity to an element stored in the local history. Because the XOR operation is part of the differentiation step and thus needs to be performed in any case, Squeeze.KOM uses the Hamming distance [Ham50] as its metric for similarity to the history elements, i.e. it counts the number of '1' bits in the result of the XOR operation. In order to cater for optimal compression results, it is necessary to define a maximum allowed number of differing bits as the threshold for similarity. Small threshold values hereby lead to highly compressible differential packets, as the resulting data features long runs of '0' bits and is thus well suited for the application of the binary distance code. However, choosing too small threshold values also results in a greater number of mismatches with the local history, and thus to an increased number of index packet transmissions. In contrast, high threshold values result in less compressible delta packets. The impact of the similarity threshold on the ratio of mismatches with the local history and the compression gain is shown in Fig. 27 for three different sizes of the local history.

The analysis of the impact of the threshold value on the mismatches with entries in the local history is thereby analyzed in Fig. 27a. The figure confirms that a small threshold values result in a high number of mismatches between the outbound packet P and the entries in the local

history. In contrast, a high value for the similarity threshold results in the smallest number of mismatches, and hence the smallest number of index packet transmissions. The Squeeze.KOM layer targets the reduction of a node's energy consumption by limiting the number of index packet transmissions, which always take place uncompressed, hence larger threshold values fulfill this requirement better.

The ratio of mismatches with the local history however cannot be used as the sole indicator for efficient operation. Although more packets are transmitted in compressed form when higher threshold values are used, the reduced compressibility of the packet difference needs to be taken into account. We hence analyze the overall compression gains, being an indicator for the achievable size reductions by applying Squeeze.KOM, for the transmission of the complete PermaSense trace in Fig. 27b. The plot confirms our assumption that high threshold values lead to degraded compression ratios of the resulting differential packet. Below the threshold value of 30 bits, the prevalence of uncompressed index packet transmissions compression gains negatively impacts the compression gain. Similarly, above a threshold level of 60 bits, more packets are transmitted in compressed form, however the compression gain is reduced due to the fact that more bits are set in the differential sequence, effectively reducing its compressibility.

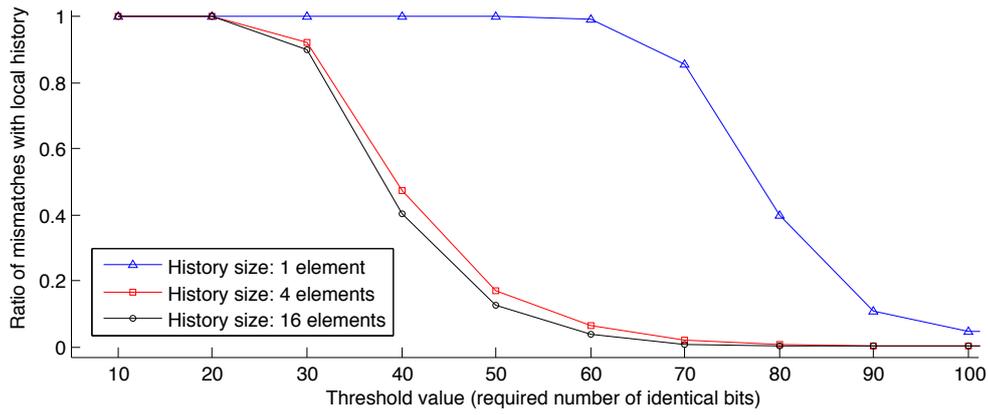
In summary, it is mandatory to find a suitable tradeoff between the threshold value and the maximum achievable compression gain. For the transport of a single stream of data with temporal correlations, the figures did not show significant discrepancies for the three history sizes compared. In the following simulations, we therefore assume a maximum allowed deviation of 40 bits for the given data set, for which an average number of history mismatches of 9% and a compression gain of 35% has been determined in Fig. 27.

IMPACT OF THE HISTORY SIZE While only a single stream has been present in the previous evaluation, the impact of the history size on the presence of multiple parallel streams between the same set of nodes is analyzed next. When multiple streams of data need to be transferred between two nodes, a correlation between these streams is not necessarily given. As a result, the use of a small number of history entries is expected to result in a large number of mismatches, and thus degraded compression ratios.

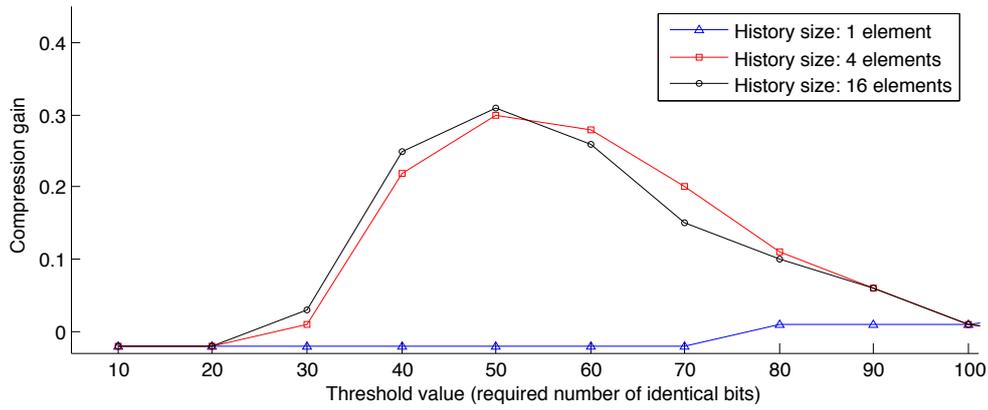
In order to analyze the impact of multiple streams, the compression layer has hence been tested with four different PermaSense streams being transferred in parallel. Besides using the data trace introduced above, we have downloaded three further traces from the PermaSense deployment during the same time period in November and December 2008. The ratio of history mismatches is shown in Fig. 28a, clearly confirming that a history size smaller than the number of distinct streams is insufficient to achieve gains by compression. Instead, the compression layer is bound to exclusively transferring index frames in this case, thus even increasing the packet size by one byte due to the status field. When the number of history entries is equal to the number of streams, the characteristics of the curve match the one in Fig. 27a well. The figure also shows that further provisioning the history with a larger number of entries than the number of streams only leads to slight improvements in the ratio of mismatches with the local history.

We have depicted the compression gain for the in Fig. 28b, proving that compression gains similar to the single stream case are possible when the history is dimensioned well. In case a threshold value of 50 bits and a history with 16 elements are assumed, our simulation shows a compression gain of 31%. It can also be noticed that, in both the one and the four stream cases, larger history sizes lead to slightly increased maximum compression gains. This originates from the fact that less index replacements take place when large history sizes are used, and thus old index entries remain longer in the history list.

IMPACT OF PACKET LOSS The presented single-hop scenario was modified to use a lossy channel, with packet success probabilities of $P_S = \{0.95, 0.9, 0.8, 0.6\}$. The application was configured to use a single PermaSense stream, a history size of one element and a similarity threshold of 40 bits. Because the loss of a single index packet already suffices to make the



(a) Relation between threshold value and ratio of history mismatches



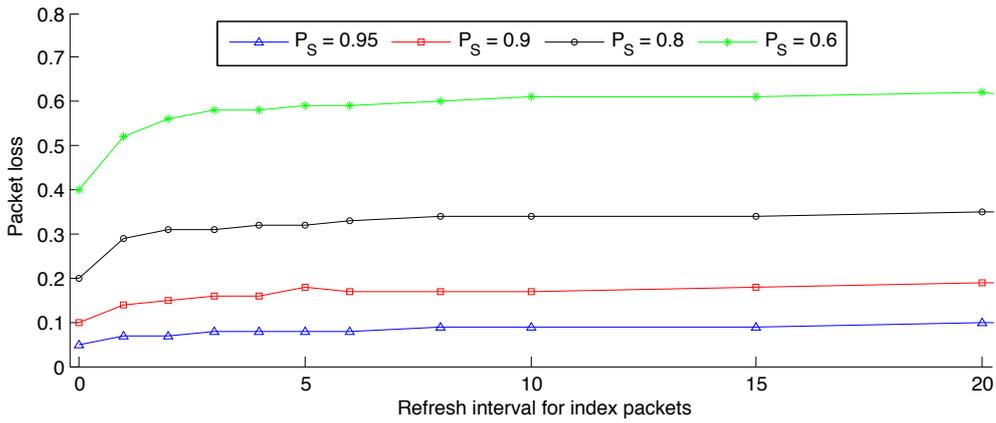
(b) Relation between threshold value and compression

Figure 28: Analysis of the ratio of history mismatches and the compression gains for four streams

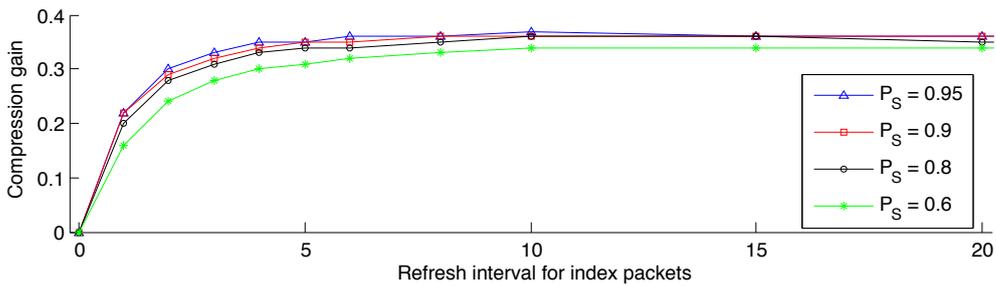
decompression of a complete stream of differential packets impossible, we have extended the sender node's implementation by an index refresh interval τ_r . When the number of sent delta packets referring to the same index packet exceeds the refresh interval, the data to be sent is instead transmitted as an index packet. Although this may be the case even in cases when the similarity requirement is fulfilled, the uncompressed transmission is important to keep the index packet list at the receiver in sync with the sender's list.

We have conducted an analysis of the impact of the refresh interval on the observed packet loss. Besides the inherent channel losses, the observed packet loss also includes packets that cannot be decompressed due to a lack of the referenced index packet (error propagation). The results of our analysis are shown in Fig. 29a, where packet loss rates identical to the channel conditions are observed when every transmission contains an index packet ($\tau_r = 0$). However, with increasing values of τ_r , the observed packet loss continues to increase because of the larger fraction of differential packets that cannot be decoded due to a lack of the index packet.

In a supplementary analysis, we have determined the compression gain for the same set of refresh intervals and packet success probabilities, which we present in Fig. 29b. Obviously, for smaller values of τ_r , i.e. in cases where index packets are sent more frequently, a smaller compression gain is given because of the prevalence of index packet transmissions. At the same time, however, higher success probabilities for the packet transmission are given. The compression gain is thus reduced in favor of an increased probability of receiving correct and complete data. In contrast, for a large refresh interval τ_r , the overall packet loss rate roughly increases by a factor of 50% as compared to uncompressed transmission.

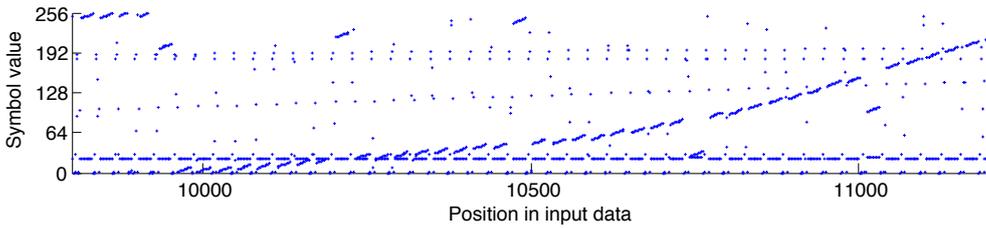


(a) Impact of the refresh interval on packet loss

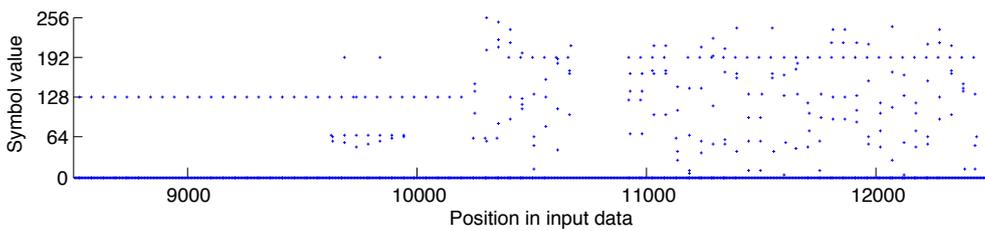


(b) Impact of the refresh interval on the compression gain

Figure 29: Impact of the refresh interval on packet loss and compression gain



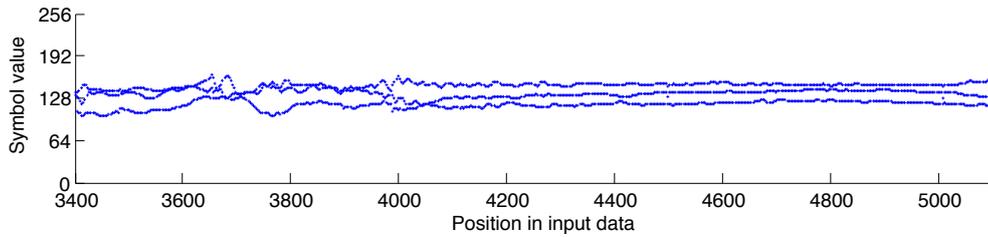
(a) Representative data trace excerpt from the PermaSense deployment



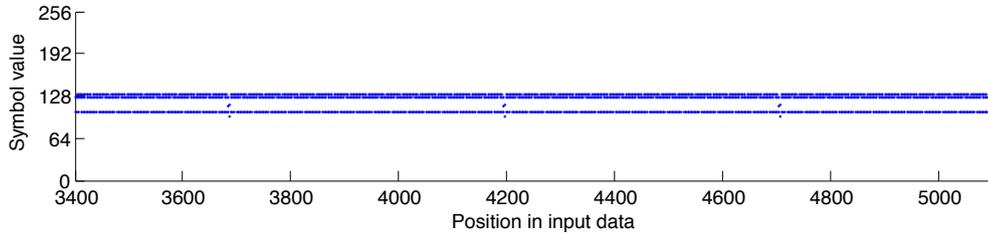
(b) Representative data trace excerpt from the Glacsweb deployment

Figure 30: Visualization of representative excerpts of the selected environmental monitoring data sets

COMPRESSIBILITY OF OTHER DATA TRACES While the evaluation with PermaSense data has shown compression gains of more than 30%, the generic applicability of Squeeze.KOM for other data sets cannot be directly concluded. We have hence considered a second data set from the field of environmental monitoring, and furthermore used two traces from the Porcupine body area sensor network in our evaluation. In order to highlight the different characteristics of these data sets, we briefly discuss their characteristics in the following list. A complete overview of the data traces is given Appendix A.2.



(a) Representative data trace excerpt from the Porcupine active phase



(b) Representative data trace excerpt from the Porcupine sleep phase

Figure 31: Visualization of representative excerpts of the Porcupine data sets

- ▷ **PERMASENSE** As discussed in Sec. 5.2.1, the PermaSense deployment in the Swiss alps [BGH⁺09] has been installed in order to analyze the environmental conditions in permafrost regions. In our analysis, PermaSense represents the first deployment from which we have collected sensor data sets¹. For our evaluation of data compression algorithms, we have used 19,730 packets of 30 byte payload each. The sensor data are composed of a sequence number, multiple timestamps, and readings from analog sensors that capture various readings, such as cleft dilatation, water pressure, or ice strain. A representative excerpt of the concatenated payload of all PermaSense packets is shown in Fig. 30a. The offset with regard to the start of the sequence is indicated on the abscissa, while the numeric value of the according symbol byte is shown on the y-axis.
- ▷ **GLACSWEB** The target of the Glacsweb project was the acquisition of insights about the interior of a glacier. Fifteen sensor nodes were thus deployed at the Briksdal glacier in Norway, and fitted with sensors for pressure, temperature, orientation, external conductivity, and strain gauge [MOH04]. We have gratefully received the traces of the Glacsweb project, and used all 523 available packets of 52 byte payload each in our analyses, from which we show a representative excerpt in Fig. 30b.
- ▷ **PORCUPINE** In case of the Porcupine [VGM06] project, we have selected two of the three traces that have already been used in our feasibility study in Sec. 5.1.3, where they were called the first activity phase and the sleep phase². As before, both sequences contain 4,406 packets of 21 acceleration readings each, which are present as one byte unsigned integer values. In order to allow for the simple comparison to the environmental monitoring traces, selected excerpts of both traces are visualized in Fig. 31.

In the evaluation, we assess the impact of the threshold value on the compression gain, targeting to confirm the high compression gains within our determined similarity threshold range. We determine achievable compression gain for history sizes of one, four, and sixteen elements, and present the results in Fig. 32. With compression gains of up to 74%, even at small threshold values, the Glacsweb data show excellent compressibility, which can be explained by the small deviations between successive packets. Similarly good results (64% compression gain) are observed for the Porcupines during the user’s sleep phase, although in this case, a history size of only one element is insufficient to achieve savings. Both the Porcupine

¹ Data collected in the PermaSense deployment can be viewed and downloaded at <http://data.permasense.ch>

² Data traces collected by Porcupine sensors are available for download at <http://www.ess.tu-darmstadt.de/datasets>

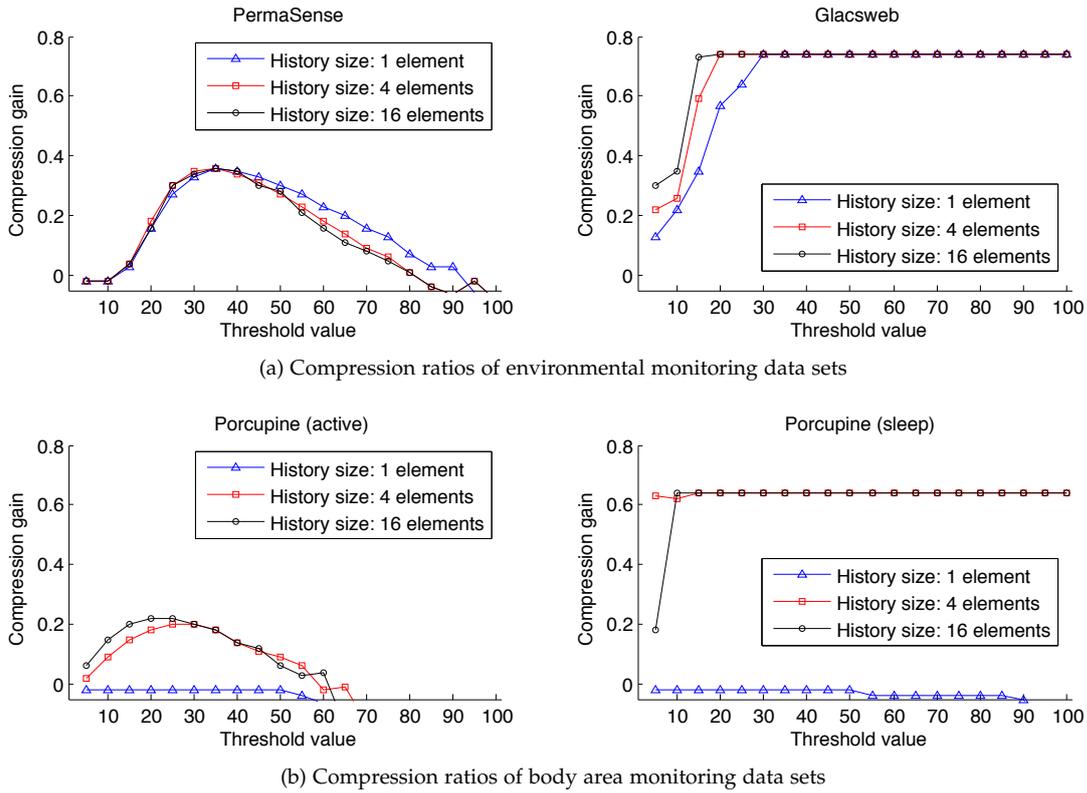


Figure 32: Achievable compression ratios for the selected data sets

deployment during user activity and the PermaSense data trace share the characteristic curve determined before, and show maximum compression gains of 36% and 22%, respectively. In summary, we have observed the highest compression gains for threshold values around 30 bits and a history size of four elements across all data traces.

ENERGY DEMAND The need for energy-efficient operation has been a result of our feasibility study presented above, i.e. less energy must be consumed by the compression operation than what is required by the radio to transmit the uncompressed data. We thus analyze the energy demand of Squeeze.KOM next, based on our TelosB implementation of Squeeze.KOM. As the decoding step is based on a single lookup in the local history, it requires less energy than the encoding of packets, during which the similarities to all history elements need to be determined. Based on this fact, we have only analyzed the energy expenditure of the transmitting node in the following evaluation. Analog to the setup used in the previous experiment, we have used the PermaSense data set, and set the similarity threshold to 30 bits as well as confining the history size to four elements. A packet transmission was triggered periodically at an interval of 250ms.

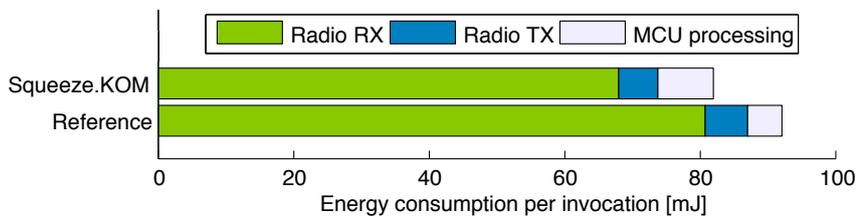


Figure 33: Energy consumption of the Squeeze.KOM compression layer

Table 16: Real-world evaluation of the compression layer

	SINK NODE 1	SINK NODE 2	SINK NODE 3	SINK NODE 4
Packet loss (uncompressed)	2.3%	6.4%	3.0%	4.3%
Packet loss (Squeeze.KOM)	2.7%	8.1%	4.7%	6.1%
Compression gain	12.1%	11.1%	12.0%	12.0%

Our simulations in the COOJA/MSPsim environment are visualized in Fig. 33 and show an increase of the processing energy demand by 57%. While the transmission of uncompressed packets requires 5.28mJ of energy, an additional 3.01mJ are required for the packet processing in Squeeze.KOM. At the same time, the transmissions of shorter packets lead to savings of 16%, i.e. the energy demand reduces from 80.88mJ to 67.94mJ per transmission. In summary, our evaluation has shown that a reduction of the total energy demand by 11.5% can be observed when Squeeze.KOM is applied to the PermaSense data set on real hardware.

REAL-WORLD EVALUATION In order to prove that Squeeze.KOM also operates under real-world conditions, we have ported to the compression layer to the SunSPOT [Sun08] platform. The SunSPOT has been particularly selected for our evaluation due to the fact that it is equipped with an acceleration sensor that has been used in our experiment. In the experiment, one mote has been configured to periodically collect sensor readings and broadcast that data to a set of four sink nodes. We have deliberately used multiple sinks in order to assess the impact of real channel conditions better, while identical packets are being transmitted to all destinations. The packet payload was comprised of a sequence number, a timestamp, and ten sensor readings (acceleration and tilt in three dimensions, temperature, light intensity, the state of the integrated momentary switches, and the current supply voltage level). The four sink nodes were located in an office environment and distributed around the sender, each at a distance of 5 meters. The sender was worn on the wrist of a person and configured to transmit 10,000 successive packets with sensor readings, sending each of them twice; once without compression to get an estimate on the real channel loss rate, and once using the Squeeze.KOM compression layer. The compression layer was configured to use a history size of one element, a threshold value of 40 bits and a refresh interval of 5 packets, based on our observations from the previous analyses.

The results for packet loss rates and the size reductions achieved by packet compression are compared in Table 16. The table confirms that the Squeeze.KOM compression layer matches the simulation results. Besides the expected slight increase in unrecoverable packets, resulting from outdated or missing index elements, the use of the compression layer leads to size reductions of around 12% even in our scenario where highly dynamic accelerometer data comprised a major part of the payload.

5.2.5 Summary

We have presented the lossless Squeeze.KOM payload compression layer, which seamlessly integrates with a mote’s operating system. The compression gain analysis of Squeeze.KOM has shown that data sets taken from real sensor network deployments can be reduced by up to 74% when compression is applied. We have discussed the tunable parameters, and determined the range of ideal settings for the given scenario. A history size equal to the number of streams transported at a time has shown to already result in good compression gains. A similarity threshold between 20 and 40 bits has resulted in the highest compressibility values for the resulting differential packets due to the selected binary distance code. Although missing index packets add to the impact of packet loss when using the compression layer, good compression ratios can already be achieved with refresh intervals of only five packets. This comparably small refresh interval hence suffices to effectively alleviate the problems of

error and loss propagation. Finally, the modular architecture allows additional preprocessing and compression algorithms to be easily integrated into the compression layer in order to adapt the layer to the packet structures used in the network.

Squeeze.KOM is also well suited to compress S-RPC messages, because the characteristic signature part of S-RPC headers (cf. Sec. 4.2) can be easily used to detect similarities to packets in the history. Instead of matching the complete outgoing packet to all history entries, a comparison of the contained data types and the name of the invoked method is generally sufficient to already exclude mismatching packets. An analysis of the achievable size reductions when S-RPC messages are being compressed is thus presented in Sec. 5.4.5. In summary, Squeeze.KOM supplements the creation of smart environments by catering for an efficient data transfer between motes and thus presents a solution to the requirement of energy-efficient operation of ubiquitous computing systems [Wei93].

5.3 ADAPTIVE HUFFMAN CODING WITH TRIMMED HUFFMAN TREES

We have shown in Sec. 5.2.1 that the application of Adaptive Huffman Coding (AHC) fails to reduce packet sizes when applied to individual packets. This size increase results from the fact that the AHC algorithm needs to establish a code tree prior to its efficient operation. If an empty Huffman tree is used for each individual outbound packet, each symbol needs to be transmitted in uncompressed form once in order to integrate it into the tree. In contrast, when a single Huffman tree is used persistently across packet transmissions, the initial construction overhead can be quickly compensated. We thus investigate the conditions under which the application of AHC in WSA-Ns results in energy savings in this section.

After briefly revisiting the foundations of Huffman coding, we study the feasibility of applying the AHC algorithm by determining the achievable compression gains when a persistent Huffman tree is used to compress packet payloads. Subsequently, we discuss the applicability of AHC on resource-constrained embedded devices, based on our observation that the AHC implementation presented in [GTHo8] requires more than half of a TelosB's RAM in order to maintain a single compressed unicast radio connection. We address this limitation by proposing a modification that relies on reducing the size of the code trees [RCH⁺10], which drastically alleviates these resource demands. This reduction of the resource demand however comes at the price of slightly degraded compression gains; we thus analyze the impact of the tree size on the compression ratios. In a complementary energy analysis, we additionally show that positive energy gains – and thus prolongations of the node's runtime – are only achieved when sufficiently small tree sizes are used. We finally prove the applicability and benefits of the proposed approach in a real-world experiment.

5.3.1 Huffman Coding Revisited

The basic idea behind Huffman coding is the maximization of a sequence's entropy [Sha48] by assigning codes to input symbols, with their length being reciprocal to their occurrence

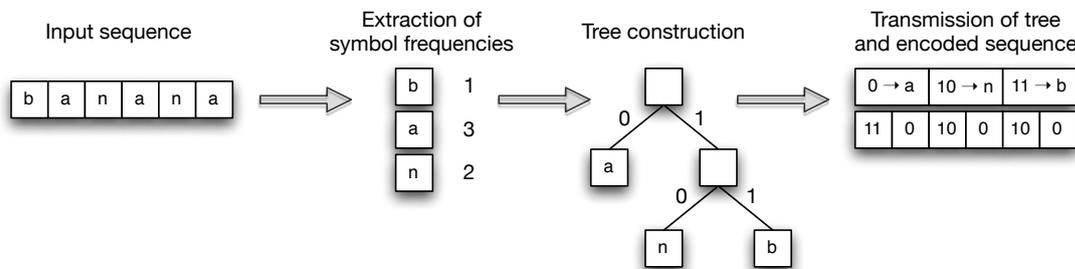


Figure 34: Visualized operation sequence of the static Huffman coding algorithm

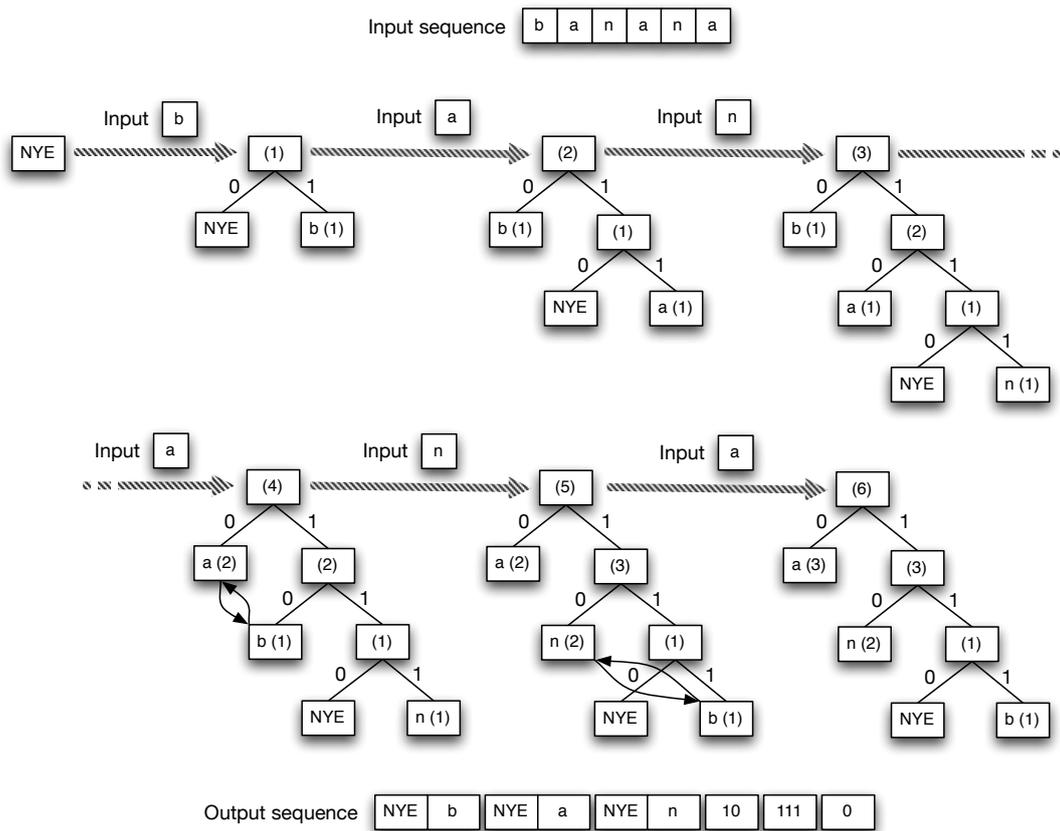


Figure 35: Tree construction in the adaptive Huffman coding algorithm

frequency within the input stream. In static Huffman coding [Huf52], the input sequence is analyzed prior to encoding, and occurrence frequencies of all contained symbols are determined. On completion of this process, a tree is constructed, containing mappings for all input symbols to their corresponding Huffman code. This tree must be sent to the receiver before the actual data is transmitted to ensure both parties operate on the same dictionary.

It is obvious that the complete sequence must be known a priori to determine the symbol distributions, and that the code tables must be exchanged before the data transfer takes place to ensure the receiver can successfully decode the packet. This represents additional overhead, which is however generally encountered by a near-optimal adaptation to the input sequence. The major drawback when using static Huffman coding is the required full knowledge of the data, which strongly limits its applicability in sensor networks, where sensor readings become available periodically. In such case, the algorithm needs to operate on individual packets, and thus transmit the code table in each of them. The operation of static Huffman coding is visualized in Fig. 34, where the input sequence “banana” is encoded. After the symbol occurrence frequencies have been determined, a tree is constructed such that shorter codes are assigned to more frequently occurring symbols, and vice versa. Once the tree is established, it is transferred to the recipient along with the encoded sequence.

In contrast to static Huffman coding, where the binary code tree is generated prior to the actual encoding step, AHC is based on creating the code tree (and possibly modifying it when occurrence frequencies change) in a dynamic way [Vit87]. To allow for these dynamic adaptations to occur, a dedicated placeholder symbol for an input symbol not yet encountered (NYE) is part of the code tree. This symbol is always maintained with an occurrence frequency of zero and thus always assigned one of the longest codes. Whenever a symbol not yet present in the Huffman tree needs to be transferred, the NYE symbol is transmitted, followed by the

unencoded representation of the symbol. The symbol is then added to the code tables of both parties, so its newly assigned code can be used on its next occurrence.

The encoding of the input sequence “banana” using AHC is shown in Fig. 35, where each node shows the assigned symbol as well as its occurrence frequency. The first symbol to be transmitted (‘b’) is not contained in the tree, so the NYE symbol and the unencoded representation of ‘b’ are transmitted to the receiver node. Subsequently, the NYE node creates two child nodes, with one being the new NYE node and the other one representing the input symbol ‘b’ with an occurrence frequency of one. This sequence is repeated two more times so that the symbols ‘a’ and ‘n’ are also added to the tree. On the second occurrence of the symbol ‘a’, the according code (‘10’) is transferred, and its occurrence counter field is incremented. A tree restructuring operation now swaps the nodes ‘a’ and ‘b’ in order to ensure that symbols with higher occurrence frequencies are always located closer to the root node, and thus allocated shorter codes. Similarly, ‘b’ and ‘n’ are swapped after the symbol for ‘n’ has been transmitted. In contrast to static Huffman coding, advance knowledge of the input sequence is not necessary when AHC is applied.

5.3.2 Compressibility of Sensor Readings

We assess the compressibility of the same data traces used in our evaluation of Squeeze.KOM (cf. Sec. 5.2.4) in order to prove the feasibility of applying AHC to sequences of multiple packets. In the analysis, we again base our observations on the complete data trace of PermaSense, Glacsweb, as well as both Porcupine traces from the activity and the sleep phases. Due to the fact that codewords in AHC are approximately proportional to the negative logarithm of the symbol occurrence probability, we analyze the symbol distributions of the data sets in Fig. 36 in order to attain an estimate for their compressibility using entropy coding.

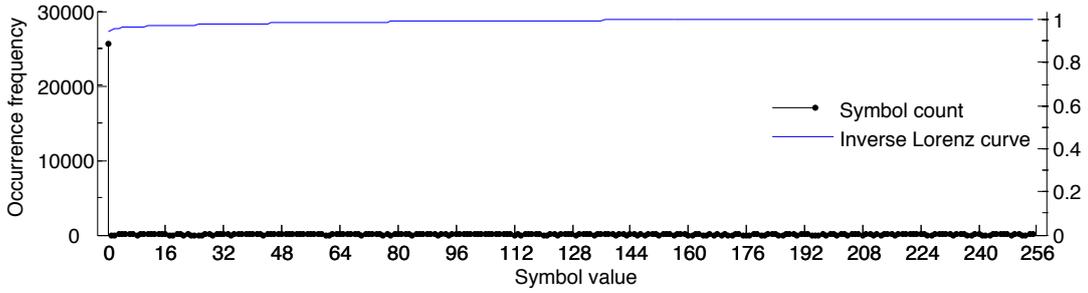
The figure shows that the occurrence frequencies of the used symbols are not distributed evenly across the data set. In contrast, only a few symbols with significantly greater occurrence numbers are present in all data sets. Targeting to quantify the deviation from an equal symbol distribution, we have also plotted the inverse Lorenz curves [Lor05] in the figure, with the according ordinate values shown on the right y-axis. The results verify that only few of the contained symbols occur with high frequencies, while the remaining symbols have almost negligible occurrence numbers. The discrepancy is strongest in the Glacsweb data set, where more than 98% of the input sequence are comprised of the same symbol. In contrast, the deviation of the inverse Lorenz curve from a straight line is smallest in case of the PermaSense data set. It should be noted at this point that only five different symbols are present in the entire data stream in the Porcupine sleep phase, whereas the active Porcupine data set is composed of 89 different values. Glacsweb makes use of 185 different symbols, and PermaSense spans the entire input symbol range of 256 values.

We next perform an analysis of the compressibility of the data sets when applying fixed code mappings on the sequence, especially considering the reduction of the tree size in order to cater for its applicability on resource-constrained systems. We reduce the size of the Huffman tree by statically designating codes to the frequent symbols only, while leaving the remaining symbols uncompressed. We thus estimate the compressibility of the data by evaluating the resulting output sizes when only a subset of symbols is being compressed while all remaining symbols are sent unencoded. Let us assume that a compression algorithm can encode n symbols of the size of a byte, leaving the remaining $256-n$ symbols uncompressed. We furthermore assume that γ_i represents the number of occurrences of the byte value i in the input sequence, and that $f(i)$ is the function that assigns a code length (in bits) to this symbol. In case of an uncompressed transmission, $f(i)$ would statically be assigned a value of eight bits. Given these definitions, the length l of the output sequence resulting from the

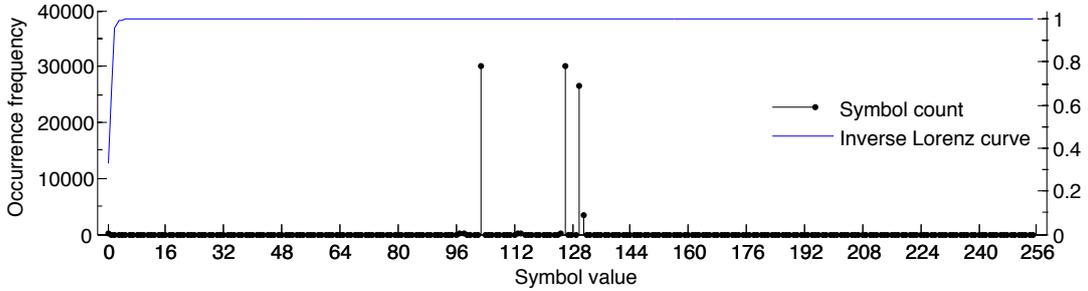
data compression step can be calculated as shown in Eq. 3, which sums the lengths of each symbol's code multiplied by the number of its occurrences within the input sequence.

$$l = \sum_{i=1}^{256} f(i) * \gamma_i \tag{3}$$

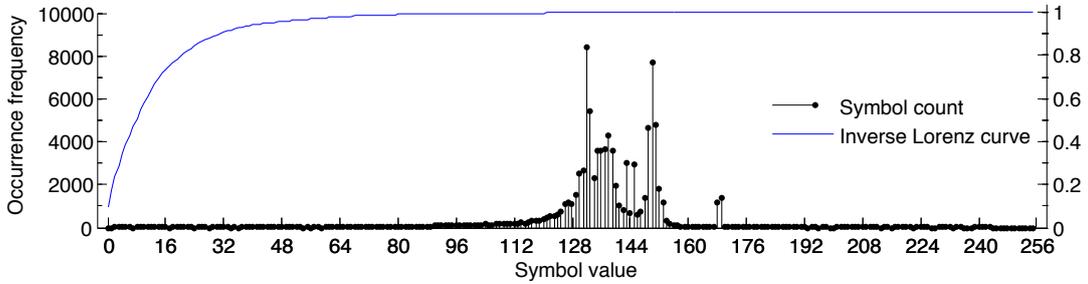
Symbol-oriented compression schemes, such as Huffman coding, create the code length function $f(i)$ from the state of their code table. In order to assess if compression with a reduced number of entries in the code tree is feasible, we have used two approximation functions for



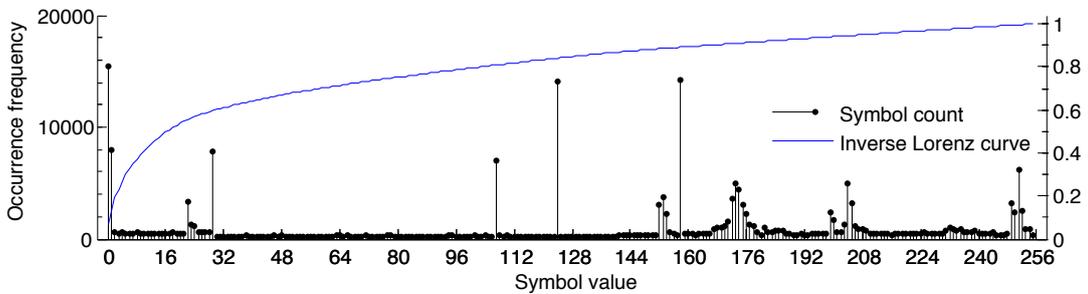
(a) Symbol distribution of the complete trace of the Glacswab deployment



(b) Symbol distribution of the Porcupine sleep phase data set



(c) Symbol distribution of the Porcupine activity phase data set



(d) Symbol distribution of the PermaSense data set

Figure 36: Symbol distributions and inverse Lorenz curves for the analyzed data traces

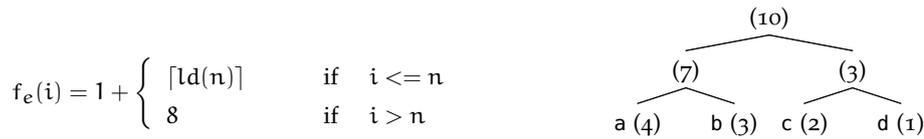


Figure 37: Definition of f_e and tree resulting from the input sequence aaaa bbb cc d for $n=4$

code lengths; while the f_e approximation defined in Fig. 37 assumes an equal code length for the symbols that are encoded, the f_f function introduced in Fig. 38 assigns the lengths of the output codes to follow the symbol’s rank $r(i)$ within the occurrence frequency list. Code trees for both functions indicating the symbols and their occurrence frequencies are also visualized in the figures for an exemplary input sequence of aaaa bbb cc d and $n=4$ entries in the tree. When only a subset of the possible input symbols is present within the table mapping from input symbol to corresponding code, an additional indicator is required to mark the following bits as plaintext or encoded symbol. We have selected a one bit prefix to allow for this distinction, which is also reflected in the two functions.

The results for our preliminary compressibility analysis are shown in Fig. 39. For reference, the compression gains when static Huffman coding is used have also been visualized in the figure in order to put the results into perspective. Although clearly indicating that savings can be achieved even when using the presented non-ideal code length distributions, the compression gain shows a strong dependence on the used data set. Both the Glacswab and Porcupine (sleep mode) data sets only expose a small number of symbols with high occurrence frequency, the f_f function presents a better basis to achieve high compression gains, as very short codes are assigned to the most frequently occurring symbols. This way, gains of 82% are achieved for Glacswab (at $n=1$), and up to 62% for the Porcupines (at $n=4$). In contrast, the active Porcupine and PermaSense data sets contain a larger number of frequent symbols, which are not covered well by the ranking performed in f_f . When applying f_e instead, compression gains of 17.3% (at $n=32$) for the active Porcupine phase, and 12% for PermaSense (at $n=16$) can be determined. By means of this feasibility study, we have determined that entropy coding with a code tree of limited size also leads to compression gains in excess of 10% for all analyzed data sets.

5.3.3 Tailoring Adaptive Huffman Coding to Sensor and Actuator Networks

We have analyzed the resource demand of the AHC implementation for WSNs presented in [GTHo8], which was kindly provided by its author Alexandre Guitton. In order to eliminate side effects and focus on the resource demand of the AHC algorithm, we have removed the additional features (e.g. fault tolerance) in the implementation. The simple compilation of the source code has shown a program memory demand of less than 3% of the available resources on a TelosB node, while 6,350 bytes of RAM (i.e. more than 62% on the TelosB) were required. The major part of the memory demand of the AHC implementation can be attributed to the Huffman tree.

In a Huffman tree, each node contains information about the symbol it represents, its occurrence frequency, its status (e.g., root node, leaf node, or NYE) as well as the identities

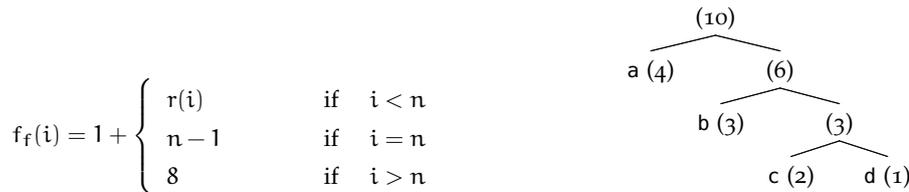
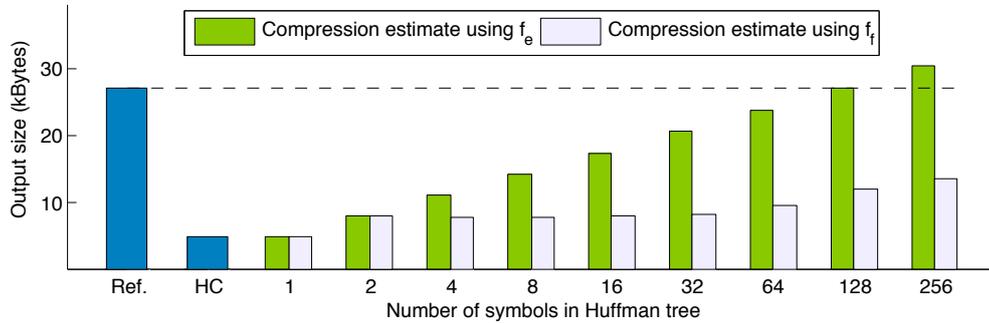
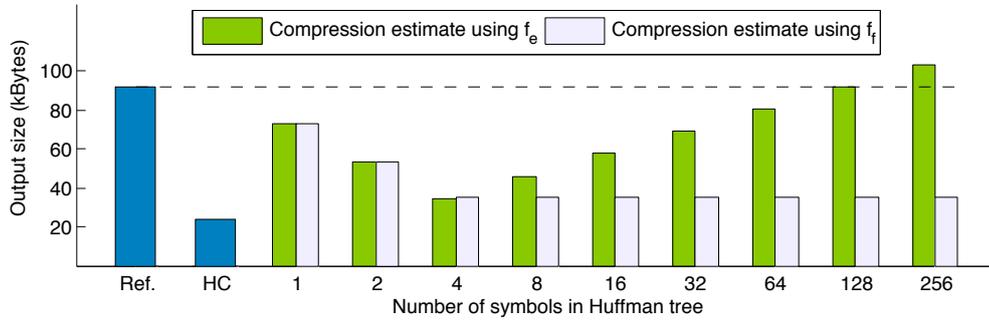


Figure 38: Definition of f_f and tree resulting from the input sequence aaaa bbb cc d for $n=4$

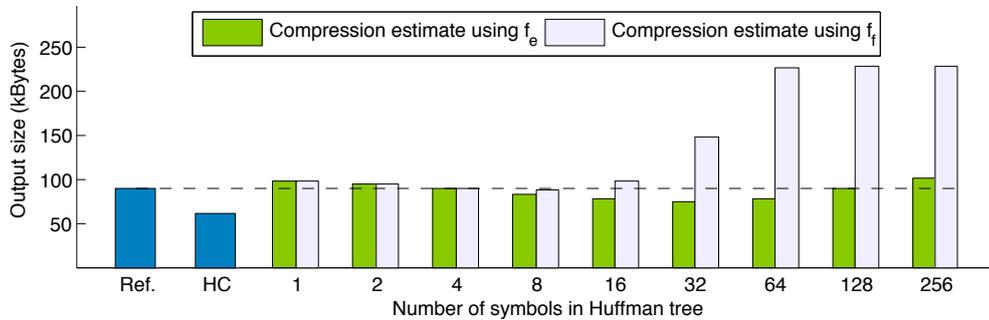
of its children nodes and its parent. In total, $2n - 1$ nodes are required to allow for n code entries in the tree, thus 511 nodes must be contained in the tree to allow for mappings of all 256 possible input symbols. The representation of this number requires nine bits, i.e. two bytes on any byte-aligned MCU. Because each tree node needs to store the identifiers of its parent and child identities as well as the input symbol it represents, its frequency and status



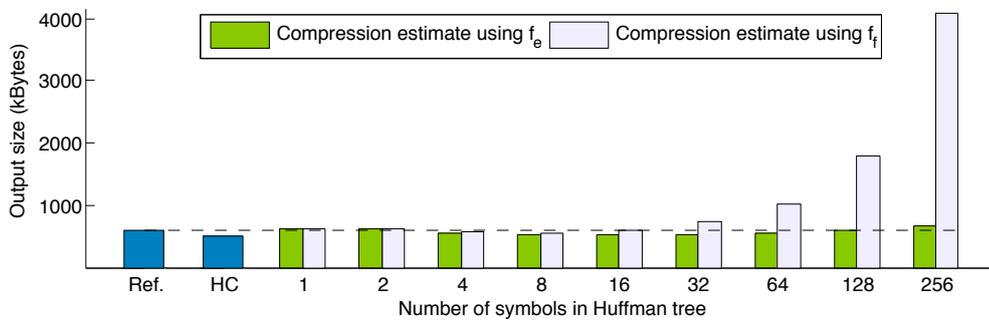
(a) Compression gain estimates for the complete trace of the Glacswab deployment



(b) Compression gain estimates for the Porcupine sleep phase data set



(c) Compression gain estimates for the Porcupine activity phase data set



(d) Compression gain estimates for the PermaSense data set

Figure 39: Size comparison of f_e , f_f , and static Huffman coding (HC) against the uncompressed reference

Table 17: Memory demand of the adaptive Huffman coder for different numbers of symbols in the tree

Symbols in tree	1	2	4	8	16	32	64	128	256
Nodes in tree	1	3	7	15	31	63	127	255	511
RAM (bytes)	1,036	1,052	1,084	1,148	1,276	1,532	2,044	3,068	5,116
% of TelosB RAM	10.2%	10.3%	10.6%	11.2%	12.5%	15.0%	20.0%	30.0%	50.0%

information, a minimum of eight bytes are consumed. This results in a demand of more than four kilobytes of RAM for a Huffman tree storing 256 symbols. Besides the tree itself, a table for the occurrence frequencies of all input symbols must be maintained, consuming another 256 bytes at least. This theoretical analysis also confirms the behavior observed in Guittón's implementation, where the memory consumption disallowed us to instantiate more than one connection at the same time. The lack of dynamic memory allocation schemes in the used TinyOS operating system [HSW⁺00] additionally constitutes a major drawback to the flexible application of AHC. When operating on statically assigned memory, worst case behavior needs to be assumed for the assignment of memory during compile time, i.e. memory needs to be reserved for all symbols, including those that never occur within the input sequence.

In addition to the demand for memory, we have also determined that a measurable computational demand is imposed on the node platform by the AHC implementation. Whenever a packet is sent or received, the Huffman tree is updated according to the new occurrence frequencies, and possibly needs to be reorganized by a number of swap operations. Furthermore, the translation between symbols and their code representations requires the traversal of the tree for both the encoding and decoding operations. Because the tree traversal operates on bit level, multiple traversal steps are generally required to convert a symbol to its code representation. The size of the underlying Huffman tree thus has a direct influence on the computational demand for the tree traversal and reorganization.

We have determined in the Sec. 5.3.2 that the symbol occurrence frequencies in current WSN deployments are often strongly biased towards a small subset of symbols, while the remaining input characters might only rarely or never be part of the input string. Our preliminary estimations of the achievable compression gains, depicted in Fig. 39, have shown that packet size reductions are also possible when less than 511 nodes, i.e. 256 symbols, are stored in the Huffman tree. Significant compression gains have been determined despite the need for an additional indicator bit, which precedes any encoded codeword in order to notify the receiver if the following bits represent a code from the Huffman tree or an unencoded 8-bit ASCII symbol. The selected estimation functions f_e and f_f were however neither adaptive to the traffic (i.e. a priori knowledge about the whole data set was required), nor did they match the characteristics of the traffic precisely.

In order to overcome this lack of adaptivity, we have decided to extend the adaptive Huffman coder by a parameter that allows us to confine the size of the Huffman tree at compile time. This decision has been based on our observation that the memory consumption of the Huffman tree is linearly dependent on the number of entries contained. Besides the fact that a significantly lower memory requirement is given when the tree is composed of fewer symbols, the size of their identifiers can also be reduced; an eight bit wide field is sufficient to store up to 255 nodes and thus 128 symbols in the tree. As a further benefit, the time to restructure the tree when changes in the occurrence frequencies are encountered also depends on the number of entries, and can in consequence be improved by reducing the tree size. The use of smaller Huffman trees also allows for the transport of multiple parallel data streams with individual code trees. In order to determine the resource overhead incurred by our AHC implementation, we have compared it to a reference application without compression functionality. Our comparison of the resource demands shows that the adaptive Huffman coder with reduced tree sizes only requires 1,150 bytes of program memory (equalling 2.3% of the available resources on a TelosB mote) more than the reference implementation. Because the

required amount of RAM depends on the size of the Huffman tree, we compare the additional memory demand for different tree sizes in Table 17. The numbers in the table confirm that the additional amount of memory required by our implementation stays within reasonable limits when only few symbols need to be stored in the tree.

TREE CONSTRUCTION AND SYMBOL REPLACEMENT The main difference between our proposed approach and conventional adaptive Huffman coding lies in the process of populating the tree. While in AHC, the NYE node is always present to attach unknown symbols to the tree, the limitation of the number of tree nodes in our algorithm can lead to situations where the NYE node, with its assumed occurrence frequency of zero, is being replaced by a symbol. We encounter this situation by keeping track of the occurrence frequencies of the symbols stored in the tree, and replacing the element with the smallest occurrence frequency in case a more frequent symbol is encountered.

We depict the operation of the proposed implementation in Fig. 40, where an input sequence of “aaaaaa bb c dd cc” and a tree capacity of 5 nodes (equalling 3 symbols) is assumed. The nodes in the tree are labeled with the symbols they represent as well as their occurrence counter. In the initial phase (Figs. 40a to 40c), updates to the code tree are performed identical to AHC, i.e. either the counter of a symbol present in the tree is incremented, or a new symbol is added to the tree through the NYE node. In Fig. 40d however, the new input symbol ‘c’ is encountered in the input sequence, while the limited number of nodes disallows the NYE to create a new tree node for the symbol. In contrast to the conventional AHC, where the NYE would create two further children nodes, our approach replaces the NYE by the symbol node; the tree thus loses the inherent capability of being extended through the NYE node. To still adapt to the input sequence during runtime, we follow the approach of replacing the node with the smallest counter value when a symbol with greater counter is present, such as shown in Figs. 40e and 40f. In order to allow for this, we keep track of all symbol occurrence frequencies during runtime in counter variables for each symbol, which are designed to be two bytes in size, i.e. a total RAM demand of 512 bytes is incurred. All resulting codes are prefixed by a single bit indicating if the following bit sequence should be interpreted as a code from the Huffman tree or as an unencoded symbol. Assuming the tree state depicted in Fig. 40f, the letter ‘c’ would thus be encoded as the binary code “101”, where the initial ‘1’ bit indicates that the following bits are taken from the Huffman tree, and the “01” bits refer to the branches taken to reach the node (0: left, 1: right). Similarly, symbols not contained in the table, like the numeric digit ‘2’ can be represented as “000100010”, where the first ‘0’ bit indicates that it is followed by an unencoded symbol, and the “00100010” bits contain the 8-bit ASCII representation of the digit.

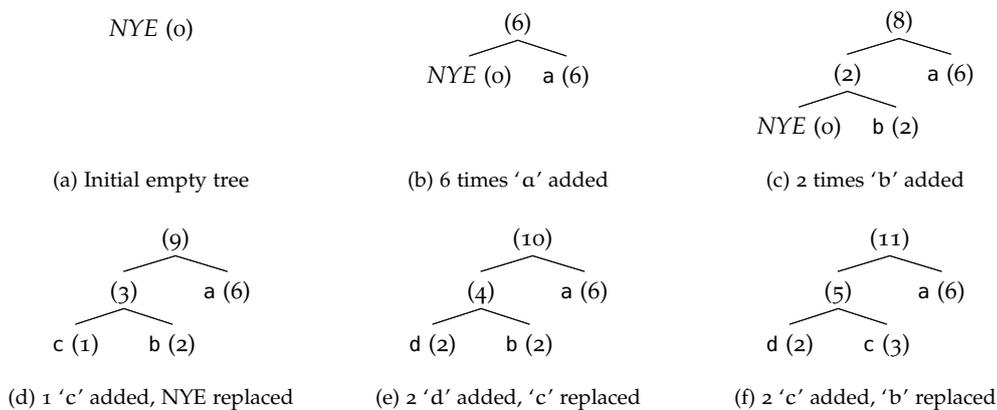


Figure 40: Populating a tree with capacity for 3 symbols with the sequence “aaaaaa bb c dd cc”

CODE LOOKUP TABLE By reducing the size of the Huffman tree, the computational efforts for both tree traversal and tree maintenance can be diminished. A further extension to reduce the computational efforts required for encoding and decoding is the use of a lookup table that maps all symbols contained in the tree to the according codewords. However, the use of a code lookup table introduces a significant demand for additional memory. Considering the lack of dynamic memory allocation in the TinyOS operating system, the worst case code length is 255 bits (analog to the definition of f_f in Fig. 38 for $n = 256$), introducing a memory demand of 32 bytes for each symbol. In addition to the binary representation of the codeword, its length also needs to be stored, resulting in 33 bytes per entry in the lookup table. The memory demand Δ_{RAM} can be approximated as $\Delta_{RAM} = n * (\lceil \frac{n}{8} \rceil + 1)$, where n represents the number of symbols in the Huffman tree. A comparison of resulting lookup table sizes for different Huffman tree sizes is presented in Table 18. In case a Huffman tree of reduced size is used, the memory demand of the lookup table reduces due to two factors:

1. **SHORTER CODE LENGTHS** The worst case code length is directly proportional to the number of symbols stored in the tree. As a result, proportionally less memory needs to be statically allocated for the code lookup table when less nodes are present in the Huffman tree.
2. **LESS ENCODED SYMBOLS** In the general case, the number of entries in the lookup table is identical to the number of symbols stored in the tree, i.e. a cache entry is present for every symbol. Huffman trees with less nodes thus inherently require less codewords to be stored in the lookup table.

In summary, while the use of a code lookup table is a feasible approach to reduce the computational overhead for the tree traversal, the incurred demand for more than eight kilobytes of RAM hampers its application when Huffman trees with space for 256 symbols are used. Although reducing the number of entries in the lookup table leads to reductions of the memory demand, its population strategy must optimize the tradeoff between frequently used symbols with short codewords and less often used symbols with longer codewords. In contrast, when less symbols are stored in the Huffman tree, the size of the code lookup tables drastically reduces due to the quadratic impact of n on the memory demand of the Huffman tree. The use of a code lookup table for all symbols contained in the Huffman tree is thus increasingly favorable when smaller trees are used.

5.3.4 Evaluation

Concluding from the compression gain estimates presented in Sec. 5.3.2, it is apparent that size reductions can already be achieved when using simplified code length approximations while limiting the number of entries within the tree. As a result, we have presented the design of an adaptive Huffman coding algorithm that operates on a limited code tree size. The algorithm's resource footprint is reduced significantly when less nodes are present in the Huffman tree, because only codes for the most frequently occurring input symbols are stored. Furthermore, less memory and computation time is required when encoding symbols due to the code lookup table, and the limited tree size also reduces the efforts for reorganizing the Huffman tree.

In this section, we analyze the achievable compression gains when the AHC algorithm with varying tree sizes is applied to the data sets introduced in Sec. 5.3.2. Furthermore, we

Table 18: Memory demand of the code lookup tables for different numbers of symbols in the tree

Symbols in tree	1	2	4	8	16	32	64	128	256
RAM (bytes)	2	4	8	16	48	160	576	2,176	8,448
% of TelosB RAM	0.02%	0.04%	0.08%	0.16%	0.49%	1.6%	5.6%	21.2%	82.5%

show the algorithm’s applicability on sensor node hardware by investigating the impact of the tree size on the energy demands of MCU and radio transceiver in the COOJA/MSPsim emulation environment. In a third and final step, we validate the energy-efficiency of our AHC implementation in a real-world experiment, where we measure the runtime gains of a TelosB node when it is supplied from a constant charge supply.

In our compression gain estimations presented above, we have assumed that the symbol distributions were known a priori and assigned the mappings between symbols and codewords statically. Due to the time-continuous nature of physical sensor readings, knowledge about the entire sequence of data to compress is generally not given in typical WSN scenarios. As a result, we have employed the adaptive variant of the Huffman coding algorithm, which adapts to the characteristics of the input sequence during runtime. We have compressed the four presented data sets with the algorithm and varied the number of symbols that can be stored in the tree. We show the sizes of the compressed sequences in Table 19 in comparison to the uncompressed data.

Notably, the achievable compression gains show a strong correlation to the used data set and its characteristics. However, the number of entries in the code tree also has a major impact on the compression gain. While very small values for the number of symbols allow to encode predominant symbols in a very efficient way, the one bit prefix increases the encoded length of all other symbols. Especially in the PermaSense and active Porcupine data sets with many different contained symbols, this even leads to size increases of the output for small tree sizes. In contrast, if too large values for n are chosen, the compression gain slightly degrades as a result of the longer code lengths of rarely occurring symbols. In summary, however, size reductions of 14% (PermaSense, $n=32$), 21% (Porcupine active, $n=32$), 62% (Porcupine sleep, $n=4$), and 82% (Glacswab, $n=1$) can be achieved when AHC with limited tree sizes is applied.

ENERGY ANALYSIS Energy-efficient operation is a mandatory prerequisite for data processing algorithms used on motes in order to maximize the intervals between regular maintenance tasks (e.g. replacing batteries). We thus conduct a careful analysis of the energy consumption of our AHC implementation, and assess the impact of both MCU and radio transceiver on the total energy balance. Only if the transmission energy savings due to the packet size reductions exceed the computational overhead incurred by the additional processing demand, energy-efficient operation – and thus our main motivation to adapt the AHC implementation to WSNs – is given.

In our evaluation of the algorithm’s energy-efficiency, we have again set up a topology composed of a single sender and a single receiver node. In order to ensure repeatable results, we have statically supplied the data sets to the simulated application, and assumed a lossless wireless channel. Packets were transmitted at a rate of ten packets per second, with packet

Table 19: Output sizes and ratio to input for AHC with different tree sizes

SYMBOLS IN TREE	PERMASENSE	GLACSWEB	PORCUPINE	
			ACTIVE	SLEEP
Uncompressed	591,900 bytes (1.0)	27,144 bytes (1.0)	89,754 bytes (1.0)	89,754 bytes (1.0)
1	625,211 bytes (1.06)	4,903 bytes (0.18)	91,172 bytes (1.02)	72,816 bytes (0.81)
2	595,944 bytes (1.01)	7,929 bytes (0.29)	88,487 bytes (0.99)	49,835 bytes (0.56)
4	567,247 bytes (0.96)	7,794 bytes (0.29)	84,249 bytes (0.94)	34,504 bytes (0.38)
8	539,434 bytes (0.91)	7,766 bytes (0.29)	79,065 bytes (0.88)	34,940 bytes (0.39)
16	517,086 bytes (0.87)	7,759 bytes (0.29)	74,431 bytes (0.83)	34,940 bytes (0.39)
32	510,933 bytes (0.86)	7,772 bytes (0.29)	70,931 bytes (0.79)	34,940 bytes (0.39)
64	519,592 bytes (0.88)	7,807 bytes (0.29)	71,884 bytes (0.80)	34,940 bytes (0.39)
128	537,240 bytes (0.91)	7,869 bytes (0.29)	71,972 bytes (0.80)	34,940 bytes (0.39)

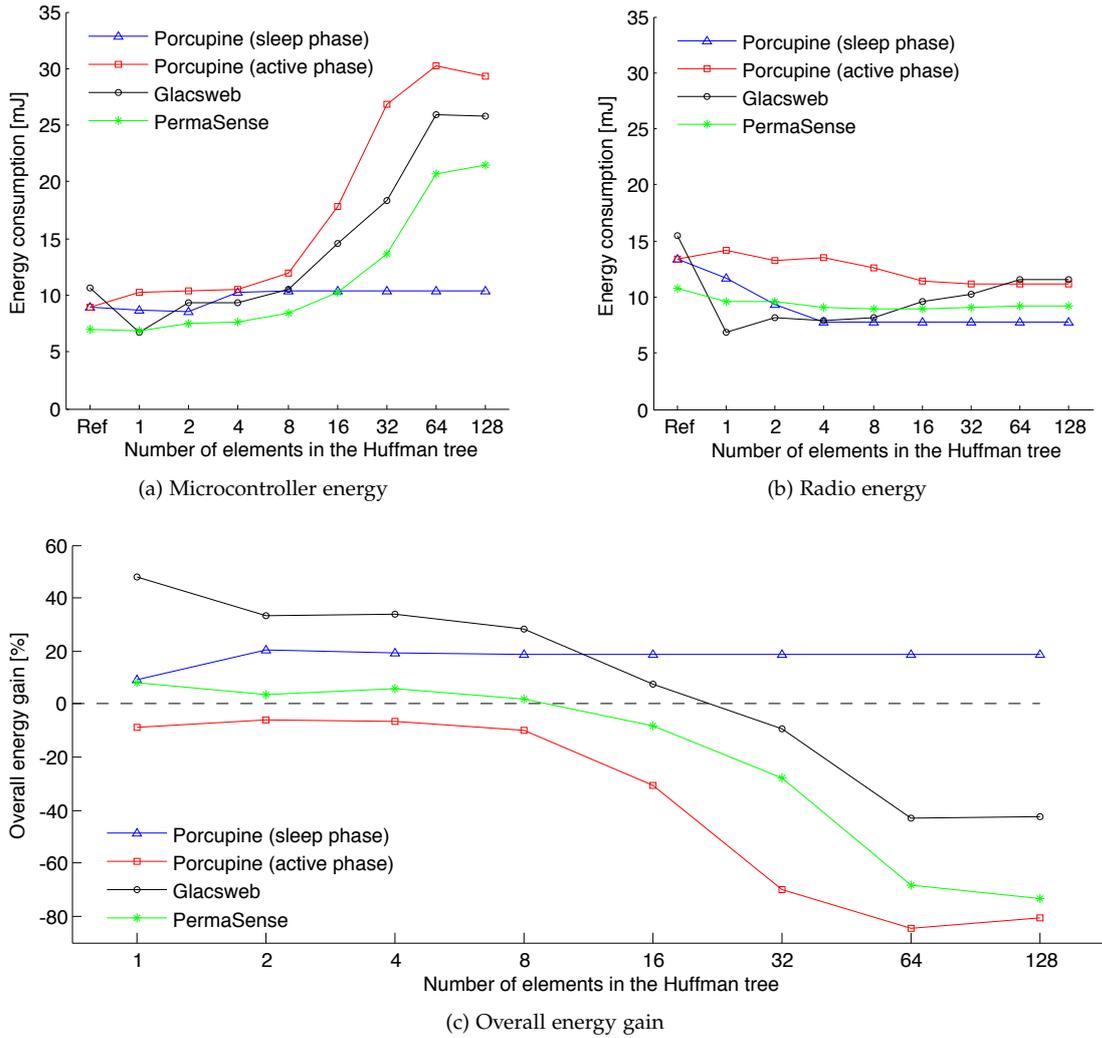


Figure 41: Energy analysis for the adaptive Huffman coder with limited tree size

sizes of 30 bytes for the PermaSense data trace, 52 bytes for the Glacsweb data set, and 42 bytes for each of the Porcupines traces. Analog to [EOF⁺09b], we use the current consumptions measured by Dunkels et al. in [DOTHo7] in our energy analysis. Analog to our previous analyses, we have used the NullMAC protocol implementation (i.e. the radio transceiver of the receiver node is always active, so the sender radio only needs to be switched on during packet transmissions) to eliminate the impact of the MAC protocol on our analysis.

We have confined our analysis to the energy requirements of the sender node, as only marginal changes occur to the receiver's energy consumption when its radio device is not duty-cycled. The results of our energy analysis of the sender node are depicted in Fig. 41, from which it becomes evident that the use of trees with a limited number of nodes can effectively lead to reductions in the packet sizes, as observed through the reduced amount of energy spent on radio transmissions in Fig. 41b. Significant radio energy savings are achieved for the Glacsweb (55% for $n=1$) and Porcupine sleep (42% for $n=4$) data sets. In case of the PermaSense and both Porcupine data sets, the reduced packet sizes lead to a consistent decrease in radio energy, until reaching a steady state at 17% compression gain (at $n \geq 8$) for the PermaSense, and 16% at $n \geq 32$ for the active phase of the Porcupine, respectively. Only in case of Glacsweb data, the great number of input symbols with low frequency leads to the assignment of long codes, resulting in degraded compression gains when larger code tree sizes are used. On the contrary, an increase in MCU utilization occurs due to the additional

Table 20: Number of packets transmitted and runtime gains in the real-world experiment

SYMBOLS IN TREE	REF.	1	2	4	8	16	32	64	128
Sent packets	4,733	6,832	6,668	6,609	5,991	5,947	4,979	4,496	2,581
Runtime gain	0%	44.3%	40.9%	39.6%	26.6%	25.6%	5.2%	-5.0%	-45.5%

processing needs, as shown in Fig. 41a. Again, the Porcupine sleep data sets exposes behavior different to the other traces, because its tree only contains five symbols. For the other data sets, a rise in the MCU energy demand is clearly visible, indicating the increased amount of energy required for managing and restructuring the trees.

The overall energy requirements, depicted in Fig. 41c however still prove that for the limited code tree size adaptive Huffman coder, energy gains can be observed for three of the data sets when appropriate tree sizes are chosen. Only the Porcupine data set captured during the activity phase shows negative energy gains for all tree sizes despite our observation in Table 19, where we have shown that up to 21% size reductions could be achieved for a tree size of 32 elements. The energy losses can be attributed to the fact that only little compression gains were achieved for small tree sizes, while the high computational demand for large Huffman trees strongly exceeds the energy savings through the transmission of slightly smaller packets.

Due to the fact that only five symbols are present in the input sequence of the Porcupine data set captured during the wearer’s sleep phase, the energy gains do not show any dependencies on tree sizes of eight symbols or more. In all other cases, the computational overhead and thus the demand for MCU energy linearly increases with the tree size (bearing in mind that the x-axis of the figures is logarithmic) for tree sizes of up to 64 symbols. While the overall energy gains remain comparably steady for trees of one to eight symbols, the increased compression gains for larger tree sizes do not counterbalance the additional computational energy demand. In order to maximize the overall energy gain, our analysis has shown that tree sizes between one and eight symbols are thus favorable. Across the four analyzed data sets, a value of $n = 4$ has shown the highest compression gains.

REAL-WORLD EXPERIMENT In order to verify if the simulation results match the algorithm’s real behavior, we have set up a real-world experiment using two TelosB devices. The first node was configured as a sender node and supplied with the Glacswab data set, for which compression gains of up to 82% and energy gains of up to 58% were determined in our previous analyses. Blocks of data were read from the node’s memory, compressed using our adaptive Huffman coder with limited code tree sizes, and transmitted over the radio. The impact of the radio transceiver’s energy consumption was minimized by turning it off after each transmission.

In contrast to sophisticated measurement setups based on specific measurement hardware, like presented in [KW08a], we have limited the energy budget available to the node by connecting its battery terminal to a boost converter powered by a supercapacitor of 25 farads capacitance at a maximum operating voltage of 2.7 volts. In order to allow for comparable measurements, we have put the same charge on the capacitor by connecting it to a regulated 2.5 volts power supply prior to each run of the experiment for at least five minutes. A receiver node with no energy restrictions was also part of the experiment, and was used to count the number of transmitted packets in the used indoor environment. Both were configured to use NullMAC, thus allowing to compare the results to the previously performed analyses. The results of the real-world experiment with the Glacswab data set are presented in Table 20 and confirm that the application of the algorithm leads to energy savings for the given data. Similarities can be observed to the results shown in Fig. 41c, where a runtime gain of up 58% for a single element in the tree as well as losses of up to 23% for a tree size of 128 elements were observed. In summary, our real-world experiment has confirmed that our simulation results also translate well into reality.

5.3.5 Summary

In this section, we have presented an adaptation of the adaptive Huffman coding algorithm for its application in WSNs. In a preliminary compressibility analysis for statically defined code trees, we have investigated whether size reductions are possible when Huffman coding is applied to a stream of packets. The results have shown that packet sizes can be reduced through the application of Huffman coding even when only a fraction of the symbols contained in the input sequence are present in the code tree. After having presented our design considerations for the AHC algorithm with limited tree sizes, we have analyzed the resulting compression gains for real-world data traces from different domains. Energy-efficient operation is mandatory in WSNs, hence we have supplemented our discussion by an analysis of the algorithm's energy demand for both the MCU and the radio transceiver. We have shown that significant energy savings are possible for all data sets, however only when tree sizes of less than 16 symbols are being used. A real-world experiment has additionally been conducted and shown that energy savings can also be achieved in practice.

The implementation of AHC with reduced tree sizes can be easily integrated into the Squeeze.KOM compression framework and serve as an additional option for compressing packet payloads in which strongly biased symbol distributions are given. By extending the lifetime of sensor and actuator nodes, the application of the adaptive Huffman coding algorithm presents a solution to the requirement of energy-efficient operation in smart spaces.

5.4 VALIDATION AGAINST REAL-WORLD DATA SETS

We have highlighted the need for real-world data sets in Sec. 3.2.2 and consequently based our previous analyses of compression and energy gains on real-world data collected from environmental monitoring and body-area sensor deployments. In order to assess the applicability of the presented data compression solutions in the envisioned application area of smart spaces, we supplement our previous evaluations by determining the achievable compression and energy gains for data collected in an indoor office setting. First, we present the collection of data sets from smart environments using the TWINS.KOM testbed [RKHS08] in this section. In a successive step, the compressibility of the collected data is analyzed, both when proprietary message formats are used as well as when the sensor readings are encapsulated into S-RPC messages.

5.4.1 Data Set Selection

Benchmarks of traditional data compression algorithms commonly rely on reference corpora like the Calgary corpus [BWC89]. While they are well suited in order to assess an algorithm's compression gain on desktop computers, a significant difference to the analysis of data compression algorithms in WSNs exists. On desktop computers, data compression algorithms mostly operate on complete files, whereas data streams are prevalent in sensor networks. This results from the fact that a fundamental property of WSNs is the collection of real-time sensor data. As a result, absolute compression gains and energy consumptions can be determined for desktop computer data compression algorithms more easily. Because the input data on motes is received in the form of a continuous stream of sensor readings, no general statement about the compressibility or the exact energy demand for the compression of a certain sensor stream can be made in advance. In contrast, only through the use of representative sets of real data traces, retrospective analyses of the according compression gains and energy consumptions can be performed.

Two possibilities exist for the collection of data sets that allow the assessment of a compression algorithm's energy demand. On the one hand, a representative number of real sensor traces can be collected and supplied to the simulation environment in the form of input data for a virtual sensor device. On the other hand, accurate models of real sensor data can be extracted from close observations of real-world sensors, and be used in simulation

to generate representative traces on demand. However, it has been shown that the artificial generation of sensor data that reflects the characteristic properties of the real sensor device is a non-trivial task [WI95, Sie01]. Furthermore, statistical models of sensor data tend to become complex, while they still cannot maintain all properties of real-world sensors, e.g. when faults or unexpected events occur. Finally, a practical issue hampers the existence of sophisticated sensor models for different sensor and scenario types: A large amount of sensor data from a representative set of deployed sensors needs to be collected in order to accurately reflect their properties and derive a valid model. As data in WSANs are being collected in real time and predictions about future values are always linked to certain levels of uncertainty, analyses are generally limited to estimations of average compression gains for different sensing modalities. We have hence refrained from devising sensor data models and instead decided to collect real-world data traces in order to provide a basis for the evaluation of the presented data compression algorithms. We present the traces collected from different sensing modalities in an indoor deployment, analyze their properties, and motivate their use in the following evaluation of our devised data compression algorithms.

Based on the observation (cf. Sec. 5.2.1) that data sets collected in real-world sensor networks are rarely available to the public, especially for sensor deployments in indoor environments, we have taken the decision to address this shortcoming by setting up a dedicated indoor sensing infrastructure. The decision to capture sensor data in an office environment has been specifically made in order to resolve the shortage of available indoor sensor data traces and to provide a foundation for the evaluation of the proposed data compression algorithms in the context of smart environments. As a result, we have designed and installed the TWiNS.KOM testbed in an office environment in order to capture realistic sensor data from an indoor setting. The data collected on our sensor testbed thus complements available data sets from other application areas of WSNs like body area sensing and environmental monitoring.

5.4.2 *The TWiNS.KOM Sensor Network Testbed*

For the collection of real-world sensor data in an office environment, the installation of sensor devices is indispensable. However, instead of configuring individual sensor nodes to collect data from the indoor environment, we have decided to realize the data collection through the use of a testbed, i.e. a specific infrastructure to facilitate experimentation. As opposed to the manual deployment of pre-programmed devices, our TWiNS.KOM testbed provides means for simplified firmware installation and node debugging, allowing for an easier retrieval of the collected sensor data. Besides catering for the collection of sensor traces, the testbed has been designed for the evaluation of WSAN applications under real-world conditions. The validation of simulated algorithms on real sensor node hardware is essential in the process of application development, because characteristics of the real world, especially the data captured by sensors and the impact of user mobility on the wireless channel conditions, cannot be completely modeled in simulation environments.

While on-the-fly deployments of motes often suffice to evaluate the practical behavior of applications and collect real-world sensor samples, they expose a number of downsides that reduce their applicability in many cases. Primarily, the time required to distribute new application images to all participating nodes, to set them up in the desired topology, and finally to perform an analysis of the measured characteristics log files, is a time-consuming process. The manual deployment of nodes often eliminates the repeatable character of experiments, because slight alterations in the node's orientation have already been shown to result in measurable discrepancies in signal reception [LLSo6]. As a result, the use of fixed testbed installations has become prevalent to ensure repeatable results and ease the deployment process. Testbeds present a ready-made deployment of nodes with versatile application developer support in different regards. Application images can be deployed easily, experiments can be controlled and supervised during runtime, and measured results and log files are processed and presented to the user after an experiment in order to allow for a detailed analysis of the application behavior. Especially in cases where simulated behavior

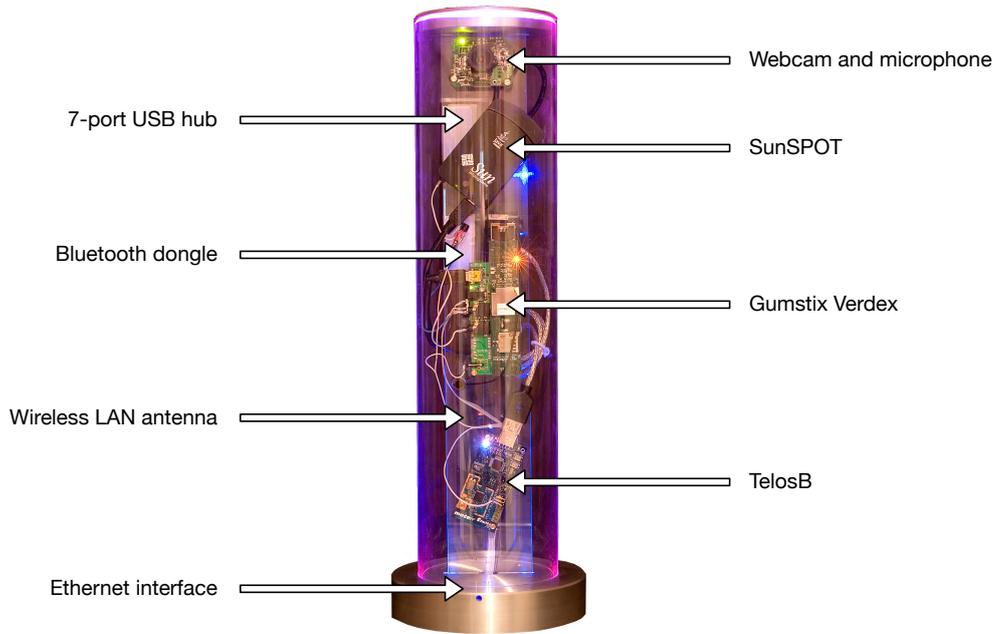


Figure 42: Photo of the tubicle, in which the locations of the individual components are indicated

and measurements on real hardware do not match, the debugging capabilities of a testbed are very useful tools to determine the origin of the discrepancy. By logging application behavior and making it available to the user, encountered events and errors can be identified, and improvements to the application can be derived from the observations. Various testbeds like Motelab [WASW05], Trio [DHJ+06], Kansei [EAR+06], and TWIST [HKWW06], have in consequence been designed and deployed.

The TWiNS.KOM testbed is composed of *tubicle* nodes, depicted in Fig. 42. An overview of the devices and interconnections in the tubicles is given in Fig. 43. Supporting multiple dimensions of heterogeneity (computational power, radio protocols, sensor devices, and available memory), and the possibility of simple relocation due to its portable design, the

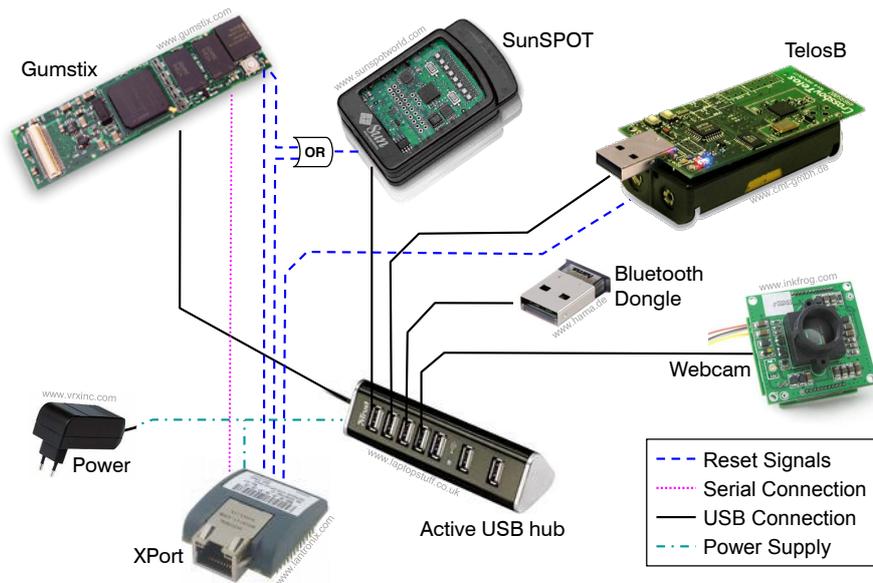


Figure 43: Internal connection diagram of the tubicle node

platform can be used to perform a broad range of experiments. Besides the integrated sensors for brightness, temperature, humidity, acceleration, sound, and video, the TelosB and SunSPOT motes can be configured to act as gateway nodes, and thus interface further sensing devices wirelessly. By means of its wireless connectivity over the IEEE 802.11 (Wireless LAN), IEEE 802.15.1 (Bluetooth), and IEEE 802.15.4 standards, devices using any of these protocols can be interfaced by the tubicle nodes easily.

5.4.3 Characteristics of Real-World Sensor Data

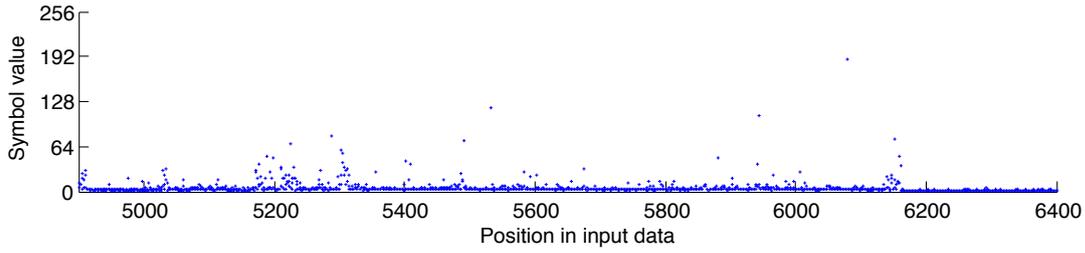
In order to assess the performance of the presented lossless payload compression algorithms in the envisioned application domain of smart environments, we have collected sensor data traces from the following modalities using our TWiNS.KOM testbed:

- ▷ **SOUND LEVEL** In order to capture the sound level, we have configured the microphone integrated in the tubicle's webcam to operate continuously. The sound level has been quantized at 8 bits resolution, and was sampled for a period of ten seconds each, after which the root mean square of the collected amplitude readings has been returned as the sensor value.
- ▷ **POWER CONSUMPTION** The electric power consumption of the user's desktop workplace, comprised of two desktop computers (consuming 32 and 93 watts in idle state) and two monitors (consuming 57 watts each), has been captured using the SmartMeter.KOM platform [RBM⁺11]. Readings were collected every twenty seconds and wirelessly sent to the tubicle.
- ▷ **DOOR MOTION** We have installed an ultrasound distance meter in order to measure the motion between the door frame. Its analog output voltage was interfaced to a mobile TelosB node, which has been configured to record the average door motion reading over a period of ten seconds, and transmit the latest readings to the tubicle. Due to the reflection of the door frame, a small offset voltage is continuously output by the sensor.
- ▷ **TEMPERATURE AND HUMIDITY** The Sensirion SHT11 sensor mounted on the tubicle's TelosB node has been used to periodically sample the temperature and humidity of the office room. The tubicle has been configured to collect samples of this environmental properties every ten seconds.

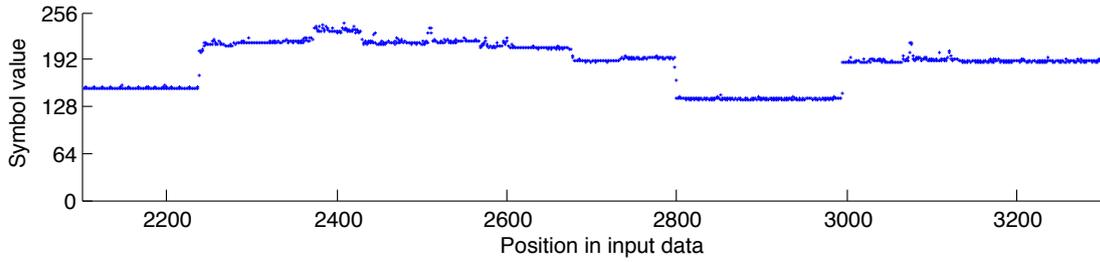
Measurements for all modalities have been collected during the time period from September 2010 to May 2011, on both weekdays and weekends. On most of the contained weekdays, the employee has been in his office during the regular working hours. Excerpts of the data traces for each sensing modality are shown in Fig. 44, while the complete traces of the data sets used in our analysis are visualized in Appendix A.3. Because all values were collected as unsigned integer values of one byte in size, the sensor readings have been directly mapped into 8-bit ASCII symbols.

The excerpt shown in Fig. 44a represents a period during which the employee was holding a conversation in his office room, before getting back to computer work at sample offset 6,150. In the power consumption curve visualized in Fig. 44b, the computer of lower power demand is activate from its sleep mode (at sample offset 2,250) and subsequently experiences different load levels, which result in slightly different power consumption values. At sample offset 2,800 the computer is again put into sleep mode, before being reactivated at sample 3,000. The trace of the door motion measurement is shown in Fig. 44c, and shows a period during which the door was closed (up to sample offset 3,170) before several people were entering and leaving the room through the open door (until offset 3,580). Finally, the temperature and humidity traces exhibit smooth characteristics with mostly continuous sensor readings, and are shown in Fig. 44d and Fig. 44e, respectively.

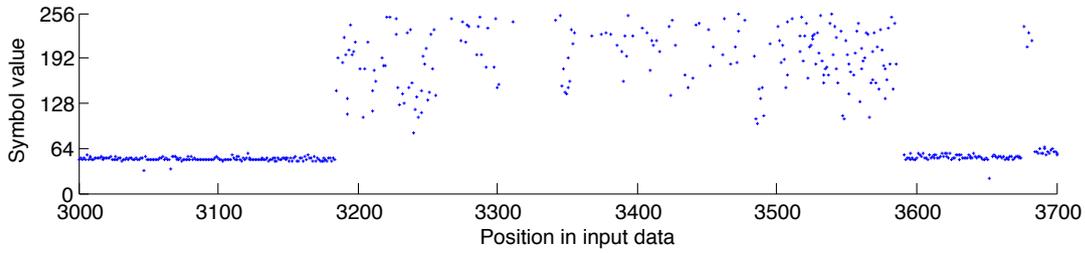
In order to attain an estimate for the compressibility of the complete data sets, we show the analysis of their symbol distributions in Fig. 45. Analog to our methodology taken in Sec. 5.3.2,



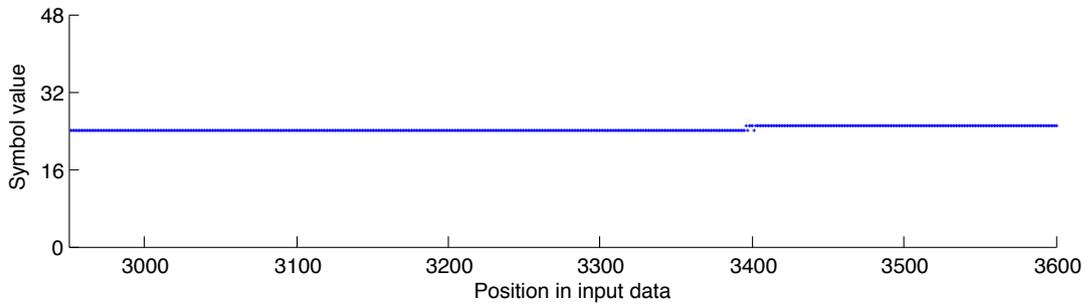
(a) Representative data trace excerpt from the TWiNS.KOM sound level measurement



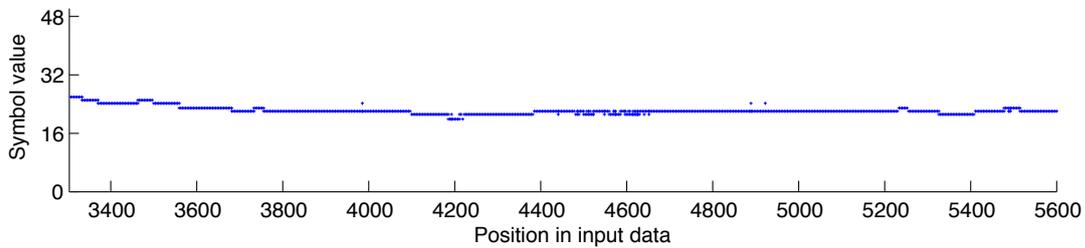
(b) Representative data trace excerpt from the TWiNS.KOM power measurement



(c) Representative data trace excerpt from the TWiNS.KOM door motion measurement

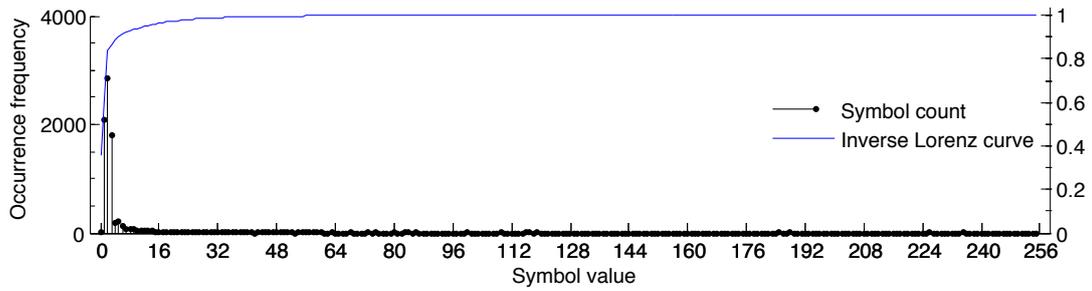


(d) Representative data trace excerpt from the TWiNS.KOM temperature measurement

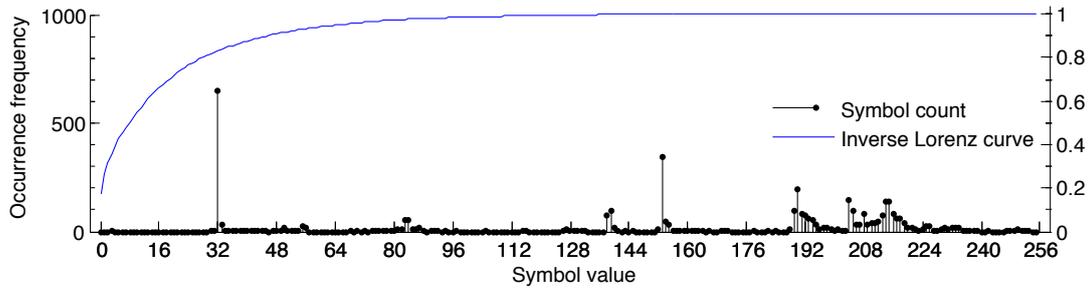


(e) Representative data trace excerpt from the TWiNS.KOM humidity measurement

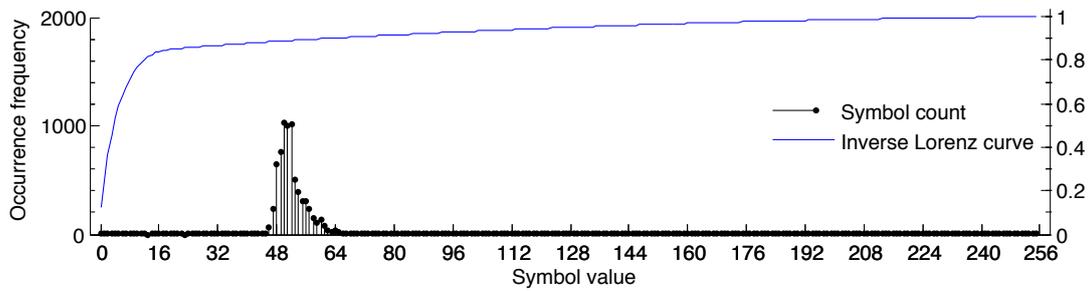
Figure 44: Visualization of representative excerpts of the used data sets



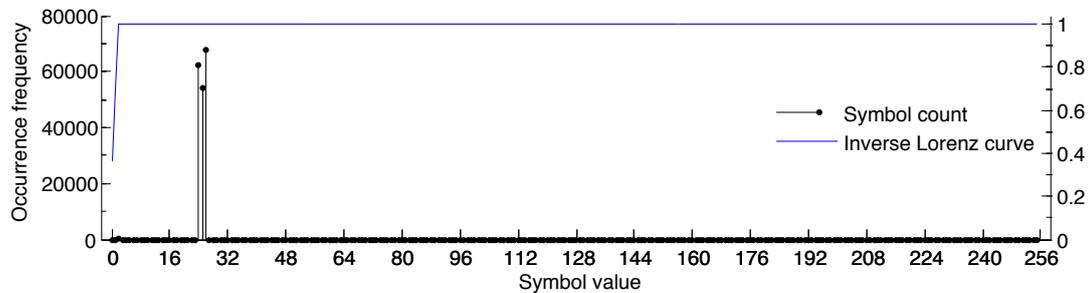
(a) Data characteristics of the TWiNS.KOM sound level measurement



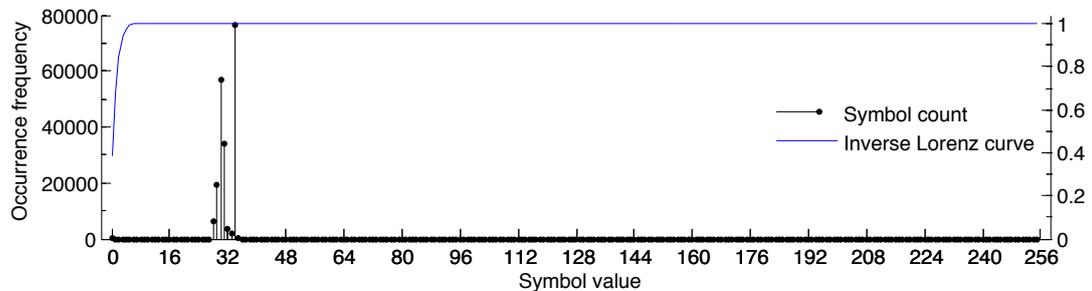
(b) Data characteristics of the TWiNS.KOM power measurement



(c) Data characteristics of the TWiNS.KOM door motion measurement



(d) Data characteristics of the TWiNS.KOM temperature measurement



(e) Data characteristics of the TWiNS.KOM humidity measurement

Figure 45: Visualization of the symbol distributions and inverse Lorenz curves for the used data sets

Table 21: Characteristics of the data sets collected using TWiNS.KOM

DATA SET	TOTAL NUMBER OF SYMBOLS	90TH PERCENTILE
Sound level	169	2
Power	256	56
Door motion	256	23
Temperature	9	3
Humidity	18	12

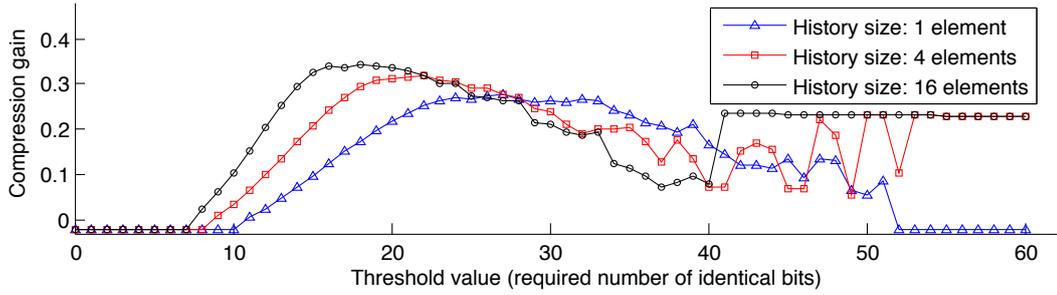
the actual occurrence numbers of each individual symbol as well their inverse Lorenz curves are shown in the figures. Similar to the environmental monitoring and body area sensing data sets, the occurrence frequencies of the used symbols are not distributed evenly over the data set. In contrast, the data sets rather expose a number of symbols with significantly greater occurrence numbers, which is also confirmed by the steep initial rise of the inverse Lorenz curve. We quantify the deviations from uniform symbol distributions by analyzing the total number of symbols as well as the 90th percentile (i.e. the number of distinct symbols that make up 90% of the input sequence of each trace) in Table 21. The latter indicator allows us to specify the deviation from a uniform symbol distribution, in which 90% of input sequence would be comprised of 230 different symbols. The most significant discrepancy from the uniform distribution can be observed in the sound level and temperature traces (Figs. 45a and 45d), in which 90% of all input symbols are represented by two or three sensor values only. In summary, uneven symbol distributions are present in all of the collected data traces, which can be seen as an indicator for the viability of data compression.

5.4.4 Applicability of Lossless Payload Compression on Real-World Data

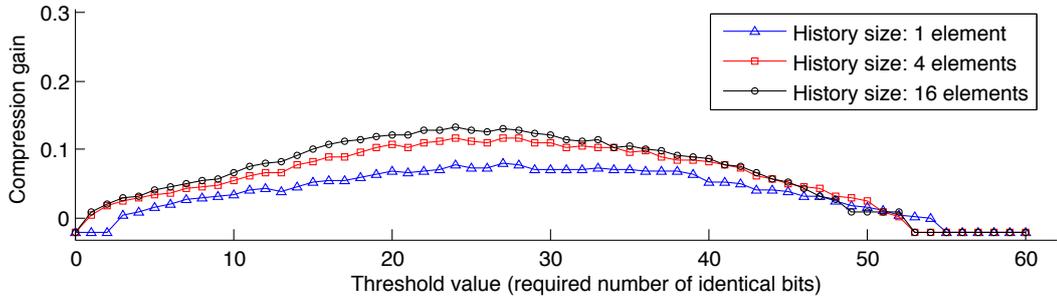
As a result of the fact that the data sets introduced above have been directly collected from time-continuous signals, only a single byte of data has been collected at each sampling point. In order to cater for a feasible ratio between packet payloads and headers, we have composed the packet payloads in the applicability study from 20 consecutive readings of the same sensor. The following analyses of both Squeeze.KOM and the adaptive Huffman coding algorithm are composed of two steps each. First, a simulative study is performed in order to assess the optimal configuration parameters of the compression algorithm. Subsequently, the compression algorithm is parametrized according to the determined values and executed in the MSPsim hardware emulator in order to assess its energy demand.

In a first step, we have analyzed the data traces for each sensing modality with regard to their compression gains and the amount of energy that can be preserved through the application of Squeeze.KOM. We have determined the parameter values for the similarity threshold th and the history size n that lead to the highest compression gains through means of simulations for threshold values below 60 bits and three different history sizes, which are visualized in Fig. 46. The resulting values for th and n that lead to optimal compression gains are listed in the third column of Table 22. In a second step, we have configured Squeeze.KOM according to the determined parameters and conducted the energy analysis in the COOJA/MSPsim environment. The energy gains achieved in the simulation environment are presented in the rightmost column of the table. Similar to our observations in Sec. 5.2.4, positive energy gains could be observed for three of the five data sets. Only the door motion and power measurements have shown small compression gains due to the high variance of the readings, and resulted in negative energy gains when compressed using Squeeze.KOM.

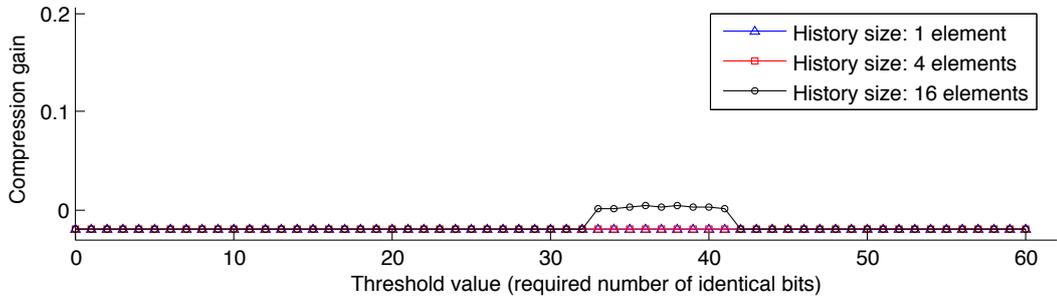
In a second step, we have assessed the size and energy gains achievable when adaptive Huffman coding with limited tree sizes is employed. Based on our observation that energy losses are almost exclusively encountered when tree sizes of more than eight symbols are used, we have confined our analysis to trees of eight or less symbols in size. In a simulative



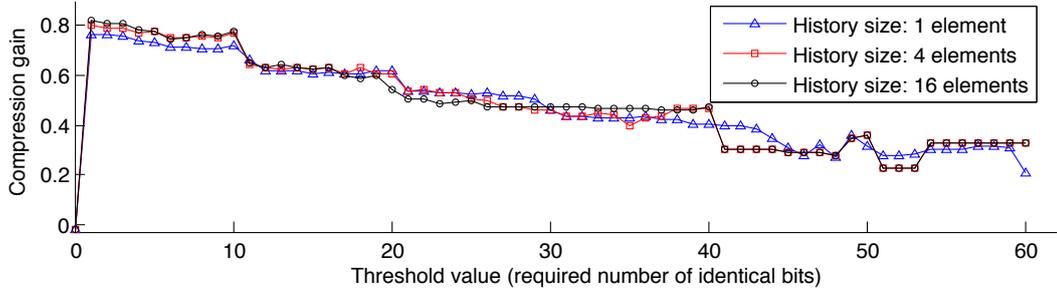
(a) Impact of history size and threshold value on the compression gain of sound level measurements



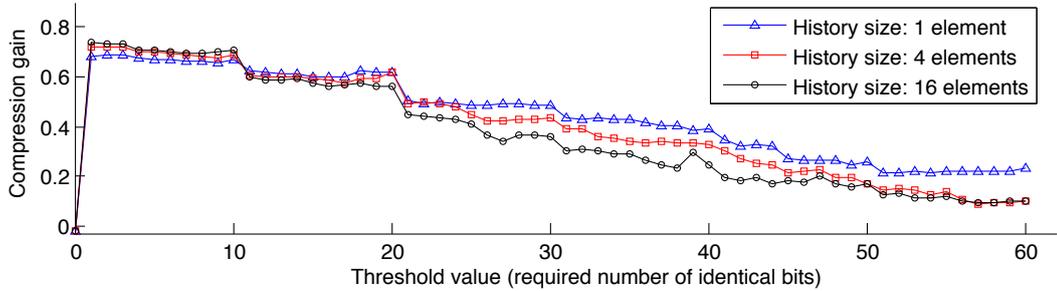
(b) Impact of history size and threshold value on the compression gain of power measurements



(c) Impact of history size and threshold value on the compression gain of door motion measurements



(d) Impact of history size and threshold value on the compression gain of temperature measurements



(e) Impact of history size and threshold value on the compression gain of humidity measurements

Figure 46: Achievable compression gains when Squeeze.KOM is applied to real-world data

Table 22: Compression and energy gains resulting from the application of Squeeze.KOM

DATA SET	OPTIMUM PARAMETER VALUES	COMPRESSION GAIN	ENERGY GAIN
Sound level	th: 18 bits, n: 16 entries	34.2%	1.4%
Power	th: 24 bits, n: 16 entries	13.2%	-2.2%
Door motion	th: 36 bits, n: 16 entries	0.45%	-2.8%
Temperature	th: 10 bits, n: 16 entries	82.9%	8.2%
Humidity	th: 1 bit, n: 16 entries	74.0%	7.6%

study, we have analyzed the compression gains for the given data sets, and studied the energy gains when the application is executed in the COOJA/MSPsim environment. The results are compared in Table 23, where the highest energy gain for each of the data sets is highlighted in bold font. Similar to the results of Squeeze.KOM, the high variance of the power meter readings has resulted in energy losses of at least 1.3%. For the traces of all other sensing modalities however, measurable positive gains are encountered that even exceed the gains achievable by Squeeze.KOM. This can be attributed to the fact that continuity of sensor readings (i.e. the absence of rapid changes) is better exploited by the application of entropy coding, while Squeeze.KOM has been designed for the compression of packets without correlations between the fields.

In summary, we have shown that both compression algorithms are suited for their operation in WSANs, and can lead to energy savings when the underlying data features biased symbol distributions or similarities between successive packets. The higher correlations between the continuous sensor data has proven the adaptive Huffman coder with limited tree sizes to result in better overall energy gains of up to 16.7%, whereas only up to 8.2% of energy gains could be achieved by applying Squeeze.KOM. Only the limited compressibility of power meter readings, which is also indicated by the comparably high number of symbols in its 90th percentile, has resulted in energy losses when compressing the power readings with either algorithm.

5.4.5 Applicability of Lossless Payload Compression on S-RPC Messages

In a final evaluation, we analyze the compressibility of S-RPC messages. The format of the messages used in S-RPC has been specifically tailored to carry a signature header which indicates the message type, the number and types of the contained parameters, as well as an eight bit sequence number. Except for the sequence number, repeating invocations of the same remote functionality can be expected to carry identical values in the S-RPC header fields as well as the same name of the invoked method. Only the sequence number can be assumed to be changing randomly under the assumption that the consumer has established RPC sessions to multiple providers. In order to evaluate the compression gains when S-RPC messages

Table 23: Compression and energy gains resulting from the application of adaptive Huffman coding

DATA SET	SIZE / ENERGY GAINS FOR TREE SIZE OF			
	1 SYMBOL	2 SYMBOLS	4 SYMBOLS	8 SYMBOLS
Sound level	16.0% / 3.7%	48.6% / 7.5%	57.5% / 6.2%	58.2% / 5.1%
Power	1.0% / -1.3%	2.1% / -1.5%	6.0% / -1.3%	11.1% / -2.7%
Door motion	1.3% / 2.4%	9.2% / 2.6%	18.7% / 3.0%	29.1% / -0.8%
Temperature	1.4% / 7.9%	22.0% / 16.7%	53.3% / 12.5%	55.6% / 12.5%
Humidity	2.5% / 6.9%	8.6% / 11.5%	17.9% / 10.4%	30.2% / 10.3%

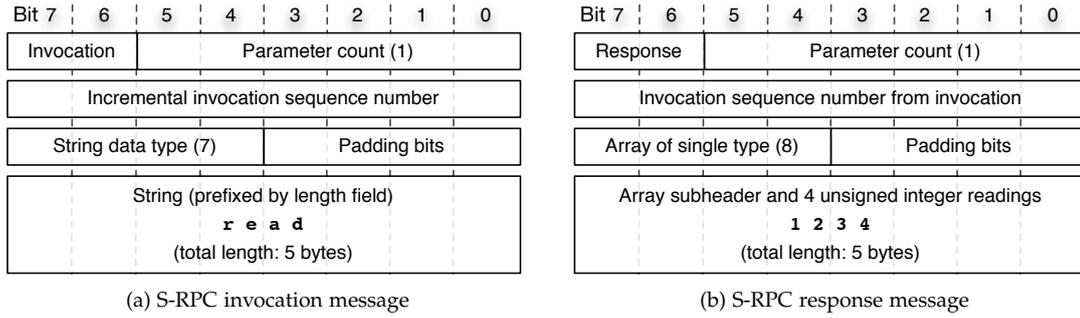


Figure 47: Messages used in the compressibility evaluation of S-RPC messages

are being compressed, we have converted the data traces collected using the TWiNS.KOM testbed into S-RPC packets, according to the structures depicted in Fig. 47. The invocation message is eight bytes in size, of which seven bytes can be assumed constant when the same remote function is invoked. For the changing field, the invocation sequence number, we have inserted a monotonically incremented value in each successive packet. The RPC response has also been designed to be eight bytes in size, of which the invocation sequence number and the actual sensor readings (present in the form of an array with four elements, according to the structure shown in Fig. 10) encounter changes, while the remaining header fields are maintained constant.

We have analyzed the compression gains achievable by the application of the adaptive Huffman coder with limited tree sizes in a first step, and present the results in Table 24. Due to the large number of constant fields in the S-RPC request messages, the application of adaptive Huffman coding results in compression gains of up to 37.9% for request messages when eight symbols are stored in the Huffman tree. The table furthermore shows the achievable size and energy gains for S-RPC responses when four consecutive sensor readings are carried in the payload. Similar to the request messages, the constant header fields allow for compression gains in excess of 19% for all data sets. The parameter selection for which the highest energy gains are attained is highlighted in bold font for each of the data sets, indicating achievable energy savings between 0.5% and 1.9%. In summary, the application of adaptive Huffman coding is viable for the use in conjunction with S-RPC, and can be utilized in order to mitigate the energy overhead of 0.11% (cf. Sec. 4.3.4) introduced by the S-RPC stack.

As opposed to the efficient encoding of correlations within a sequence of sensor readings, the Squeeze.KOM compression layer dedicatedly exploits similarities between successive packets. Analog to our previous evaluations, we have determined the optimum parameter values for the differential coder in a preliminary analysis and subsequently assessed the energy demand of the implementation in the simulation environment. The resulting compression and energy gains are shown in Table 25, and indicate that the compression of S-RPC requests

Table 24: Compression and energy gains resulting from the application of adaptive Huffman coding

DATA SET	SIZE / ENERGY GAINS FOR TREE SIZE OF			
	1 SYMBOL	2 SYMBOLS	4 SYMBOLS	8 SYMBOLS
S-RPC request	0.0% / -0.5%	0.1% / -1.0%	24.9% / 0.5%	37.9% / -0.8%
S-RPC response (sound level)	8.0% / 1.3%	11.7% / 1.4%	27.6% / 0.8%	35.7% / -0.3%
S-RPC response (power)	0.0% / -0.3%	0.0% / -0.3%	17.8% / 0.5%	19.6% / -2.4%
S-RPC response (door motion)	1.6% / 0.5%	1.7% / 0.2%	19.8% / 0.6%	24.6% / -1.6%
S-RPC response (temperature)	2.9% / 1.9%	4.8% / 1.9%	22.8% / 1.9%	35.9% / 0.5%
S-RPC response (humidity)	2.8% / 1.4%	4.3% / 1.6%	17.8% / 1.7%	25.6% / -0.6%

Table 25: Compression and energy gains resulting from the application of Squeeze.KOM

MESSAGE TYPE	OPTIMUM PARAMETER VALUES	ACHIEVABLE GAINS	
		SIZE	ENERGY
S-RPC request	th: 3 bits, n: 16 entries	34.1%	1.5%
S-RPC response (sound level)	th: 6 bits, n: 16 entries	10.2%	-2.1%
S-RPC response (power)	th: 7 bits, n: 16 entries	1.1%	-2.7%
S-RPC response (door motion)	th: 9 bits, n: 16 entries	-4.9%	-2.9%
S-RPC response (temperature)	th: 3 bits, n: 16 entries	23.9%	1.1%
S-RPC response (humidity)	th: 4 bits, n: 16 entries	22.2%	1.1%

leads to compression gains of up to 34.1% when the number of elements in the history is at least equal to the number of distinct remote functions invoked. Accordingly, energy gains of up to 1.5% are observed for S-RPC invocation messages, in which only the sequence number field encounters changes. When response messages are compressed, the sensor readings in the payload impact the achievable compression gains. Furthermore, the transmission of differential messages entails a two byte overhead for the status field and the hash value of the used index packet, and as a result, the corresponding energy gains range from -2.9% for highly varying data to 1.1% for the temperature and humidity readings.

In contrast to the application of Squeeze.KOM on messages that are exclusively composed of sensor readings, slightly higher energy savings can be achieved when Squeeze.KOM and S-RPC are combined. The smaller energy demand results from the fact that matching packets to the history is reduced to a comparison of the S-RPC signature, i.e. the structure definition and the name of the invoked method. As a result of the small number of bytes to compare, differential thresholds of less than eight bits of the signature header are sufficient to detect similarities between successive packets.

5.4.6 Summary

We have presented the TWiNS.KOM testbed, which has been used to collect sensor data sets from an office environment during a deployment period of nine months. Using the testbed, data traces for five sensing modalities (door motion, power consumption, temperature, humidity, and sound level) have been captured. Our analysis of their symbol distributions has shown that all data sets deviate from a uniform symbol distributions, however to a different extent. While time-continuous signals of slowly changing physical properties like temperature and humidity rarely show large differences between successive sensor readings, the power meter data traces exhibit more fluctuations and less continuity. As a result of this fact, neither of the presented mechanisms could compress the power metering data sufficiently to achieve energy savings. For all other traces, our study has shown that the proposed lossless payload compression algorithms achieve measurable size reductions. Positive energy gains were observed for all data sets, except for the data set composed of power readings, clearly confirming the feasibility of our proposed data compression algorithms in WSAWs deployed in smart environments. Our analysis of the compressibility of S-RPC messages has furthermore shown that less overhead is incurred when S-RPC messages are being compressed, because the determination of the similarity to history elements reduces to a comparison of the signature header and the method name. As a result, compression gains of up to 34.1% are observed, leading to runtime gains of 1.5% in case Squeeze.KOM is used. Similarly, the adaptive Huffman coder reduces S-RPC messages by approximately 20% of their size and has been shown to result in energy savings between 0.5% and 1.9% for the collected data sets. In summary, both are well applicable for the application in conjunction with S-RPC and easily counterbalance the energy overhead entailed by the S-RPC stack.

Be sincere; be brief; be seated.

Franklin D. Roosevelt

In our analyses of lossless packet payload compression algorithms, we have determined that energy savings can be achieved by reducing the sizes of the transmitted packets. While we have exclusively focused on the compression of payloads in the previous chapter, actual packets are comprised of several other fields. We have therefore depicted the packet structure of the TinyOS operating system in Fig. 48, according to the IEEE 802.15.4 standard [IEE06] and the TinyOS Extension Proposal 126 [MHLCo7]. The payload is surrounded by several header and footer fields of 18 bytes in total, of which only a single byte — the ActiveMessage (AM) type identifier — is introduced by the TinyOS operating system. Based on the observation that header and footer fields comprise a significant part of a radio packet, we have analyzed the potential of packet Header Compression (HC) in wireless sensor and actuator networks. Analog to the previous chapter, our contributions exploit the fact that shorter transmission durations directly map to a reduced energy consumption of the radio transceiver. As a result, we eliminate redundancies in header fields of subsequent packets by omitting fields with static or deterministic values from transmission.

First, we investigate the compressibility of packet headers by determining the transmission relevance of each field individually. We subsequently analyze the applicability of existing HC concepts to the characteristics of WSANs. Based on the attained insights, we present the design of our SFHC.KOM algorithm, a solution for stateful header compression tailored to the demands of WSANs with both mobile and fixed nodes. We finally prove the applicability and benefits of our approach, and analyze the packet overhead and energy consumption of our header compression algorithm in different representative settings.

6.1 COMPRESSIBILITY OF HEADER FIELDS

In order to assess the maximum extent of size reductions that can be achieved through the application of HC, we analyze the compressibility of all individual header fields shown in Fig. 48 in the following list:

- ▷ **SYNCHRONIZATION HEADER** The synchronization header is present at the beginning of every wireless transmission and caters for the correct synchronization between sender and receiver node. It is composed of the preamble sequence (generally realized in the form of a repeating pattern of the same symbol) and the Start-of-Frame Delimiter (SFD), which initiates the receiver's transition from synchronization to the actual reception mode. The synchronization header has an overall length of five bytes and cannot be compressed, because the compliance to the IEEE 802.15.4 standard is otherwise lost and the risk of synchronization errors increases.

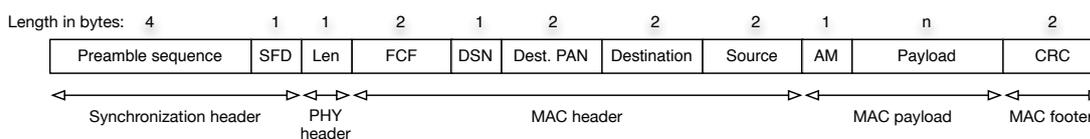


Figure 48: Structure of a complete TinyOS radio packet

- ▷ **PHYSICAL LAYER HEADER** The Physical Layer (PHY) header contains a single byte of data only which indicates the length of the following MAC Protocol Data Unit (MPDU) in the lowermost seven bits. As a result, the maximum MPDU size is 127 bytes, equalling a maximum MAC payload of 116 bytes. Although the length field could be omitted from transmission when packets of fixed sizes are exclusively used, the resulting solution would no longer conform to the IEEE 802.15.4 standard, and packets without the PHY header would be silently discarded by the radio transceiver.
- ▷ **MAC HEADER** The MAC header is composed of the Frame Control Field (FCF), a sequence number field, and three address fields. The FCF contains seven flags which indicate the type of the following data, if a hardware acknowledgement is requested for the given packet and in which form the following addresses are represented. The FCF is followed by a Data Sequence Number (DSN), which is automatically incremented by the transceiver device and used to determine duplicate packets. Finally, the MAC header contains the identifier of the destination Personal Area Network (PAN) as well as both sender and receiver addresses. Although different addressing modes are supported by the IEEE 802.15.4 standard, TinyOS uses address fields of 16 bits in size in its packet headers. In a communication sequence between two nodes within the same PAN, all addressing fields and most fields in the FCF remain constant, whereas the DSN is monotonically incremented until it reaches the wraparound point. As a result, most fields from the MAC header are constant or only change deterministically. A more detailed analysis of the MAC header fields is presented in Sec. 6.3.3.
- ▷ **MAC FOOTER** The end of the MPDU is marked by a two-byte frame check sequence, typically realized in the form of a Cyclic Redundancy Check (CRC) which is calculated over the MAC payload. Its omission would reduce the total size of the packet, however at the cost of increased susceptibility to undetected transmission errors.

The loss of compatibility with IEEE 802.15.4-compliant radio transceivers disallows the omission of the synchronization and PHY headers, while the frame check sequence in the MAC footer is essential in order to detect bit errors introduced on the transmission channel. As a result, the efficient encoding of the MAC header remains the only viable option to reduce the packet header size. At a total size of nine bytes, the MAC header also represents the largest contributor to the 17 byte packet overhead. Although the IEEE 802.15.4 standard requires that the first field in the FCF “shall be set to one of the nonreserved values” [IEE06], its use in order to signal the presence of compressed packet headers is supported by most radio transceivers, even though full compliance to the standard is no longer given.

When analyzing traffic in WSN deployments, similarities in the MAC header fields can be observed in the majority of cases. For sensor networks in monitoring and surveillance scenarios, all sensor data are commonly forwarded to a dedicated sink node. Consequently, tree-based routing protocols are applied in order to establish routes along which the data packets are being forwarded. When adopting a node’s local perspective, packets carrying sensed data are forwarded to the next hop in the direction of the sink. Because this next hop is typically located within the same PAN, and transmission parameters defined in the FCF rarely change between individual transmissions, the DSN represents the only field in the packet header that undergoes changes. Similarly, when broadcasting-based data dissemination protocols are used, only the DSN field of the MAC header experiences changes, while all remaining fields maintain constant values. Finally, even when operating on a mesh topology, the number of individual addresses contained in the MAC header is limited. Only the next hop of a multi-hop transmission is defined in the MAC header, while the packet’s logical destination address needs to be provided by higher layers. Even when mobility is present in the network, or routing schemes introducing diversity [EGBM09] are employed, many fields of the packet header remain unchanged and are suitable to be compressed. We have thus decided to design a solution that fits all aforementioned needs and reduces the energy expenditure for packet transmissions by reducing the size of the MAC header. Although header compression

could also be applied to headers on higher layers of the protocol stack, specialized solutions would be required in order to support the large variety of existing transport protocols in WSN research. We thus confine our analysis to a compression of the MAC header, which is constant throughout all implementations of the IEEE 802.15.4 standard.

6.2 REQUIREMENTS AND DESIGN DECISIONS

After having determined that header compression in WSNs can only be applied to the entries in the MAC header, the requirements and design considerations for our header compression algorithm are presented in this section.

REQUIREMENTS We have observed that smart spaces differ from traditional sensor network application domains in various regards. In order to cater for the applicability of header compression in WSNs in smart spaces, we have thus determined the following requirements to the HC algorithm:

1. **SUPPORT FOR NODE MOBILITY** In smart spaces, sensor and actuator nodes may be both statically deployed (e.g. temperature probes) or affixed to users or mobile objects (e.g. wireless beacons for indoor positioning). As a result, the possible presence of mobile nodes needs to be taken into account during the design of our HC algorithm. The first major requirement to the proposed header compression algorithm is thus its quick adaptation to mobility-induced changes to the network topology. In other words, it must exhibit a small overhead for connection establishment and low management traffic overhead throughout a communication session.
2. **AVOIDANCE OF LOSS AND ERROR PROPAGATION** Both the aforementioned requirement of support for mobile nodes and the unreliable nature of radio links in WSNs necessitate additional means to recover from packet losses. Especially when header compression algorithms require a mutual exchange of state information before commencing their operation, the loss of these initialization packets can lead to situations where none of the following packets can be decoded. Each individual hop on a multi-hop route from the source to the sink represents a possible point of failure for the success of the entire data transmission. Loss propagation, i.e. the fact that once a packet loss has occurred, all subsequent packets cannot be transmitted successfully any longer, represents a major issue in WSNs and needs to be specifically addressed in the header compression approach. Similarly, when the loss of updated compression information is not detected, errors are propagated through the network because further packets are decoded incorrectly based on outdated information. The negative impacts of both loss and error propagation should thus be minimized by taking suitable countermeasures.
3. **ENERGY EFFICIENCY AND LOW RESOURCE DEMAND** Due to the resource limitations of embedded sensing devices, the HC algorithm should not consume significant resources of the underlying mote platform. In accordance with our prior observations, an essential requirement to compression algorithms is their positive energy balance, i.e. the compression of the MAC header must consume less energy than the transmission of uncompressed packet headers. The tradeoff therefore is to find a header compression algorithm which reduces the sizes of compressed headers to a small fraction of their initial size to allow for savings through shorter radio transmission times, while adhering to the resource limitations of the given node platform.

DESIGN DECISIONS Taking the aforementioned requirements for a header compression algorithm into account, we have taken a set of decisions during the design phase of our header compression scheme. The fundamental idea behind header compression is the omission of constant or redundant header fields from their transmission. While packet payloads can be compressed both in lossless and lossy form, the need for an exact representation of the

fields in the packet header is mandatory. A second similarity to the presented packet payload compression approaches is the fact that header compression algorithms can either operate on a stateless per-packet basis, or maintain state information and exploit knowledge about previous packet headers.

In order to highlight the differences between both approaches, we summarize the characteristics of both stateful and stateless header compression in the following list:

- ▷ **STATELESS HEADER COMPRESSION** In case of stateless operation, neither sender nor receiver maintain any connection-specific information. As a result, most header fields cannot be completely omitted from transmission, because the receiver cannot decode the packet otherwise. A prominent example for stateless HC is presented in RFC 4944 [MKHC07]. It is optimized for link-local IPv6 communication with specific efficient encodings for UDP, TCP, and ICMPv6 packets. While drastic size reductions can be achieved when these transport protocols are employed, transport protocols specifically tailored to WSNs (e.g. ESRT [OBAA05] or ERTIP [LHCJ09]) are not supported. Although the reduction of TCP headers from 20 to only 6 bytes can be considered a remarkable compression gain, the size of the MAC header is barely reduced when the stateless 6LoWPAN header compression approach [MKHC07] is applied.
- ▷ **STATEFUL HEADER COMPRESSION** When stateful header compression is applied, state information about previous packet headers or specific agreements for the interpretation of encoded packet headers is stored at both sender and receiver. The application of stateful header compression thus incurs a negotiation overhead to mutually establish a common shared knowledge about the connection — the so called compression context. Although establishing a context naturally poses additional overhead on a connection, the stateful operation allows to replace a greater fraction of information through referring to the previously defined context. Besides, by maintaining a list of shorthand terms for frequently used header field values, a limitation to previously defined header fields is not inherently given. Stateful header compression thus allows for even smaller packet header sizes than stateless HC mechanisms, however they suffer from one important drawback, namely the susceptibility to packet losses. When updates to the compression context are lost, error or loss propagation are likely to occur.

In order to maximize the achievable size reductions whilst keeping the resource demand low, we have chosen to implement the stateful compression of packet headers. We base this decision on the fact that many header fields in WSNs only change in a deterministic manner or even remain constant throughout an entire communication session. In the chosen stateful header compression approach, such fields are maintained in a mutually agreed compression context instead of being present in each outgoing packet. In further communications, this context is then referred to by a unique identifier, the so called Context Identifier (CID).

We specifically design our algorithm with options to cater for mobile behavior, such as persons carrying sensor nodes while walking through the WSN deployment. Upon relocation of a node that has already established compression contexts with neighbors, specific care needs to be taken to avoid collisions between the CIDs it is currently using and the CIDs used in its new neighborhood. When a node is known to be mobile, the HC algorithm must thus provide support to maintain the local character of CIDs. In order to avoid CID collisions when a node enters a new environment, we propose a two-stage approach for the connection establishment, specifically adapted to scenarios where fixed gateways and mobile leaf nodes are present. The corresponding compression states are presented in detail in Sec. 6.3.1. We assume that nodes have local knowledge about their mobility, defined using a flag in the application software. However, further means could also be used to set this mobility flag, such as GPS information, acceleration, or the extent of changes in the set of neighboring nodes over time.

Finally, in order to avoid error and loss propagation, error recovery is a crucial point in stateful HC. When packet losses or transmission errors occur during establishing a context, the entire HC session is affected, possibly degrading the packet reception rate at the receiver to

zero. We dedicatedly address error recovery by applying framed referential coding [SRMF06], a mechanism where a reference frame is transmitted uncompressed, and followed by a series of incremental differential packets that allow for the reconstruction of the original value when combined with the reference frame.

6.3 SFHC.KOM: STATEFUL HEADER COMPRESSION

In this section, we present the details of our realization of the stateful SFHC.KOM header compression mechanism. Based on our design decision to implement a stateful solution, a compression context composed of a reference header and the allocated CID is established prior to the exchange of data. Once the compression state has been exchanged, i.e. the context has been established, all packet headers are replaced by the corresponding CID, and only differential information on the changed fields is inserted to the packets by applying framed referential coding. In order to keep the overhead low, the CID negotiation is not performed proactively with all possible communication partners; instead, our reactive approach uses different compression states depending on the number of packets exchanged between the nodes.

6.3.1 Compression States

Our realization of SFHC.KOM is based on the state machine depicted in Fig. 49. After initialization, all nodes start in the C-NONE state, where no header compression is applied. Once the first packet transfer over a link without established context is requested by the application, SFHC.KOM immediately tries to establish a C-DEST context with the destination node, and on success enters the C-DEST mode, where all header fields except the destination address are contained in the context. If the context negotiation fails, e.g. because header compression is unsupported at the destination node, the C-NONE state is maintained. In case a CID has been successfully established, the C-DEST mode is used in all successive data transmissions between the sender and the receiver node. In case no packets have been transferred for an extended amount of time, a context loss is assumed and the state machine reverts to the C-NONE state.

The transition to the C-LONG mode is only made when both source and destination node have static positions and the number of packets transferred over the connection has exceeded a threshold th , which should be adapted depending on the average node degree. In our implementation, we have used a value of 20 packets. Analog to C-DEST, the efficient C-LONG encoding is used on all packets until a timeout occurs. If the C-LONG mode is not supported at the destination node, or the context establishment does not complete successfully, the nodes remain in the C-DEST state. The contents of the different compression contexts of SFHC.KOM are presented in the following list:

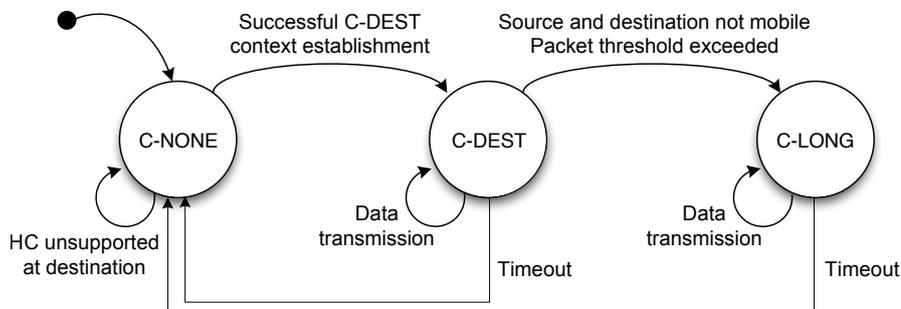


Figure 49: State diagram of the SFHC.KOM header compression algorithm

- ▷ C-NONE : In the initial state, headers fully compliant to the IEEE 802.15.4 standard are used, as if no header compression was present within the network. This mode has been implemented for interoperability with nodes that do not support header compression or in case of errors, such as when all connection identifiers are already occupied, and serves as the initialization state for all nodes. Also, all broadcast messages are sent uncompressed, obsoleting the a priori establishment of a common context state for all nodes.
- ▷ C-DEST : In order to refer to a mutually established compression context, the hardware address of the destination node as well as the CID are used in the C-DEST mode. After a context with a node has been established, header compression can be immediately used over this link. Mobility does not affect this mode of operation, as an individual addressing of the nodes is given and enables operation independently of any node movement. The unique addressing allows for the direct and quick establishment of a common context between two nodes without the necessity to take other nodes into account, so the context establishment overhead is comparably small. These properties also make the C-DEST mode very suitable for short-lived connections and when mobility is present.
- ▷ C-LONG : This mode of operation allows for highest compression gains, as the destination address is also stored in the compression context. However, in order to achieve a unique identifier for the context, more communication overhead is necessary, as its CID must be unique in both the sender's and the receiver's neighborhood. Although, depending on the number of neighbors, a larger number of packets is necessary for context establishment, this overhead can be amortized due to the long duration of a connection and the increased compression ratio. C-LONG is best suited for long-lived connections between static nodes, and for this reason, CID negotiation for the C-LONG mode will only take place after a defined number of packets have been transmitted.

The motivation to distinguish between two separate modes of operation is based on the impact of node mobility and thus the expected duration of the connection. When mobility is present within the network, and corresponding nodes change their location and thus their neighborhood frequently, complex context establishment processes are obviously inapplicable due to their overhead. On the contrary, when the topology is fixed, a more complex negotiation protocol can be applied for connection establishment, as the additional expenditure is expected to be counterbalanced by the reduced sizes of packet headers throughout the remainder of the communication session. When considering the three possible HC states, it becomes clear that the header compression module on each node needs to maintain a table containing the state of all connections. Once a message transfer to any node has taken place, its address is entered into one of three lists. In list L_{NONE} , all nodes without support for header compression are stored. The list L_{DEST} contains the addresses of all nodes to which a C-DEST context has been established, and list L_{LONG} stores all nodes for which a C-LONG context is present. While the necessity for the latter two tables is obvious, maintaining L_{NONE} is essential to avoid periodic attempts to establish a context with nodes without support for header compression. When a packet transmission is triggered by the application software, the destination address is extracted and the algorithm shown in Listing 4 executed.

6.3.2 Assignment of Connection Identifiers

The C-DEST mode has been designed to have a short context establishment phase with low overhead. Its CID negotiation operation is shown schematically in Fig. 50. In C-DEST, the address of the destination node is retained in the packets to allow for unique addressing even when mobility is present. All CIDs are thus only relevant in combination with the destination address, which is contained in a dedicated two byte header field. Given the presence of the destination address in the compressed headers, all nodes are addressed in a unique manner,

Listing 4: Application flow executed upon reception of a packet transmission request

Definitions:

- ▷ d : Address of destination node
- ▷ c_d : Counter for number of packets sent to d
- ▷ th : Context establishment packet threshold
- ▷ $L_{NONE}, L_{DEST}, L_{LONG}$: Lists of neighbors in C-NONE, C-DEST, or C-LONG modes

```

if ( $d \in L_{LONG}$ ) {
  send packet using established C-LONG context
} else if ( $d \in L_{DEST}$ ) {
  send packet using established C-DEST context
  increment  $c_d$ 
  if ( $c_d > th$  and source node is not mobile) {
    try establishing C-LONG context
    reset  $c_d$ 
  }
} else if ( $d \in L_{NONE}$ ) {
  send packet uncompressed
  increment  $c_d$ 
  if ( $c_d > th$ ) {
    try establishing C-DEST context
    reset  $c_d$ 
  }
} else {
  send packet uncompressed
  try establishing C-DEST context
}

```

so the process of context establishment reduces to the efforts of finding a free CID at both nodes. As shown in the figure, the sender node therefore broadcasts a C-DEST request packet, containing the destination address and a bit vector containing its locally available CIDs. The destination node determines the smallest available CID from the received packet and its locally available CIDs, and confirms the C-DEST request with an accept message containing the selected CID. The context state is then confirmed by sending a single packet with an uncompressed header in order to establish the context. When a C-DEST context cannot be established (e.g. because no CIDs are available), the attempt to set up the C-DEST context is terminated. Similarly, when header compression is not supported by the destination node, the expiry of a timer, which is started whenever a C-DEST request packet has been sent, cancels the negotiation and adds the destination node to L_{NONE} .

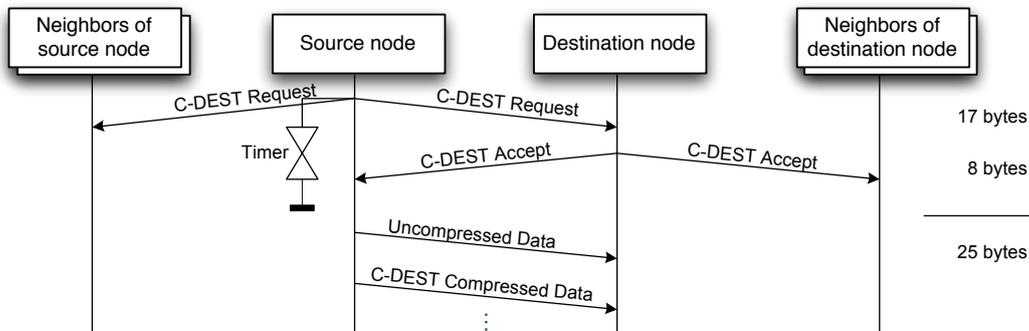


Figure 50: Context identifier establishment sequence of the C-DEST mode

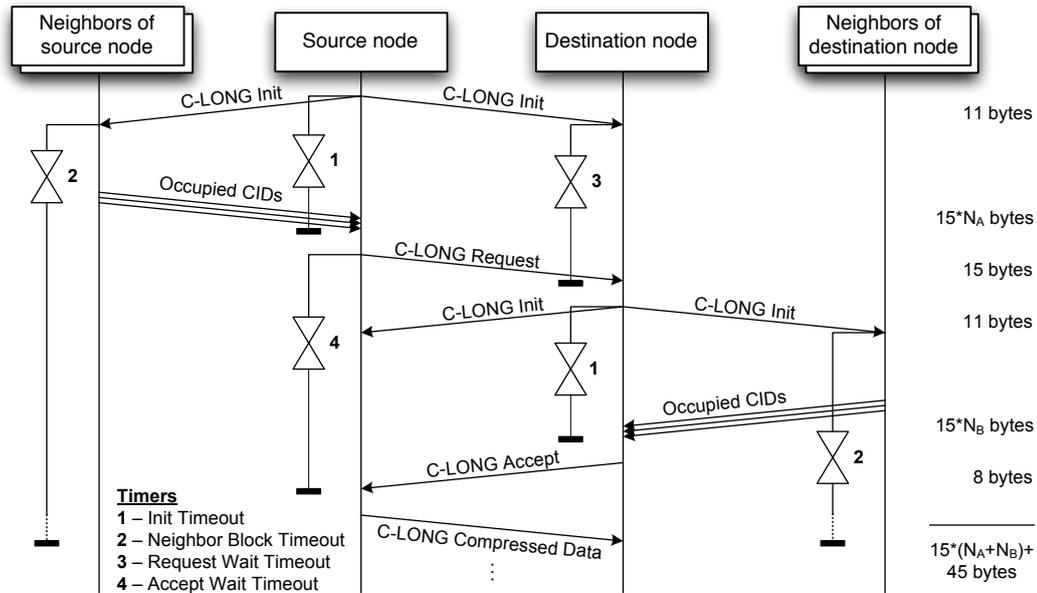


Figure 51: Context identifier establishment sequence of the C-LONG mode

Due to the greater overhead induced by negotiating a C-LONG context, it is only triggered when a sufficient number of packets have been exchanged, indicating that a long-lived session is present. When the initiating node is not marked as mobile, it can start the negotiation process of the C-LONG mode, which is depicted in Fig. 51. The initiating node starts the process by sending an initialization packet. It contains the desired destination node as well as the request to receive all of its neighbors' occupied CIDs. At the same time, it starts a timer (Timer 1) for the initialization timeout, at whose expiry it assumes that all occupied CIDs have been returned. All neighbors apart from the destination node return the bit vector containing their occupied CIDs, and start a timer (Timer 2) themselves, during which they do not initiate an own C-LONG negotiation. At the reception of a C-LONG initialization message, the destination node starts a third timer (Timer 3), during whose runtime no other context establishment is undertaken. It is only canceled when a request message is received, containing all CIDs which are available at both the sender and all its neighbor nodes. Once the destination has successfully received the C-LONG request message from the source, containing the list of available CIDs, timer 3 is stopped.

When sending a C-LONG request message, the source node also starts an expiry timer (Timer 4), during which the C-LONG establishment must have completed successfully, indicated by the reception of a C-LONG accept packet. The destination node however only sends this acceptance packet once it has collected all of its neighbors' occupied lists to choose a unique CID available at source and destination as well as all of their neighbors. The smallest available CID from both the request message and the destination's neighboring nodes is then returned to the initiator in the accept message, which concludes the initialization process. In case no CID is available, a reject message is sent to notify the sender to remain in C-DEST state. At a later time, the establishment of C-LONG can then be retried. In order to increase the reliability of context negotiations, link-layer acknowledgements are used for all C-LONG packets that contain bit vectors of occupied CIDs, as well as for the request and accept messages in both modes.

6.3.3 Compressed Header Structure

In order to successfully transmit compressed MAC headers, a flag must be contained in the packets to indicate if the header should be interpreted in compressed or uncompressed form. We have thus analyzed the FCF, in which the three most significant bits are used to discriminate

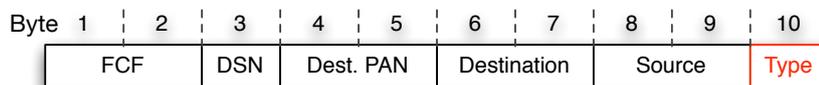
Table 26: Interpretation of the first six bits of the FCF according to IEEE 802.15.4 and SFHC.KOM

	b ₀	b ₁	b ₂	b ₃	b ₄	b ₅	INTERPRETATION
IEEE 802.15.4	0	0	0				Beacon frame
	1	0	0				Data frame
	0	1	0				Acknowledgment frame
	1	1	0				MAC command frame
SFHC.KOM	x	0	1	1	0	0	Acknowledgment
	x	0	1	0	1	x	Data frame
	x	0	1	1	1	x	Data frame with acknowledgment
	x	1	1	x	x	x	CID negotiation

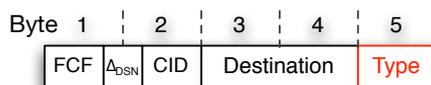
between different packet types. The corresponding excerpt of the IEEE 802.15.4 standard is shown in the upper part of Table 26. It becomes clear from the table that combinations where bit b_2 is set are not defined for actual packet types. Instead, the combinations are marked as reserved in the standard. We have thus decided to use these reserved bit combinations in order to differentiate between the frame types used in our stateful header compression approach. An experimental evaluation on the Texas Instruments CC2420 radio transceiver [Tex07] has shown that packets carrying a reserved frame type indicator in the FCF are not discarded by the radio transceiver, but instead forwarded to the mote’s operating system.

During normal operation, the IEEE 802.15.4 standard uses three bit flags in the FCF to signal if security enabled (b_3), a frame is pending (b_4), or an acknowledgment is requested (b_5). In combination with the reserved frame type bits, however, the interpretation of these fields can be changed in the operating system. As a result, SFHC.KOM uses the four frame types shown in the lower part of Table 26 (x indicates that further subtypes are defined). A complete reference of all FCF values is given in Appendix A.4. The resulting structures of the overall MAC headers for the different compression modes are shown in Fig. 52. We describe the altered definitions and interpretations of the header fields which we have used in SFHC.KOM in more detail in the following list:

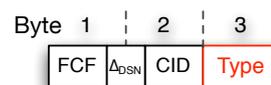
- ▷ **REDUCED FRAME CONTROL FIELD** Based on the fact that most fields in the FCF remain unchanged during a communication session, only a subset of values need to be transmitted. As a result, both C-LONG and C-DEST reduce the FCF to only six bits, relocating most fields from the original FCF into the mutually established compression context. The remaining FCF fields (indicated by x in Table 26) determine the following header structure (i.e. whether C-DEST or C-LONG is being used, or if an AM type field is present) and the interpretation of the following fields (data, acknowledgement, or compression context establishment traffic).



(a) Uncompressed IEEE 802.15.4-compliant header



(b) C-DEST compressed packet header



(c) C-LONG compressed packet header

Figure 52: Comparison of MAC headers for uncompressed and compressed packets

- ▷ **DELTA DATA SEQUENCE NUMBER** The DSN allows detection of duplicated frames and is incremented with each sent packet. In order to achieve robustness against loss and error propagation and simultaneously maintain the capability of duplicate detection, we apply framed referential coding on the DSN, encoding only the difference to an uncompressed reference value. The resulting Δ_{DSN} is four bits long, such that reference frames need to be sent every 15 packets the latest. The reference frame is an uncompressed packet, not only updating the DSN, but also refreshing the entire context state at the receiver side.
- ▷ **CONNECTION IDENTIFIER** Following the DSN field, the CID is transmitted. We have selected a size of six bits for the CID, allowing for up to 64 compression contexts to be established in parallel. The length of the CID field limits the number of compressed connections a node can handle and determines the size of the context tables used for header compression. Because the CID in C-LONG is responsible for context identification (decompression) and destination addressing, a unique CID allocation is particularly important in this mode.
- ▷ **TYPE FIELD** The ActiveMessage type is not part of the IEEE 802.15.4 MAC header, but specific to the TinyOS operating system and comparable to the notion of ports in TCP or UDP. ActiveMessage types are used to enable the operating system to dispatch incoming packets to the corresponding application. However, due to the fact that the number of applications being used concurrently in a deployed WSA_Ns is commonly small, only very few different ActiveMessage types are generally used. We have thus based our design of SFHC.KOM on the assumption that the AM type used in the communication between two entities will only change seldom. As a result, the AM type is also part of the compression context and thus not part of each outgoing packet.
- ▷ **ADDRESSING FIELDS** The comparison of header structures in Fig. 52 shows that the most remarkable size reductions of SFHC.KOM result from the omission of addressing fields. Once a compression context negotiation has taken place, both sender and receiver addresses are known to the communication partners, and can from then on be encapsulated in the context. Only the destination address is present in the compressed C-DEST packet header in case one communication partner is possibly mobile in order to avoid collisions between CIDs. The identifier of the destination PAN can be omitted in the general case. We have thus decided to maintain static addressing fields in the compression context in order to allow SFHC.KOM to reconstruct the fields from the table of compression contexts stored at the receiving node.

The contents of each compression context entry are shown in Table 27. Besides its CID, the constant fields from a full packet header are contained in the entry, i.e. the complete FCF as well as the destination PAN address and the hardware addresses of both communicating nodes. In order to omit the ActiveMessage types from transmission, their values are also stored in the context for data transmissions in both directions. Similarly, the base offsets of the DSN in both directions are stored, such that the current DSN can be easily restored by adding the differential DSN to the offset. A counter for the number of packets transmitted in a given context is included in order to assess if the threshold th has been exceeded and a state transition should be triggered. Finally, a periodically decremented counter is part of the context, and enables the deletion of unused contexts. In total, each compression context incurs a memory demand of 15 bytes for its storage.

While many of the existing header compression mechanisms targeted for the Internet assume a lossless channel, this assumption does not generally hold true in WSA_Ns [CWK⁺05, ZG03]. In order to recover from losses of data packets, we apply framed referential coding to the DSN, i.e. an offset value is periodically updated, and only the difference between the actual DSN and the offset is transmitted. Due to the infrequent changes of the ActiveMessage type, SFHC.KOM headers also contain the AM type only when differing from its previous value. To avoid loss propagation, i.e. allow for reconstruction of all transferred differential packets,

Table 27: Description of the fields stored in a compression context

DESCRIPTION	SIZE
CID of this compression context	1 byte
Complete 16 bit FCF field for this entry	2 bytes
Destination PAN address	2 bytes
16 bit address of the destination node	2 bytes
16 bit address of the source node	2 bytes
ActiveMessage type in transmission direction	1 byte
ActiveMessage type in reception direction	1 byte
DSN offset in transmission direction	1 byte
DSN offset in reception direction	1 byte
Counter for the number of packets transmitted using this CID	1 byte
Counter which triggers deletion of the CID upon expiry	1 byte
Total size	15 bytes

SFHC.KOM thus uses acknowledged transfers for all packets used to establish or maintain a compression context.

We quantify the expected packet size reductions by comparing the header sizes resulting from the application of SFHC.KOM against standard TinyOS headers and headers to which the stateless 6LoWPAN header compression [MKHC07] has been applied. The results are shown in Table 28. While C-DEST reaches size reductions of up to 56%, the C-LONG mode can even reduce packet headers by up to 78% of their original size. At maximum savings of 33%, the stateless 6LoWPAN header compression cannot reach the compression gains achieved by SFHC.KOM.

When modeling the packet size reductions analytically, and also taking the required overhead for establishing the context into account, the results shown in Fig. 53 are obtained. Including the reference frame transmitted whenever the four bit wide DSN field reaches its wraparound point, it becomes clear that the additional packets required to establish a context amortize after a number of transferred packets with compressed headers. For C-DEST, this is the case after only five transmissions, while it depends on the number of immediate neighbors in case of a C-LONG context establishment. In our analysis, between 20 (for five neighbors each) and 67 (for twenty neighbors each) packets are required to make up for the overhead of context establishment under the assumption that no losses occur on the radio channel. The figure also confirms that the overhead for context negotiation in short-lived neighborhoods with a small number of packet exchanges introduces additional overhead to cater for the establishment of the compression context.

Table 28: Compression ratios for compressed and uncompressed packet headers

HEADER TYPE	HEADER LENGTH AND RATIO TO UNCOMPRESSED HEADER	
	WITH AM TYPE/PORT	WITHOUT AM TYPE/PORT
TinyOS header	80 bits (1.0)	72 bits (1.0)
6LoWPAN HC	56 bits (0.7)	48 bits (0.67)
C-DEST header	40 bits (0.5)	32 bits (0.44)
C-LONG header	24 bits (0.3)	16 bits (0.22)

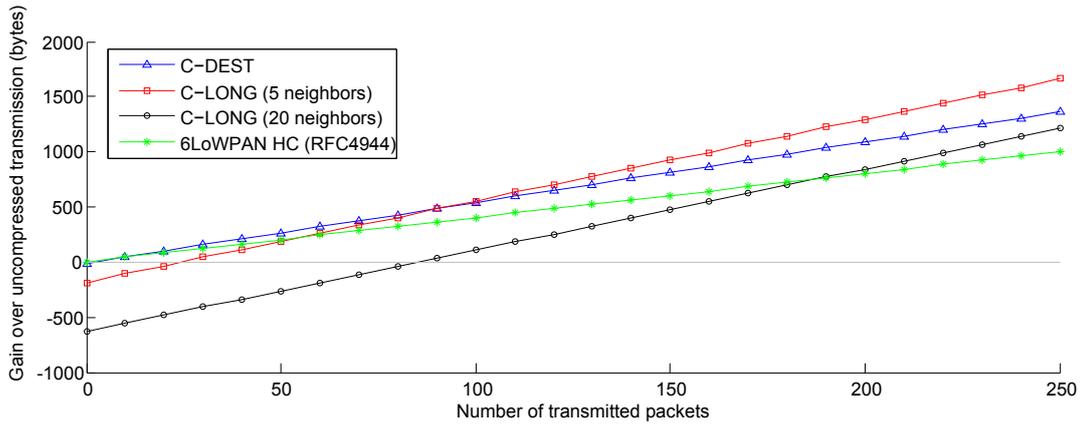


Figure 53: Analytical model of size savings when using C-LONG and C-DEST

6.4 EVALUATION

We analyze the algorithm’s performance and applicability based on an implementation of SFHC.KOM in the TinyOS operating system. While a real-world experiment is used in order to assess the compression gains of the solution, the energy demand of SFHC.KOM is evaluated in the COOJA/MSPsim environment. In order to meet our requirement of resource- and energy-efficient operation in real sensor and actuator networks, we base our evaluation on the TelosB mote platform [MEMo6b], which has been used in all conducted experiments and simulations. We assume a tree-based network topology, where all sensor data is forwarded to the root node, and adopt a unit disk radio model, in which wireless communication is only possible to a node’s immediate neighbors. The multi-hop tree-based CTP [GF⁺09] is used as the routing protocol.

Our evaluation is based on the topology depicted in Fig. 54, in which node 1 represents the sink node and thus the destination of all packets. A chain of nodes (labeled node 2 to 5) represents a tree part with great depth, and is useful in order to analyze the effects on the data transfer over long distances. In contrast, a cluster of nodes (labeled 6 to 9) represents a wide tree, where node 6 must aggregate and forward the data of its three neighboring nodes. Three mobile node positions (labeled I, II, and III) are depicted in the figure, but only used

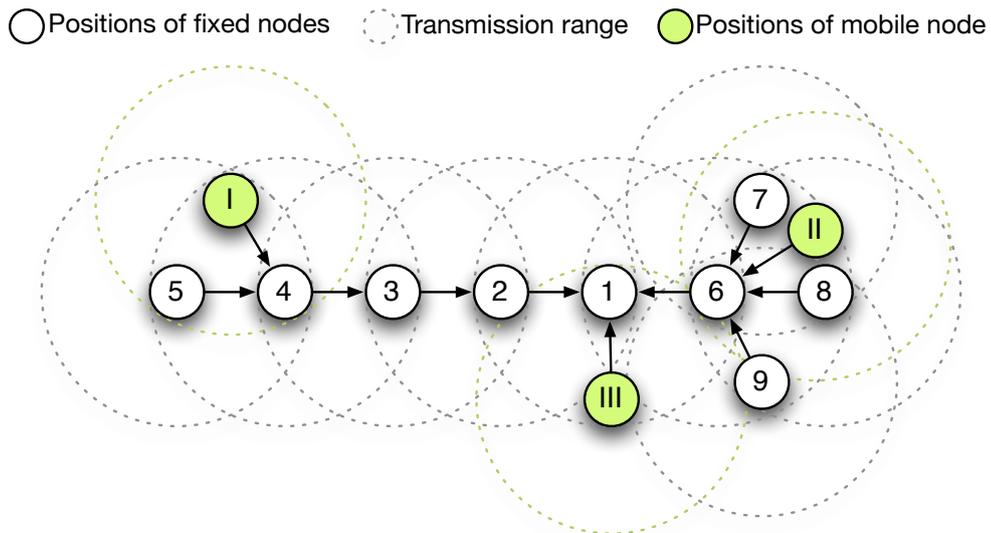


Figure 54: Network topology used in all conducted experiments

for the evaluation of our algorithm under the assumption that a hybrid network is present, comprised of a static network deployment and a single mobile node that takes one of the possible locations.

Based on the fact that packet payloads with realistic characteristics are irrelevant when assessing a header compression algorithm, we have used data packets that allowed us to easily confirm the successful transmission of packets. We have thus composed the payload of a two byte sequence number as well as the packet's source address in order to allow backtracking packets to their originating node. The CTP header added another 8 bytes to the payload, so that the overall payload size was 12 bytes. Packets were periodically transmitted at an interval of 2.5 seconds by each node, in order to eliminate the likelihood of congestion at the sink. In our evaluations of the compression gain, the experiment was run for 1,600 seconds, during which an overall number of more than 10,000 packets were emitted, with more than 600 packets originating from each source node. Similarly, the average energy demand over a time frame of 1,600 seconds has been analyzed in our energy evaluation. In both cases, the initialization overhead introduced by CTP has been dedicatedly disregarded.

The implementation of SFHC.KOM requires 13,786 bytes of program memory and 2,540 bytes of RAM, each representing about 25% of the TelosB's available resources. The origin of the significant size increment lies in the tables containing the context state information; almost half of the algorithms RAM demand results from the 64 context entries which need to be maintained at each node.

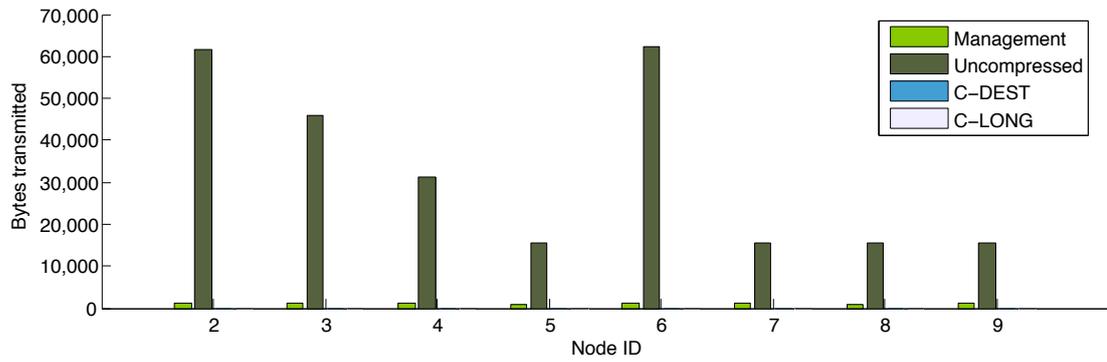
6.4.1 Compression Gain

The primary goal of SFHC.KOM is a reduction of the radio energy due to the fact that shorter packets are transferred. In order to attain a reference for the compression gains of SFHC.KOM, we have collected the total traffic volume transferred through the network without the application of header compression in a real-world experiment. The results are visualized in Fig. 55a. The chain topology from nodes 2 to 5 can be identified by the linearly decreasing traffic volume, resulting from the fact that each node forwards its own data as well as all data received from its children to the sink. Similar, node 6 acts as a clusterhead and forwards all traffic from nodes 7, 8, and 9 to the sink.

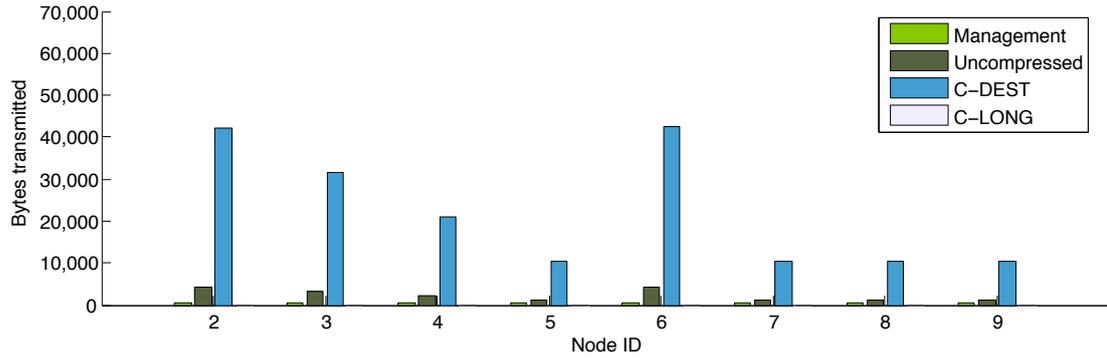
In the first step of our evaluation, we have configured all nodes to assume that they are mobile. This disallows them to enter the C-LONG mode, and thus enables dedicated measurements when only the C-DEST mode is used. After the CTP tree had been established, we have initiated the periodic transmission of packets. According to the behavior shown in Listing 4, the establishment of a C-DEST context is already tried after the first transmission of an uncompressed packet. We have measured the time required to establish the C-DEST context to be 47ms on average, confirming our assumption that context negotiation is performed quickly and sufficiently small timeout values can be selected. Even in dense networks, the nodes can thus create the required contexts within reasonable time.

Detailed results on the overall volume of transmitted data and the corresponding types are indicated in Fig. 55b, differentiating between unicast management traffic (such as CTP initialization and unicast messages employed for context establishment), broadcast messages for network management, and actual data volume transferred in the C-DEST and C-LONG modes. The figure clearly shows the prevalence of C-DEST traffic as well as the uncompressed broadcast messages accompanying the context establishment process. In total, a traffic volume of 201,606 bytes (composed of both packet headers and the 12 byte payload) has been transported through the network in this simulation. In contrast to the uncompressed volume of 271,993 bytes, even when confining the HC algorithm to the use of C-DEST, size reductions of 25.9% are observed.

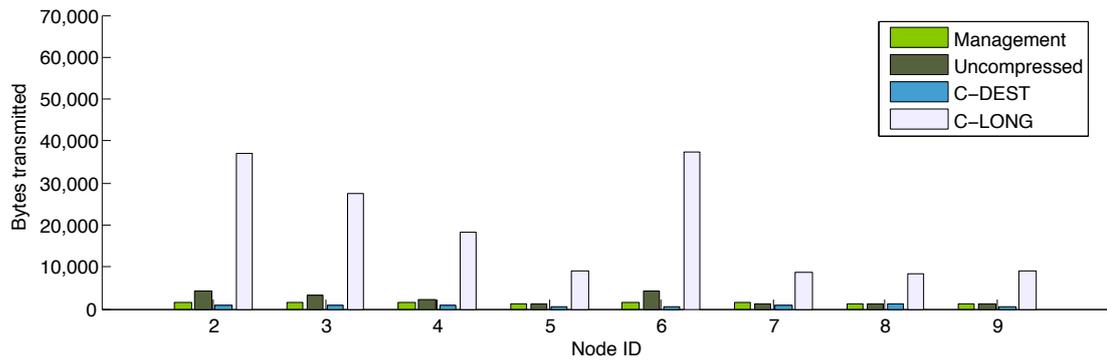
In a second simulation, we have configured all nodes to assume that they are statically located. This allows all of them to enter the C-LONG mode after the threshold of 20 successful transmissions have taken place using C-DEST. On average, SFHC.KOM requires 5,249ms to establish the C-LONG context, mostly influenced by the timeout values (cf. Timer 1 in



(a) Total number of bytes transmitted without header compression



(b) Total number of bytes transmitted in C-DEST mode



(c) Total number of bytes transmitted in C-LONG mode

Figure 55: Comparison of the total traffic volume with and without header compression

Fig. 51) at both source and destination. These timeouts, defining the maximum time allowance until neighboring nodes need to transmit their available CID vector, are set to 2,500ms in our implementation. When comparing our HC algorithm to the transmission of uncompressed packets, header size reductions of 74.7% are observed, effectively reducing the headers from 10 to 2.53 bytes on average. To allow for a direct comparison, the results are shown in Fig. 55c. Notably, the greater management overhead is counterbalanced by a smaller data volume transferred using C-LONG, clearly exposing the benefits of reduced header sizes. The traffic volume in this simulation has further reduced to a total of 191,060 bytes, i.e. a 29.8% gain over the uncompressed reference volume.

In order to assess the algorithm's behavior when a static network topology under presence of a mobile leaf node is given, we have assumed the same topology shown in Fig. 54 and marked all nodes as static, allowing them to use the C-LONG mode. An additional mobile node (labeled node 10) was inserted into the network, and assigned three different positions

Table 29: Savings when applying SFHC.KOM in a scenario with one mobile node

POSITION	TIME IN RANGE	EMITTED DATA	HC OVERHEAD	SAVINGS
I	3.9 sec	19 bytes	17 bytes	-10 bytes
II	14.3 sec	95 bytes	17 bytes	18 bytes
III	306.8 sec	2318 bytes	17 bytes	837 bytes

within the topology in successive time steps. The mobile node was placed at positions I, II, and III to establish a context with the corresponding nodes 4, 6, or 1, respectively. The results for the size savings are shown in Table 29, and indicate that already after a number of 5 packets successfully transmitted in C-DEST mode, equalling a duration of less than 15 seconds within range of the static node, savings in terms of the overall transmission sizes are gained. Due to the use of the C-DEST mode, this threshold value is also independent from the node's number of neighbors.

6.4.2 Energy Consumption

While we have shown that size reductions are possible, the corresponding energy demand has not been regarded in the above experimental evaluation. We have thus used the COOJA environment [EOF+09b] again in order to evaluate the energy demand of our solution. In the energy simulation, we have set up a simple scenario with a single sender and receiver node each. A payload size of four bytes has been used in our analysis, and we have set the packet transmission interval to two packets per second. The receiver was configured to use the NullMAC implementation, i.e. the radio transceiver was always in an active state. In contrast, we have put the sender node into its sleep mode immediately after each transmission to better observe the overall energy required to process and transmit a packet.

In the three runs of the simulation, the sender node was first configured to transmit complete packet headers. For the second run, SFHC.KOM was deployed on the node, however the device has been marked as possibly mobile, such that only the C-DEST could be used. In the third and last run, the C-LONG mode has been enabled by marking the node as statically positioned. The duration of each of the experiments was set to 100 seconds.

Our energy simulation shows that the energy required by the radio transceiver in transmission mode reduces from 13.3mJ to 11.2mJ, representing radio energy savings of 16% in the C-DEST mode. When C-LONG is used, the consumption even drops to 10.1mJ, a decrease of 24% over uncompressed transmission. At only 2.7% additional computational energy demand for C-DEST, overall energy savings of 3.5% have been observed. In case of C-LONG, the additional energy demand for computation has even reduced further to 2.3%. This results from the fact that reconstruction of the full header is performed identically in both variants, but the shorter durations for the necessary communication between MCU and radio device leads to a smaller energy requirement. As a result, the application of C-LONG has been shown reduce the platform's overall energy demand by 6.6%.

6.5 SUMMARY

We have presented the concept and design of SFHC.KOM, a stateful header compression approach for WSANs. Based on the dynamic allocation of connection identifiers, SFHC.KOM can be applied to both entirely static networks as well as WSANs with mobile device, such as present in the context of smart environments. The two underlying compression modes, C-DEST and C-LONG, allow for both compression sessions to nodes with short-lived communication sessions and to static nodes with long connection durations.

We have evaluated the compression gains of SFHC.KOM in a real-world experiment and determined that energy savings are already achieved when five or more packets are exchanged

between two nodes in C-DEST mode. Even when sender and receiver are only within reach for a shorter duration, the compression context is stored locally at both nodes, and used again during their next encounter. Our HC algorithm compiles to code sizes well within the limits of the TelosB platform, and achieves overall energy savings of up to 6.6% on a local level. By compressing packet headers in WSANs, SFHC.KOM thus complements the lossless packet payload compression approaches presented in Sec. 5. Both solutions address the imbalance between the energy demands for computation and wireless data exchange. When applied in combination, long-lived sensor and actuator networks are enabled, deemed one of the essential requirements for smart spaces.

The world is round and the place which may seem like the end may also be the beginning.

Ivy Baker Priest

7.1 SUMMARY AND CONCLUSIONS

The successful realization of smart environments requires multiple challenges to be tackled, most prominently the need for impromptu interoperability and energy-efficient communication between low-power devices with confined energy budgets [Wei93, EGo1]. Within the scope of this thesis, both of these aspects, i.e. the creation of interoperability between heterogeneous sensor and actuator nodes as well as energy-efficient networking by means of lossless data compression, have been addressed. Unified access to sensing and actuation functionality is a fundamental requirement in order to allow smart environments to act and react based upon the current contextual situation. Similarly, the need for energy-efficient operation is deemed one of the grand challenges in sensor networks research, and especially important in smart spaces, where the need for frequent maintenance of deployed devices strongly degrades the user experience. Both the demand for computational efficiency and the presence of low-power embedded hardware have been specifically regarded in our contributions in order to ensure their applicability in the envisioned application domain.

Our first contribution is the S-RPC stack, a lightweight and extensible mechanism to invoke remote methods in wireless sensor and actuator networks. Specifically designed for resource-constrained node platforms, S-RPC features a modular parameter serialization concept, allowing its configuration to the relevant data types. A lightweight data representation layer, which only adds a few bytes of overhead to all packets, caters for a correct interpretation of the exchanged messages while minimizing the overhead introduced by the use of S-RPC. Both features have been specifically designed for application on embedded sensing and actuation systems in order to ensure the real-world applicability of S-RPC. Our experimental validation has shown that S-RPC introduces only marginal delays and a very small energy overhead over the proprietary interfacing of nodes. It is thus well suited to unify the access to the heterogeneous set of embedded sensing and actuation systems envisioned to comprise the Internet of Things.

Our survey of the components on current sensor networking platforms has shown that most energy is consumed by the operation of the radio transceiver and the on-board Flash memory chip. While local storage of large amounts of data is not necessary in all applications, wireless communication represents an inherent characteristic of WSA_Ns. We have thus analyzed possibilities to preserve energy by reducing the activity phases of the radio transceiver device in order to maximize the node's operational durations. In particular, we have based our contributions on the linear relation between the transmission duration and the corresponding energy demand, and thus investigated approaches to reduce the size of radio packets. An initial feasibility study has shown that the application of data compression represents a suitable approach to preserve energy. We have evaluated a mote's energy balance when the run-length encoding algorithm is applied to real-world sensor data. Our results have shown that even if additional operations are required to encode the data, the corresponding energy demand is still easily counterbalanced by the radio energy gains resulting from the transmission of smaller packets.

Based on our observation that data compression in WSNs is feasible, we have presented the Squeeze.KOM data compression layer. In the Squeeze.KOM layer, temporal similarities in the transferred data streams are determined and efficiently compressed using a binary distance coding approach. Instead of confining the data compression component to its use in conjunction with a particular WSN application, we have instantiated an application-agnostic compression framework. Due to its transparent integration into the node's operating system, current sensor network applications can be easily adapted to reduce the size of their exchanged messages by means of data compression. Its performance on sensor data taken from real nodes has shown that energy savings of more than 10% can be achieved, even when only basic differentiation and compression mechanisms are applied to the data. Due to its lossless character, Squeeze.KOM can be used for the efficient transfer of sensor readings as well as network management and topology control messages.

As a result of our analysis of the packet characteristics present in wireless sensor networks, we have determined highly non-uniform symbol distributions in most packet payloads from environmental monitoring and body-area sensing. In the majority of the analyzed data sets, a small number of symbols are strongly represented in the input sequence, while the majority of symbols only have low occurrence frequencies. We have shown that by encoding the prevalent symbols in an efficient way, i.e. by applying adaptive Huffman coding, considerable compression gains could be achieved. Our analysis of a Huffman coder's resource demand has however confirmed its significant need for computational power and memory, rendering its application on embedded sensing systems barely possible. In order to improve the applicability of adaptive Huffman coding on wireless sensor nodes, we have presented a lightweight AHC implementation, operating on Huffman code trees with a limited number of entries. As a result, measurable reductions of the implementation's demands for computation, memory, and energy have been shown, because only prevalent symbols in the input sequence are efficiently encoded, while all other symbols are transferred uncompressed. Simulation results have indicated that even when only a small number of symbols are stored in the code tree, overall energy gains of up to almost 50% can be achieved. Finally, a supplementary real-world experiment has been conducted, confirming our simulation results.

Besides the compression of packet payloads, we have presented the stateful SFHC.KOM header compression algorithm for WSNs. It is based on lightweight mechanisms for the establishment and maintenance of compression contexts, which encapsulate constant and deterministic fields from the packet headers. After a compression context has been established, the original packet header is replaced by a reference to the context, and thus decreased in its size. SFHC.KOM has been designed to operate on both mobile and fixed nodes, and is thus directly applicable in many application scenarios. Our evaluation of SFHC.KOM has shown that despite the increased utilization of the microcontroller, energy savings of 6.6% can be achieved when both communication partners remain at constant locations, and 3.5% energy gains were observed when at least one partner was mobile. Header compression is orthogonal to the compression of packet payloads, thus it can be easily combined with the presented lossless packet payload compression algorithms.

Major contributions towards the realization of smart environments were presented in this thesis. Their combination leads to long-lived and interoperable sensor and actuator networks, which represent the foundation upon which smart environments can be established. Only by means of combining heterogeneous sensor data with versatile actuation capabilities, can the context of locations, objects, and persons be adequately captured and proactively adapted to the user's wishes and preferences. All of our contributions have been specifically designed with the resource constraints of the target platforms in mind. Furthermore, we have validated their applicability on embedded systems through analyses of their implementation footprints and energy demands based on proof-of-concept implementations. In conclusion, we have confirmed their practical applicability in wireless sensor and actuator networks for the realization of smart environments.

7.2 OUTLOOK

The contributions of this thesis represent fundamental building blocks for the energy-efficient integration of embedded sensing and actuation systems into smart environments. Through the creation of the presented solutions, a number of new research challenges and questions have emerged:

- ▷ **AUTONOMOUS ADAPTATION TO PACKET CHARACTERISTICS** We have based the presented packet payload compression approaches on our observation that symbol distributions in most WSN applications are strongly biased. In our evaluations, we have statically selected the parameters to maximize the compression gains of the underlying data sets. A dynamic adaptation to the data characteristics could however improve the compression gains further, e.g. by adjusting the tree size used for the adaptive Huffman coder in order to optimally match the properties of the underlying data.
- ▷ **CHARACTERISTICS OF THE COMMUNICATION LINK** Communication links in WSNs are often susceptible to errors and variable link qualities [SPMCo4]. A thorough analysis of the impact of packet losses on the presented mechanisms represents an issue that has only been addressed to a small extent in this thesis. Similarly, the envisioned participation of mobile entities in the WSN and the resulting node mobility can also be expected to lead to an increased number of unsuccessful packet transmissions. Even though our stateful header compression approach has been specifically adapted to the possible presence of mobile nodes, a thorough analysis based on realistic mobility patterns in smart environments remains open for further research. An extension of the proposed approaches by suitable means to encounter the impact of real radio channel characteristics and packet losses is thus necessary prior to their application in a real system.
- ▷ **NETWORK-WIDE ENERGY BUDGETS** In our analyses, we have focused on the local energy savings that can be achieved by means of data compression. On a network level, the use of smaller packets can also be expected to result in further energy savings when the messages are forwarded along a multi-hop route. A detailed analysis of the network-wide energy savings entailed by the application of data compression would be required in order to quantify the achievable savings, but has not been explicitly regarded in this thesis.
- ▷ **SECURITY AND PRIVACY** In order to provide context-aware services, smart environments generally operate in a user-centric manner, i.e. they regard the user's current context in their decisions. Maintaining the user's privacy thus represents an essential prerequisite for the wide acceptance of ubiquitous computing. Mechanisms from the field of network security, e.g. the encrypted transfer of sensitive information, are necessary to reduce the risk of unauthorized access to sensitive data and the success of replay attacks. At the same time, adequate means are required in order to maintain the users' privacy, and thus their acceptance of the smart environment. Analog to the contributions of this thesis, lightweight solutions need to be developed in order to cater for their applicability on the resource-constrained devices anticipated in smart spaces.

Finally, although Mark Weiser's concept of ubiquitous computing was presented more than 20 years ago, widely available solutions for smart environments still have not materialized to this day. Truly smart buildings are still in their infancy, and many further issues can thus be expected to arise on the way towards the fulfillment of Mark Weiser's vision.

BIBLIOGRAPHY

- [AAG⁺07] C. Alippi, G. Anastasi, C. Gelperti, F. Mancini, and M. Roveri: *Adaptive Sampling for Energy Conservation in Wireless Sensor Networks for Snow Monitoring Applications*. In *Proceedings of the 4th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 1–6, 2007.
- [AC98] R. Amirtharajah and A. P. Chandrakasan: *Self-Powered Signal Processing Using Vibration-Based Power Generation*. *IEEE Journal of Solid-State Circuits*, 33(5):687–695, 1998.
- [ACKM03] G. Alonso, F. Casati, H. Kuno, and V. Machiraju: *Web Services – Concepts, Architectures and Applications*. Springer, 2003.
- [AFWH05] E. Arens, C. C. Federspiel, D. Wang, and C. Huizenga: *How Ambient Intelligence Will Improve Habitability and Energy Efficiency in Buildings*, chapter Ambient Intelligence, pages 63–80. Springer, 2005.
- [AGAL03] T. Arici, B. Gedik, Y. Altunbasak, and L. Liu: *PINCO: A Pipelined In-Network Compression Scheme for Data Collection in Wireless Sensor Networks*. In *Proceedings of the 12th International Conference on Computer Communications and Networks (ICCCN)*, pages 539–544, 2003.
- [AGS03] V. Annamalai, S. K. S. Gupta, and L. Schwiebert: *On Tree-Based Convergecasting in Wireless Sensor Networks*. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, volume 3, pages 1942–1947, 2003.
- [AKo4] I. F. Akyildiz and I. H. Kasimoglu: *Wireless Sensor and Actor Networks: Research Challenges*. *Ad Hoc Networks Journal*, 2(4):351–367, 2004.
- [APA⁺05] J. Arango, S. Pink, S. Ali, D. Hampel, and S. Dipierro: *Header Compression for Ad-Hoc Networks*. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, volume 5, pages 3080–3086, 2005.
- [Apa08] Apache Software Foundation: *Apache Thrift*. Online: <http://thrift.apache.org>, 2008. Last access: 14 October 2011.
- [Apa09] Apache Software Foundation: *Apache Etch*. Online: <https://cwiki.apache.org/ETCH>, 2009. Last access: 14 October 2011.
- [Apa10] Apache Software Foundation: *Apache Celix*. Online: <http://incubator.apache.org/celix>, 2010. Last access: 14 October 2011.
- [APS⁺09] C. Antonopoulos, A. Prayati, T. Stoyanova, C. Koulamas, and G. Papadopoulos: *Experimental Evaluation of a WSN Platform Power Consumption*. In *Proceedings of the 23rd IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, pages 1–8, 2009.
- [ASSCo2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci: *A Survey on Sensor Networks*. *IEEE Communications Magazine*, 40(8):102–114, 2002.
- [Atm10] Atmel Corporation: *ATmega128L 8-bit Microcontroller with 128K Bytes In-System Programmable Flash*. Online: http://www.atmel.com/dyn/products/product_card.asp?part_id=2018, 2010. Last access: 14 October 2011.

- [AV10] I. F. Akyildiz and M. C. Vuran: *Wireless Sensor Networks*. John Wiley and Sons, 2010.
- [AY05] K. Akkaya and M. Younis: *Energy and QoS Aware Routing in Wireless Sensor Networks*. *Cluster Computing*, 8(2–3):179–188, 2005.
- [BA03] K. Barr and K. Asanović: *Energy Aware Lossless Data Compression*. In *Proceedings of the 1st International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 231–244, 2003.
- [BE02] D. Braginsky and D. Estrin: *Rumor Routing Algorithm For Sensor Networks*. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 22–31, 2002.
- [Bee08] Beecham Research: *Business Opportunities from Remote Device Management: Adopter Survey*. Online: <http://www.koretelematics.com/research-papers/2008BeechamResearchReport.pdf>, 2008. Last access: 14 October 2011.
- [BEH⁺06] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava: *Participatory Sensing*. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 117–134, 2006.
- [BGH⁺09] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle, and M. Yuecel: *PermaDAQ: A Scientific Instrument for Precision Sensing and Data Recovery in Environmental Extremes*. In *Proceedings of the 8th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 265–276, 2009.
- [BISV08] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli: *The Hitchhiker’s Guide to Successful Wireless Sensor Network Deployments*. In *Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 43–56, 2008.
- [BKS⁺08] T. Becker, M. Kluge, J. Schalk, T. Otterpohl, and U. Hilleringmann: *Power Management for Thermal Energy Harvesting in Aircrafts*. In *Proceedings of the 7th IEEE Conference on Sensors (SENSORS)*, pages 681–684, 2008.
- [BL11] D. Bade and W. Lamersdorf: *An Agent-Based Event Processing Middleware for Sensor Networks and RFID Systems*. *The Computer Journal*, 54(3):321–331, 2011.
- [BLTS08] A. Bamis, D. Lymberopoulos, T. Teixeira, and A. Savvides: *Towards Precision Monitoring of Elders for Providing Assistive Services*. In *Proceedings of the 1st International Conference on Pervasive Technologies Related to Assistive Environments (PETRA)*, pages 1–8, 2008.
- [BN84] A. D. Birrell and B. J. Nelson: *Implementing Remote Procedure Calls*. *ACM Transactions on Computer Systems*, 2(1):39–59, 1984.
- [BP07] E. Bergstrom and R. Pandey: *Anycast-RPC for Wireless Sensor Networks*. In *Proceedings of the 4th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 1–8, 2007.
- [BPSBoo] T. Burd, T. Pering, A. Stratakos, and R. Brodersen: *A Dynamic Voltage Scaled Microprocessor System*. In *Proceedings of IEEE International Solid-State Circuits Conference (ISSCC)*, pages 294–295 and 466, 2000.
- [Bro11] J. Brockmeier: *Gartner Adds Big Data, Gamification, and Internet of Things to Its Hype Cycle*. Online: <http://www.readwriteweb.com/enterprise/2011/08/gartner-adds-big-data-gamifica.php>, 2011. Last access: 14 October 2011.
- [BSTW86] J. L. Bentley, D. D. Sleator, R. E. Tarjan, and V. K. Wei: *A Locally Adaptive Data Compression Scheme*. *Communications of the ACM*, 29(4):320–330, 1986.

- [BTK⁺04] S. P. Beeby, M. J. Tudor, E. Koukharenko, N. M. White, T. O'Donnell, C. Saha, S. Kulkarni, and S. Roy: *Micromachined Silicon Generator for Harvesting Power from Vibrations*. In *Proceedings of the 4th International Workshop on Micro and Nanotechnology for Power Generation and Energy Conversion Applications (PowerMEMS)*, pages 104–107, 2004.
- [BTZ06] E.-O. Blass, L. Tiede, and M. Zitterbart: *An Energy-Efficient and Reliable Mechanism for Data Transport in Wireless Sensor Networks*. In *Proceedings of the 3rd International Conference on Networked Sensing Systems (INSS)*, pages 211–216, 2006.
- [BW94] M. Burrows and D. J. Wheeler: *A Block-sorting Lossless Data Compression Algorithm*. SRC Research Report SRC-RR-124, Digital Systems Research Center, 1994.
- [BWC89] T. Bell, I. H. Witten, and J. G. Cleary: *Modeling for Text Compression*. *ACM Computing Surveys*, 21(4):557–591, 1989.
- [BYAH06] M. Buettner, G. Yee, E. Anderson, and R. Han: *X-MAC: A Short Preamble MAC Protocol For Duty-Cycled Wireless Sensor Networks*. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 307–320, 2006.
- [CCMW01] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana: *Web Services Description Language (WSDL) 1.1*. Online: <http://www.w3.org/TR/wsdl.html>, 2001. Last access: 14 October 2011.
- [CDE⁺05] D. Culler, P. Dutta, C. T. Ee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, and J. Zhao: *Towards a Sensor Network Architecture: Lowering the Waistline*. In *Proceedings of the 10th Conference on Hot Topics in Operating Systems (HotOS)*, pages 139–144, 2005.
- [CEL⁺06] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. Peterson: *People-Centric Urban Sensing*. In *Proceedings of the 2nd ACM/IEEE Annual International Wireless Internet Conference (WICON)*, pages 1–14, 2006.
- [Chro0] P. Christiansson: *Knowledge Representations and Information Flow in the Intelligent Building*. In *Proceedings of the 8th International Conference on Computing in Civil and Building Engineering (ICCCBE)*, pages 1–8, 2000.
- [CIB11] S. Chiaravalloti, F. Idzikowski, and L. Budzisz: *Power Consumption of WLAN Network Elements*. Technical Report TKN-11-002, Telecommunication Networks Group, Technische Universität Berlin, 2011.
- [CKYL06] M. Chen, T. Kwon, Y. Yuan, and V. C. Leung: *Mobile Agent Based Wireless Sensor Networks*. *Journal of Computers*, 1(1):14–21, 2006.
- [Clio8] Climate Group: *The SMART 2020 Report*. Online: <http://www.smart2020.org>, 2008. Last access: 14 October 2011.
- [CMH10] D. Christin, P. S. Mogre, and M. Hollick: *Survey on Wireless Sensor Network Technologies for Industrial Automation: The Security and Quality of Service Perspectives*. *Future Internet*, 2(2), 2010.
- [CMMP06] P. Costa, L. Mottola, A. L. Murphy, and G. P. Picco: *TeenyLIME: Transiently Shared Tuple Space Middleware for Wireless Sensor Networks*. In *Proceedings of the 1st ACM International Workshop on Middleware for Sensor Networks (MIDSENS)*, pages 43–48, 2006.
- [CP09] G. Chrysos and I. Papaefstathiou: *Heavily Reducing WSN's Energy Consumption by Employing Hardware-Based Compression*. In *Proceedings of the 8th International Conference on Ad-Hoc, Mobile and Wireless Networks (ADHOC-NOW)*, pages 312–326, 2009.

- [CPR03] J. Chou, D. Petrović, and K. Ramchandran: *A Distributed and Adaptive Signal Processing Approach to Reducing Energy Consumption in Sensor Networks*. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1054–1062, 2003.
- [CPRR07] M. Cohen, T. Ponte, S. Rossetto, and N. Rodriguez: *Using Coroutines for RPC in Sensor Networks*. In *Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–8, 2007.
- [CRKH11] D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick: *A Survey on Privacy in Mobile Participatory Sensing Applications*. *Journal of Systems and Software*, 84(11):1928–1946, 2011.
- [Cro02] Crossbow Technology, Inc.: *MICA2DOT – Wireless Microsensor Mote*. Online: <http://www.cmt-gmbh.de/Mica2dot.pdf>, 2002. Last access: 14 October 2011.
- [CS09] C. Cappiello and F. Schreiber: *Quality- and Energy-aware Data Compression by Aggregation in WSN Data Streams*. In *Proceedings of the 7th Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, pages 1–6, 2009.
- [CS10] P. C. Chapin and C. Skalka: *SpartanRPC: Secure WSN Middleware for Cooperating Domains*. In *Proceedings of the 7th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 61–70, 2010.
- [CSB92] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen: *Low-Power CMOS Digital Design*. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, 1992.
- [CW08] E. J. Candes and M. B. Wakin: *An Introduction To Compressive Sampling*. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.
- [CWK⁺05] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin: *Statistical Model of Lossy Links in Wireless Sensor Networks*. In *Proceedings of the 4th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 81–88, 2005.
- [DA99] A. K. Dey and G. D. Abowd: *Towards a Better Understanding of Context and Context-Awareness*. Technical Report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology, 1999.
- [DAV04] A. Dunkels, J. Alonso, and T. Voigt: *Making TCP/IP Viable for Wireless Sensor Networks*. In *Proceedings of the 1st European Workshop on Wireless Sensor Networks (EWSN)*, 2004.
- [DBS⁺01] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J.-C. Burgelman: *Scenarios for Ambient Intelligence in 2010*. ISTAG Report, European Commission Information Society Technologies Advisory Group, 2001.
- [DCSH08] D. Diamond, S. Cole, S. Scarmagnani, and J. Hayes: *Wireless Sensor Networks and Chemo-/Biosensing*. *Chemical Reviews*, 108(2):652–679, 2008.
- [DE06] M. Ditzel and F. H. Elferink: *Low-power Radar for Wireless Sensor Networks*. In *Proceedings of the 3rd European Radar Conference (EuRAD)*, pages 139–141, 2006.
- [Deo02] S. Deorowicz: *Second Step Algorithms in the Burrows-Wheeler Compression Algorithm*. *Software Practice and Experience*, 32(2):99–111, 2002.
- [Deu96] P. Deutsch: *DEFLATE Compressed Data Format Specification Version 1.3*. Request for Comments (RFC) 1951, 1996.

- [DFo1] O. Duboisson and P. Fouquart: *ASN.1: Communication Between Heterogeneous Systems*. Morgan Kaufmann, 2001.
- [DFMFMo6] H. Dubois-Ferrière, R. Meier, L. Fabre, and P. Metrailler: *TinyNode: A Comprehensive Platform for Wireless Sensor Network Applications*. In *Proceedings of the 5th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 358–365, 2006.
- [DGVo4] A. Dunkels, B. Grönvall, and T. Voigt: *Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors*. In *Proceedings of the 29th IEEE Conference on Local Computer Networks (LCN)*, pages 455–462, 2004.
- [DHJ⁺o6] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler: *Trio: Enabling Sustainable and Scalable Outdoor Wireless Sensor Network Deployments*. In *Proceedings of the 5th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 407–415, 2006.
- [DKSo6] M. Danninger, T. Kluge, and R. Stiefelhagen: *MyConnector: Analysis of Context Cues to Predict Human Availability for Communication*. In *Proceedings of the 8th International Conference on Multimodal Interfaces (ICMI)*, pages 12–19, 2006.
- [DNHo4] H. Dai, M. Neufeld, and R. Han: *ELF: An Efficient Log-structured Flash File System for Micro Sensor Nodes*. In *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 176–187, 2004.
- [DNP99] M. Degermark, B. Nordgren, and S. Pink: *IP Header Compression*. Request for Comments (RFC) 2507, 1999.
- [Dono6] D. L. Donoho: *Compressed Sensing*. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.
- [DOTHo7] A. Dunkels, F. Österlind, N. Tsiiftes, and Z. He: *Software-based On-line Energy Estimation for Sensor Nodes*. In *Proceedings of the 4th Workshop on Embedded Networked Sensors (EmNets)*, pages 28–32, 2007.
- [DWBPO1] L. Doherty, B. A. Warneke, B. Boser, and K. S. J. Pister: *Energy and Performance Considerations for Smart Dust*. *International Journal of Parallel and Distributed Systems and Networks*, 4(3):121–133, 2001.
- [EAR⁺o6] E. Ertin, A. Arora, R. Ramnath, M. Nesterenko, V. Naik, S. Bapat, V. Kulathurmani, M. Sridharan, H. Zhang, and H. Cao: *Kansei: A Testbed for Sensing at Scale*. In *Proceedings of the 5th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 399–406, 2006.
- [EBWo2] J.-P. Ebert, B. Burns, and A. Wolisz: *A Trace-based Approach for Determining the Energy Consumption of a WLAN Network Interface*. In *Proceedings of the European Wireless Conference*, pages 230–236, 2002.
- [EGo1] W. K. Edwards and R. E. Grinter: *At Home with Ubiquitous Computing: Seven Challenges*. In *Proceedings of the 3rd International Conference on Ubiquitous Computing (UbiComp)*, pages 256–272, 2001.
- [EGBMo9] N. El Rachkidy, A. Guitton, B. Bakhache, and M. Misson: *PiRAT: Pivot Routing for Alarm Transmission in Wireless Sensor Networks*. In *Proceedings of the 34th IEEE Conference on Local Computer Networks (LCN)*, pages 85–92, 2009.
- [EGHoo] D. Estrin, R. Govindan, and J. Heidemann: *Embedding the Internet*. *Communications of the ACM*, 43(5):38–41, 2000.

- [EGPS01] D. Estrin, L. Girod, G. Pottie, and M. Srivastava: *Instrumenting the World with Wireless Sensor Networks*. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 4, pages 2033–2036, 2001.
- [EHD04] A. El-Hoiydi and J.-D. Decotignie: *WiseMAC: An Ultra Low Power MAC Protocol for the Downlink of Infrastructure Wireless Sensor Networks*. In *Proceedings of the 9th IEEE Symposium on Computers and Communication (ISCC)*, pages 244–251, 2004.
- [EOF⁺09a] J. Eriksson, F. Österlind, N. Finne, A. Dunkels, N. Tsiftes, and T. Voigt: *Accurate Network-Scale Power Profiling for Sensor Network Simulators*. In *Proceedings of the 6th European Conference on Wireless Sensor Networks (EWSN)*, pages 312–326, 2009.
- [EOF⁺09b] J. Eriksson, F. Österlind, N. Finne, N. Tsiftes, A. Dunkels, T. Voigt, R. Sauter, and P. J. Marrón: *COOJA/MSPsim: Interoperability Testing for Wireless Sensor Networks*. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques (SimuTools)*, pages 1–7, 2009.
- [ETH05] ETH Zürich: *BTNodes – A Distributed Environment for Prototyping Ad Hoc Networks*. Online: <http://www.btnode.ethz.ch>, 2005. Last access: 14 October 2011.
- [FDC84] D. Farber, G. Delp, and T. Conte: *Thinwire Protocol for Connecting Personal Computers to the Internet*. Request for Comments (RFC) 914, 1984.
- [FRWZ07] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi: *In-network Aggregation Techniques for Wireless Sensor Networks: A Survey*. *IEEE Wireless Communications*, 14(2):70–87, 2007.
- [GFJ⁺09] 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)*, pages 1–14, 2009.
- [GHH⁺02] R. Govindan, J. M. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker: *The Sensor Network as a Database*. Technical Report 02-771, USC/Information Sciences Institute, 2002.
- [GHR⁺05] B. Gyselinckx, C. V. Hoof, J. Ryckaert, R. F. Yazicioglu, P. Fiorini, and V. Leonov: *Human++: Autonomous Wireless Sensors for Body Area Networks*. In *Proceedings of the IEEE Custom Integrated Circuits Conference (CICC)*, pages 13–19, 2005.
- [GJAS06] R. K. Ganti, P. Jayachandran, T. Abdelzaher, and J. A. Stankovic: *SATIRE: A Software Architecture for Smart AtTIRE*. In *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services (MobiSys)*, pages 110–123, 2006.
- [Gol66] S. W. Golomb: *Run-Length Encoding*. *IEEE Transactions on Information Theory*, 12:399–401, 1966.
- [Goo09] Google Inc.: *Protocol Buffers*. Online: <http://code.google.com/p/protobuf/>, 2009. Last access: 14 October 2011.
- [Gör05] M. Görtz: *Effiziente Echtzeit-Kommunikationsdienste durch Einbeziehung von Kontexten*. Dissertation, Technische Universität Darmstadt, Fachgebiet Multimedia Kommunikation, 2005.
- [Gra53] F. Gray: *Pulse Code Communication*. U.S. Patent 2,632,058, filed 13 November 1947, issued 17 March 1953.

- [GTHo8] A. Guitton, N. Trigoni, and S. Helmer: *Fault-Tolerant Compression Algorithms for Delay-Sensitive Sensor Networks with Unreliable Links*. In *Proceedings of the 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 190–203, 2008.
- [Ham50] R. W. Hamming: *Error Detecting and Error Correcting Codes*. The Bell System Technical Journal, 26(2):147–160, 1950.
- [Hau05] S. Haustein: *kXML v2*. Online: <http://kxml.sf.net/>, 2005. Last access: 14 October 2011.
- [HB09] P. Hurni and T. Braun: *Calibrating Wireless Sensor Network Simulation Models with Real-World Experiments*. In *Proceedings of the 8th International IFIP-TC 6 Networking Conference (NETWORKING)*, pages 1–13, 2009.
- [HC02] J. L. Hill and D. E. Culler: *Mica: A Wireless Platform for Deeply Embedded Networks*. IEEE Micro, 22(6):12–24, 2002.
- [HC08] J. W. Hui and D. E. Culler: *IP is Dead, Long Live IP for Wireless Sensor Networks*. In *Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 15–28, 2008.
- [HC10] J. W. Hui and D. E. Culler: *IPv6 in Low-Power Wireless Networks*. Proceedings of the IEEE, 98(11):1865–1878, 2010.
- [Hen07] B. Henderson: *XML-RPC Introspection*. Online: <http://xmlrpc-c.sourceforge.net/introspection.html>, 2007. Last access: 14 October 2011.
- [HKHo1] C.-H. Hsu, U. Kremer, and M. Hsiao: *Compiler-directed Dynamic Voltage/Frequency Scheduling for Energy Reduction in Microprocessors*. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, pages 275–278, 2001.
- [HKS⁺06] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, G. Zhou, J. Hui, and B. Krogh: *VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance*. ACM Transactions on Sensor Networks, 2(1):1–38, 2006.
- [HKWWo6] V. Handziski, A. Köpke, A. Willig, and A. Wolisz: *TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Networks*. In *Proceedings of the 2nd International Workshop on Multi-hop Ad Hoc Networks: From Theory to Reality (RealMAN)*, pages 63–70, 2006.
- [HL05] V. Handziski and T. Lentsch: *The eyesIFX Platform*. TinyOS Technology Exchange II, 2005.
- [HL07] F. Huang and Y. Liang: *Towards Energy Optimization in Environmental Wireless Sensor Networks for Lossless and Reliable Data Gathering*. In *Proceedings of the 4th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 1–6, 2007.
- [HM99] R. Hakenes and Y. Manoli: *A Segmented Gray Code for Low-Power Microcontroller Address Buses*. In *Proceedings of the 25th Euromicro Conference (EUROMICRO)*, volume 1, pages 240–243, 1999.
- [HMo6] J. K. Hart and K. Martinez: *Environmental Sensor Networks: A Revolution in the Earth System Science?* Earth-Science Reviews, 78(3–4):177–191, 2006.
- [HPH⁺05] V. Handziski, J. Polastre, J.-H. Hauer, C. Sharp, A. Wolisz, and D. E. Culler: *Flexible Hardware Abstraction for Wireless Sensor Networks*. In *Proceedings of the 2nd European Workshop on Wireless Sensor Networks (EWSN)*, pages 145–157, 2005.

- [HRN⁺10] N. Hoeller, C. Reinke, J. Neumann, S. Groppe, M. Lipphardt, B. Schuett, and V. Linnemann: *Stream-Based XML Template Compression for Wireless Sensor Network Data Management*. In *Proceedings of the 4th International Conference on Multimedia and Ubiquitous Engineering (MUE)*, pages 1–9, 2010.
- [HSI⁺01] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan: *Building Efficient Wireless Sensor Networks with Low-Level Naming*. *SIGOPS Operating Systems Review*, 35(5):146–159, 2001.
- [HSW⁺00] J. L. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister: *System Architecture Directions for Networked Sensors*. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, pages 93–104, 2000.
- [HTCP09] D. C. Hoang, Y. K. Tan, H. B. Chng, and S. K. Panda: *Thermal Energy Harvesting From Human Warmth For Wireless Body Area Network In Medical Healthcare System*. In *Proceedings of the 8th International Conference on Power Electronics and Drive Systems (PEDS)*, pages 1277–1282, 2009.
- [Huf52] D. A. Huffman: *A Method for the Construction of Minimum-Redundancy Codes*. *Proceedings of the IRE*, 40(9):1098–1101, 1952.
- [IEE06] IEEE Standards Association: *IEEE 802.15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-rate Wireless Personal Area Networks (LR-WPANs)*. Online: <http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf>, 2006. Last access: 14 October 2011.
- [IGE00] C. Intanagonwiwat, R. Govindan, and D. Estrin: *Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks*. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 56–67, 2000.
- [Int00] Interlaboratory Working Group: *Scenarios for a Clean Energy Future*. ORNL/CON-476 (Oak Ridge National Laboratory) and LBNL-44029 (Lawrence Berkeley National Laboratory), 2000.
- [Into5a] Intel Corporation: *Intel Pro/Wireless 2200BG Network Connection*. Online: http://www.dell.com/downloads/us/products/latit/intel2200_spec.pdf, 2005. Last access: 14 October 2011.
- [Into5b] International Telecommunication Union: *ITU Internet Reports 2005: The Internet of Things*, 2005.
- [Into8] International Telecommunication Union – Radiocommunication Sector: *Radio Regulation Recommendations 5.138 and 5.150*, 2008.
- [IPS11] IPSO Alliance: *Enabling the Internet of Things*. Online: <http://ipso-alliance.org>, 2011. Last access: 14 October 2011.
- [Jac90] V. Jacobson: *Compressing TCP/IP Headers for Low-Speed Serial Links*. Request for Comments (RFC) 1144, 1990.
- [JBL07] R. Jurdak, P. Baldi, and C. V. Lopes: *Adaptive Low Power Listening for Wireless Sensor Networks*. *IEEE Transactions on Mobile Computing*, 6(8):1–14, 2007.
- [JC05] H. Ju and L. Cui: *EasiPC: A Packet Compression Mechanism for Embedded WSN*. In *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 394–399, 2005.

- [JML10] R. Jedermann, A. Moehrke, and W. Lang: *Supervision of Banana Transports by the Intelligent Container*. In *Proceedings of the 4th International Workshop on Coolchain-Management*, pages 75–84, 2010.
- [JOW⁺02] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein: *Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with Zebrantet*. In *Proceedings of the 10th Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 96–107, 2002.
- [JSO10] JSON-RPC Working Group: *JSON-RPC 2.0 – A Lightweight Remote Procedure Call Protocol*. Online: <http://www.jsonrpc.org/>, 2010. Last access: 14 October 2011.
- [KAB⁺05] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis: *Design and Deployment of Industrial Sensor Networks: Experiences from a Semiconductor Plant and the North Sea*. In *Proceedings of the 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 64–75, 2005.
- [KEW02] B. Krishnamachari, D. Estrin, and S. B. Wicker: *The Impact of Data Aggregation in Wireless Sensor Networks*. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS)*, pages 575–578, 2002.
- [KGKS05] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker: *Geographic Routing Made Practical*. In *Proceedings of the 2nd USENIX/ACM Symposium on Networked System Design and Implementation (NSDI)*, pages 217–230, 2005.
- [KKPG98] J. Kymissis, C. Kendall, J. Paradiso, and N. Gershenfeld: *Parasitic Power Harvesting in Shoes*. In *Proceedings of the 2nd IEEE International Symposium on Wearable Computers (ISWC)*, pages 132–139, 1998.
- [KL05] N. Kimura and S. Latifi: *A Survey on Data Compression in Wireless Sensor Networks*. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC)*, volume 2, pages 8–13, 2005.
- [KRV⁺08] J. Kwong, Y. Ramadass, N. Verma, M. Koesler, K. Huber, H. Moormann, and A. Chandrakasan: *A 65nm Sub-V Microcontroller with Integrated SRAM and Switched-Capacitor DC-DC Converter*. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, pages 318–319 and 616, 2008.
- [KS04] M. Kaddoura and S. Schneider: *SEEHOC: Scalable and Robust End-to-End Header Compression Techniques for Wireless Ad Hoc Networks*. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, pages 141–146, 2004.
- [KSS02] G. Kulkarni, C. Schurgers, and M. Srivastava: *Dynamic Link Labels for Energy Efficient MAC Headers in Wireless Sensor Networks*. In *Proceedings of the 1st IEEE International Conference on Sensors (SENSORS)*, volume 2, pages 1520–1525, 2002.
- [KW05] H. Karl and A. Willig: *Protocols and Architectures for Wireless Sensor Networks*. John Wiley and Sons, 2005.
- [KW08a] A. Köpke and A. Wolisz: *Measuring the Node Energy Consumption in USB based WSN Testbeds*. In *Proceedings of the 28th International Conference on Distributed Computing Systems Workshops (ICDCS)*, pages 333–338, 2008.
- [KW08b] A. Köpke and A. Wolisz: *What's new? Message Reduction in Sensor Networks using Events*. In *Proceedings of the 3rd IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, pages 756–761, 2008.

- [LBV06] K. Langendoen, A. Baggio, and O. Visser: *Murphy Loves Potatoes – Experiences from a Pilot Sensor Network Deployment in Precision Agriculture*. In *Proceedings of the 14th International Workshop on Parallel and Distributed Real-Time Systems (IPDPS)*, pages 1–8, 2006.
- [LHCJ09] T. Le, W. Hu, P. Corke, and S. Jha: *ERTP: Energy-efficient and Reliable Transport Protocol for Data Streaming in Wireless Sensor Networks*. *Computer Communications*, 32(7–10):1154–1171, 2009.
- [LLS06] D. Lymberopoulos, Q. Lindsey, and A. Savvides: *An Empirical Characterization of Radio Signal Strength Variability in 3-D IEEE 802.15.4 Networks Using Monopole Antennas*. In *Proceedings of the 3rd European Workshop on Wireless Sensor Networks (EWSN)*, pages 326–341, 2006.
- [LM10] K. Langendoen and A. Meier: *Analyzing MAC Protocols for Low Data-Rate Applications*. *ACM Transactions on Sensor Networks*, 7(1):1–34, 2010.
- [LMG⁺04] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler: *The Emergence of Networking Abstractions and Techniques in TinyOS*. In *Proceedings of the 1st USENIX/ACM Symposium on Networked System Design and Implementation (NSDI)*, pages 1–14, 2004.
- [Lor05] M. O. Lorenz: *Methods of Measuring the Concentration of Wealth*. *Publications of the American Statistical Association*, 9(70):209–219, 1905.
- [LP10] Y. Liang and W. Peng: *Minimizing Energy Consumptions in Wireless Sensor Networks via Two-Modal Transmission*. *SIGCOMM Computer Communication Review*, 40(1):12–18, 2010.
- [LR01] D. Linden and T. B. Reddy (editors): *Handbook of Batteries*. McGraw-Hill Professional, 2001.
- [LRW04] 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 International Conference on Communications (ICC)*, pages 3769–3776, 2004.
- [LTS07] D. Lymberopoulos, T. Teixeira, and A. Savvides: *Detecting Patterns for Assisted Living Using Sensor Networks*. In *Proceedings of the International Conference on Sensor Technologies and Applications (SENSORCOMM)*, pages 590–596, 2007.
- [LY02] K. Lyytinen and Y. Yoo: *Issues and Challenges in Ubiquitous Computing*. *Communications of the ACM*, 45(12):62–65, 2002.
- [LYC03] S.-Y. R. Li, R. W. Yeung, and N. Cai: *Linear Network Coding*. *IEEE Transactions on Information Theory*, 49(2):371–381, 2003.
- [LYLL03] J. M. H. Lee, S. C. L. Yuen, W. J. Li, and P. H. W. Leong: *Development of an AA Size Energy Transducer with Micro Resonators*. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, volume 4, pages 876–879, 2003.
- [Man08] R. Manson: *SOAPjr*. Online: <http://soapjr.org/>, 2008. Last access: 14 October 2011.
- [Mar79] G. Martin: *Range Encoding: An Algorithm for Removing Redundancy from a Digitised Message*. In *Proceedings of the IERE Conference on Video and Data Recording*, pages 24–27, 1979.
- [MCP⁺02] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson: *Wireless Sensor Networks for Habitat Monitoring*. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 88–97, 2002.

- [MDDHo6] T. D. May, S. H. Dunning, G. A. Dowding, and J. O. Hallstrom: *An RPC Design for Wireless Sensor Networks*. International Journal of Pervasive Computing and Communications, 2(4):384–397, 2006.
- [Meio9] A. Meier: *Safety-Critical Wireless Sensor Networks*. Dissertation, ETH Zürich, 2009.
- [MEMo5] MEMSIC Inc.: *MCS410 – Cricket Wireless Location System*. Online: <http://memic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=140:mcs410-cricket>, 2005. Last access: 14 October 2011.
- [MEMo6a] MEMSIC Inc.: *Imote2 – High-performance Wireless Sensor Network Node*. Online: <http://www.memic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=139:imote2-multimedia>, 2006. Last access: 14 October 2011.
- [MEMo6b] MEMSIC Inc.: *TELOSB Mote Platform*. Online: <http://memic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=152:telosb>, 2006. Last access: 14 October 2011.
- [MEMo7] MEMSIC Inc.: *MICAz – Wireless Measurement System*. Online: <http://memic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=148:micaz>, 2007. Last access: 14 October 2011.
- [MFHHo2] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong: *TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks*. ACM SIGOPS Operating Systems Review, 36(Special Issue):131–146, 2002.
- [MHLCo7] D. Moss, J. Hui, P. Levis, and J. I. Choi: *TinyOS Extension Proposal 126: CC2420 Radio Stack*. Online: <http://www.tinyos.net/tinyos-2.x/doc/html/tep126.html>, 2007. Last access: 14 October 2011.
- [Mico7] Microchip Technology Inc.: *MRF24J40: IEEE 802.15.4 2.4GHz RF Transceiver*. Online: <http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en027752>, 2007. Last access: 14 October 2011.
- [MITo3] MIT Technology Review: *10 Emerging Technologies That Will Change the World*. Online: <http://www.technologyreview.com/Infotech/13060/>, 2003. Last access: 14 October 2011.
- [Mito5] R. L. Mitchell: *The Rise of Smart Buildings*. Computerworld Online: http://www.computerworld.com/s/article/100318/The_Rise_of_Smart_Buildings, 2005. Last access: 14 October 2011.
- [MKHCo7] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler: *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. Request for Comments (RFC) 4944, 2007.
- [MLo7] N. Mitra and Y. Lafon: *Simple Object Access Protocol (SOAP) 1.2*. Online: <http://www.w3.org/TR/soap/>, 2007. Last access: 14 October 2011.
- [Mog10] P. S. Mogre: *Cross-Layer Bandwidth Management and Optimization in TDMA Based Wireless Mesh Networks using Network Coding*. PhD thesis, Technische Universität Darmstadt, Fachgebiet Multimedia Kommunikation, 2010.
- [MOHo4] K. Martinez, R. Ong, and J. Hart: *Glacsweb: A Sensor Network for Hostile Environments*. In *Proceedings of the 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, pages 81–87, 2004.

- [MPSHH08] R. Marin-Perianu, J. Scholten, P. Havinga, and P. Hartel: *Cluster-based Service Discovery for Heterogeneous Wireless Sensor Networks*. *International Journal of Parallel, Emergent and Distributed Systems*, 23(4):325–346, 2008.
- [MR03] S. Meyer and A. Rakotonirainy: *A Survey of Research on Context-aware Homes*. In *Proceedings of the Australasian Information Security Workshop Conference on ACSW Frontiers (AISW)*, pages 159–168, 2003.
- [MSG⁺05] E. D. Marsman, R. M. Senger, M. R. Guthaus, R. A. Ravindran, G. S. Dasika, and S. A. Mahlke: *A 16-Bit Low-Power Microcontroller with Monolithic MEMS-LC Clocking*. In *Proceedings of the International Conference on Circuits and Systems (ISCAS)*, pages 624–627, 2005.
- [MTSG10] G. Moritz, D. Timmermann, R. Stoll, and F. Golatowski: *Encoding and Compression for the Devices Profile for Web Services*. In *Proceedings of the 5th IEEE International Workshop on Service Oriented Architectures in Converging Networked Environments (SOCNE)*, pages 514–519, 2010.
- [MV08] F. Marcelloni and M. Vecchio: *A Simple Algorithm for Data Compression in Wireless Sensor Networks*. *IEEE Communication Letters*, 12(6):411–413, 2008.
- [MV09] F. Marcelloni and M. Vecchio: *An Efficient Lossless Compression Algorithm for Tiny Nodes of Monitoring Wireless Sensor Networks*. *The Computer Journal*, 52(8):969–987, 2009.
- [MWN02] S. McLean, K. Williams, and J. Naftel: *Microsoft .Net Remoting*. Microsoft Press, 2002.
- [MWW11] T. Menzel, D. Willkomm, and A. M. Wolisz: *Improving Battery-Efficiency of Embedded Devices by Favorably Discharging only towards End-of-life*. In *Proceedings of the 2nd International Workshop on Networks of Cooperating Objects (CONET)*, pages 1–11, 2011.
- [Nel98] G. J. Nelson: *Context-Aware and Location Systems*. PhD thesis, University of Cambridge, 1998.
- [NFPG11] H. A. Nguyen, A. Förster, D. Puccinelli, and S. Giordano: *Sensor Node Lifetime: An Experimental Study*. In *Workshop Proceedings of the 9th Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, pages 146–151, 2011.
- [OBAA05] Özgür B. Akan and I. F. Akyildiz: *Event-to-Sink Reliable Transport in Wireless Sensor Networks*. *IEEE/ACM Transactions on Networking*, 13(5):1003–1016, 2005.
- [Obj02] Object Management Group: *Interface Definition Language (IDL) Syntax and Semantics*. Online: <http://www.omg.org/cgi-bin/doc?formal/02-06-39.pdf>, 2002. Last access: 14 October 2011.
- [ODL09] K. Ota, M. Dong, and X. Li: *TinyBee: Mobile-Agent-Based Data Gathering System in Wireless Sensor Networks*. In *Proceedings of the IEEE International Conference on Networking, Architecture, and Storage (NAS)*, pages 24–31, 2009.
- [OT09] P. C. Ölveczky and S. Thorvaldsen: *Formal Modeling, Performance Estimation, and Model Checking of Wireless Sensor Network Algorithms in Real-Time Maude*. *Theoretical Computer Science*, 410(2–3):254–280, 2009.
- [Oxf11] Oxford Dictionaries: *Definition of “mote”*. Online: <http://oxforddictionaries.com/definition/mote>, 2011. Last access: 14 October 2011.

- [Pav11] I. Pavlov: *7-Zip – A File Archiver with a High Compression Ratio*. Online: <http://www.7-zip.org/>, 2011. Last access: 14 October 2011.
- [PCo4] C. Park and P. H. Chou: *Power Utility Maximization for Multiple-Supply Systems by a Load-Matching Switch*. In *Proceedings of the International Symposium on Low Power Electronic Design (ISLPED)*, pages 168–173, 2004.
- [Pen99] A. Pentland: *Perceptual Intelligence*. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC)*, pages 74–88, 1999.
- [PGN⁺03] C. Pereira, S. Gupta, K. Niyogi, I. Lazaridis, S. Mehrotra, and R. Gupta: *Energy Efficient Communication for Reliability and Quality Aware Sensor Networks*. Technical Report TR 03-15, University of California at Irvine, 2003.
- [PGP⁺00] E. M. Petriu, N. D. Georganas, D. C. Petriu, D. Makrakis, and V. Z. Groza: *Sensor-Based Information Appliances*. *IEEE Instrumentation and Measurement Magazine*, 3(4):31–35, 2000.
- [PHCo4] J. Polastre, J. Hill, and D. Culler: *Versatile Low Power Media Access for Wireless Sensor Networks*. In *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 95–107, 2004.
- [PKoo] G. J. Pottie and W. J. Kaiser: *Wireless Integrated Network Sensors*. *Communications of the ACM*, 43(5):51–58, 2000.
- [PKG04] S. Patten, B. Krishnamachari, and R. Govindan: *The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks*. In *Proceedings of the 3rd ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 28–35, 2004.
- [PKGZ08] N. B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao: *Tiny Web Services: Design and Implementation of Interoperable and Evolvable Sensor Networks*. In *Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 253–266, 2008.
- [PKRo2] S. S. Pradhan, J. Kusuma, and K. Ramchandran: *Distributed Compression in a Dense Microsensor Network*. *IEEE Signal Processing Magazine*, 19(2):51–60, 2002.
- [Pos80] J. Postel: *User Datagram Protocol*. Request for Comments (RFC) 768, 1980.
- [Pos81] J. Postel: *Transmission Control Protocol*. Request for Comments (RFC) 793, 1981.
- [PSC05] J. Polastre, R. Szewczyk, and D. Culler: *Telos: Enabling Ultra-Low Power Wireless Research*. In *Proceedings of the 4th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 364–369, 2005.
- [PWH⁺07] D. Pfisterer, M. Wegner, H. Hellbrück, C. Werner, and S. Fischer: *Energy-optimized Data Serialization For Heterogeneous WSNs Using Middleware Synthesis*. In *Proceedings of the 6th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pages 180–187, 2007.
- [RAR07] J. S. Rellermeier, G. Alonso, and T. Roscoe: *R-OSGi: Distributed Applications through Software Modularization*. In *Proceedings of the 8th ACM/IFIP/USENIX International Conference on Middleware (Middleware)*, pages 1–20, 2007.
- [RBM⁺11] A. Reinhardt, D. Burkhardt, P. S. Mogre, M. Zaheer, and R. Steinmetz: *Smart-Meter.KOM: A Low-cost Wireless Sensor for Distributed Power Metering (accepted for publication)*. In *Proceedings of the 6th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, pages 1044–1051, 2011.

- [RCH⁺10] A. Reinhardt, D. Christin, M. Hollick, J. Schmitt, P. Mogre, and R. Steinmetz: *Trimming the Tree: Tailoring Adaptive Huffman Coding to Wireless Sensor Networks*. In *Proceedings of the 7th European Conference on Wireless Sensor Networks (EWSN)*, pages 33–48, 2010.
- [RCHSo9] A. Reinhardt, D. Christin, M. Hollick, and R. Steinmetz: *On the Energy Efficiency of Lossless Data Compression in Wireless Sensor Networks*. In *Proceedings of the 4th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, pages 873–880, 2009.
- [REo7] A. Rezgui and M. Eltoweissy: *Service-Oriented Sensor-Actuator Networks*. *IEEE Communications Magazine*, 45(12):92–100, 2007.
- [RGSo6] V. Raghunathan, S. Ganeriwal, and M. Srivastava: *Emerging Techniques for Long Lived Wireless Sensor Networks*. *IEEE Communications Magazine*, 44(4):108–114, 2006.
- [RHC10] R. Rana, W. Hu, and C. T. Chou: *Energy-Aware Sparse Approximation Technique (EAST) for Rechargeable Wireless Sensor Networks*. In *Proceedings of the 7th European Conference on Wireless Sensor Networks (EWSN)*, pages 306–321, 2010.
- [RHSo9] A. Reinhardt, M. Hollick, and R. Steinmetz: *Stream-oriented Lossless Packet Compression in Wireless Sensor Networks*. In *Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 1–9, 2009.
- [RHSNo6] M. Rabbat, J. Haupt, A. Singh, and R. Nowak: *Decentralized Compression and Predistribution via Randomized Gossiping*. In *Proceedings of the 5th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 51–59, 2006.
- [RKH⁺05] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava: *Design Considerations for Solar Energy Harvesting Wireless Embedded Systems*. In *Proceedings of the 4th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 457–462, 2005.
- [RKHSo8] A. Reinhardt, M. Kropff, M. Hollick, and R. Steinmetz: *Designing a Sensor Network Testbed for Smart Heterogeneous Applications*. In *Proceedings of the 3rd IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, pages 715–722, 2008.
- [RMKS10] A. Reinhardt, P. Mogre, T. Koenig, and R. Steinmetz: *SFHC.KOM: Stateful Header Compression for Wireless Sensor Networks*. In *Proceedings of the 35th IEEE Conference on Local Computer Networks (LCN)*, pages 264–271, 2010.
- [RMS11] A. Reinhardt, P. Mogre, and R. Steinmetz: *Lightweight Remote Procedure Calls for Wireless Sensor and Actuator Networks*. In *Workshop Proceedings of the 9th Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, pages 116–121, 2011.
- [RMW⁺97] M. D. Rowe, G. Min, S. G. K. Williams, A. Aoune, K. Matsuura, V. L. Kuznetsov, and L. W. Fu: *Thermoelectric Recovery of Waste Heat – Case Studies*. In *Proceedings of the 32nd Intersociety Energy Conversion Engineering Conference (IECEC)*, volume 2, pages 1075–1079, 1997.
- [Rouo3] S. J. Roundy: *Energy Scavenging for Wireless Sensor Nodes with a Focus on Vibration to Electricity Conversion*. PhD thesis, The University of California, Berkeley, 2003.

- [RP71] R. F. Rice and J. R. Plaunt: *Adaptive Variable-Length Coding for Efficient Compression of Spacecraft Television Data*. IEEE Transactions on Communication Technology, 19(6):889–897, 1971.
- [RR07a] L. Richardson and S. Ruby: *RESTful Web Services*. O’Reilly Media, 2007.
- [Sal07] D. Salomon: *Data Compression: The Complete Reference*. Springer, 2007.
- [SAT⁺99] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, and W. Van de Velde: *Advanced Interaction in Context*. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC)*, pages 89–101, 1999.
- [SAW94] B. N. Schilit, N. Adams, and R. Want: *Context-Aware Computing Applications*. In *Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 85–90, 1994.
- [SBC⁺05] B. Schott, M. Bajura, J. Czarnaski, J. Flidr, T. Tho, and L. Wang: *A Modular Power-aware Microsensor with 1000x Dynamic Power Range*. In *Proceedings of the 4th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 469–474, 2005.
- [SBR10] P. Suriyachai, J. Brown, and U. Roedig: *Time-Critical Data Delivery in Wireless Sensor Networks*. In *Proceedings of the 6th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 216–229, 2010.
- [Sch09] J. Schmitt: *Anpassungsfähige Kontextbestimmung zur Unterstützung von Kommunikationsdiensten*. PhD thesis, Technische Universität Darmstadt, Fachgebiet Multimedia Kommunikation, 2009.
- [Sew11] J. Seward: *bzip2 – A Freely Available, Patent Free, High-Quality Data Compressor*. Online: <http://www.bzip.org/>, 2011. Last access: 14 October 2011.
- [SGJ08] Y. Sun, O. Gurewitz, and D. B. Johnson: *RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks*. In *Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 1–14, 2008.
- [SGO⁺04] T. Schoellhammer, B. Greenstein, E. Osterweil, M. Wimbrow, and D. Estrin: *Lightweight Temporal Compression of Microclimate Datasets*. In *Proceedings of the 29th IEEE Conference on Local Computer Networks (LCN)*, pages 516–524, 2004.
- [Sha48] C. E. Shannon: *A Mathematical Theory of Communication*. Bell System Technical Journal, 27:379–423 and 623–656, 1948.
- [SHB10] P. Serrano, M. Hollick, and A. Banchs: *On the Trade-Off between Throughput Maximization and Energy Consumption Minimization in IEEE 802.11 WLANs*. Journal of Communications and Networks, 12(2):150–157, 2010.
- [She10] Z. Shelby: *Embedded Web Services*. IEEE Wireless Communications, 17(6):52–57, 2010.
- [Shi10] Shimmer Research: *Shimmer – Wireless Sensor Platform for Wearable Applications*. Online: <http://www.shimmer-research.com>, 2010. Last access: 14 October 2011.
- [SHL⁺08] M. Seok, S. Hanson, Y.-S. Lin, Z. Foo, D. Kim, Y. Lee, N. Liu, D. Sylvester, and D. Blaauw: *The Phoenix Processor: A 30pW Platform for Sensor Applications*. In *Proceedings of the IEEE Symposium on VLSI Circuits (VLSIC)*, pages 188–189, 2008.

- [Sieo1] M. Siegel: *Sensor Modeling and Simulation: Can it Pass the Turing Test?* In *Proceedings of the IEEE International Workshop on Virtual and Intelligent Measurement Systems (VIMS)*, pages 92–96, 2001.
- [SMo6] C. M. Sadler and M. Martonosi: *Data Compression Algorithms for Energy-Constrained Devices in Delay Tolerant Networks*. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 265–278, 2006.
- [SN04] R. Steinmetz and K. Nahrstedt: *Multimedia Systems*. Springer, 2004.
- [Soh09] K. Soh: *Is it the End of the Desktop PC?* Reuters Online: <http://www.reuters.com/article/2009/01/07/us-laptops-idUSTRE50601320090107>, 2009. Last access: 14 October 2011.
- [SPo1] N. S. Shenck and J. A. Paradiso: *Energy Scavenging with Shoe-Mounted Piezoelectronics*. *IEEE Micro*, 21(3):30–42, 2001.
- [SPJ10] K. Sandlund, G. Pelletier, and L.-E. Jonsson: *The RObust Header Compression (ROHC) Framework*. Request for Comments (RFC) 5795, 2010.
- [SPMC04] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler: *Lessons from a Sensor Network Expedition*. In *Proceedings of the 1st European Workshop on Wireless Sensor Networks (EWSN)*, pages 307–322, 2004.
- [SRMFo6] P. Seeling, M. Reisslein, T. K. Madsen, and F. H. Fitzek: *Performance Analysis of Header Compression Schemes in Heterogeneous Wireless Multi-Hop Networks*. *Wireless Personal Communications*, 38(2):203–232, 2006.
- [SSo1] C. Schurgers and M. B. Srivastava: *Energy Efficient Routing in Wireless Sensor Networks*. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, pages 357–361, 2001.
- [Sta96] T. Starner: *Human-powered Wearable Computing*. *IBM Systems Journal*, 35(3–4):1–12, 1996.
- [Sun87] Sun Microsystems: *XDR: External Data Representation Standard*. Request for Comments (RFC) 1014, 1987.
- [Sun06] Sun Microsystems, Inc.: *Java Remote Method Invocation - Distributed Computing for Java*. Online: <http://java.sun.com/javase/technologies/core/basic/rmi/whitepaper>, 2006. Last access: 14 October 2011.
- [Sun08] Sun Microsystems Inc.: *Project SunSPOT - Sun Small Programmable Object Technology*. Online: <http://www.sunspotworld.com>, 2008. Last access: 14 October 2011.
- [TDHV09] N. Tsiftes, A. Dunkels, Z. He, and T. Voigt: *Enabling Large-Scale Storage in Sensor Networks with the Coffee File System*. In *Proceedings of the 8th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 349–360, 2009.
- [TDVo8] N. Tsiftes, A. Dunkels, and T. Voigt: *Efficient Sensor Network Reprogramming through Compression of Executable Modules*. In *Proceedings of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 359–367, 2008.
- [Tex07] Texas Instruments Inc.: *CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-Ready RF Transceiver (Rev. B)*. Online: <http://www.ti.com/lit/gpn/cc2420>, 2007. Last access: 14 October 2011.

- [Tex09] Texas Instruments Inc.: *MSP430 16-bit Ultra-Low Power Microcontroller*. Online: <http://www.ti.com/ww/de/msp430.html>, 2009. Last access: 14 October 2011.
- [TIK11] TIK WSN Research Group, ETH Zurich: *The Sensor Network Museum*. Online: <http://www.snm.ethz.ch>, 2011. Last access: 14 October 2011.
- [TK75] F. A. Tobagi and L. Kleinrock: *Packet Switching in Radio Channels: Part II - The Hidden Terminal Problem in Carrier Sense Multiple-Access Modes and the Busy-Tone Solution*. *IEEE Transactions on Communications*, 23(12):1417–1433, 1975.
- [Use03] UserLand Software, Inc.: *XML-RPC – Simple Cross-platform Distributed Computing, based on the Standards of the Internet*. Online: <http://www.xmlrpc.com/>, 2003. Last access: 14 October 2011.
- [VD10] J.-P. Vasseur and A. Dunkels: *Interconnecting Smart Objects with IP - The Next Internet*. Morgan Kaufmann, 2010.
- [vdBNWo9] A. van der Byl, R. Neilson, and R. Wilkinson: *An Evaluation of Compression Techniques for Wireless Sensor Networks*. In *Proceedings of the IEEE Africon Conference*, pages 1–6, 2009.
- [vdDKLo9] B. van der Doorn, W. Kavelaars, and K. Langendoen: *A Prototype Low-Cost Wakeup Radio for the 868 MHz Band*. *International Journal on Sensor Networks*, 5(1):22–32, 2009.
- [vDL03] T. van Dam and K. Langendoen: *An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks*. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 171–180, 2003.
- [VGM06] K. Van Laerhoven, H.-W. Gellersen, and Y. G. Malliaris: *Long-Term Activity Monitoring with a Wearable Sensor Node*. In *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks (BSN)*, pages 171–174, 2006.
- [Vin97] S. Vinoski: *CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments*. *IEEE Communications Magazine*, 35(2):46–55, 1997.
- [Vit87] J. S. Vitter: *Design and Analysis of Dynamic Huffman Codes*. *Journal of the Association for Computing Machinery*, 34(4):825–845, 1987.
- [VWS⁺06] G. Virone, A. Wood, L. Selavo, Q. Cao, L. Fang, T. Doan, Z. He, R. Stoleru, S. Lin, and J. A. Stankovic: *An Assisted Living Oriented Information System Based on a Residential Wireless Sensor Network*. In *Proceedings of the 1st Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare (D2H2)*, pages 95–100, 2006.
- [WAJR⁺05] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh: *Monitoring Volcanic Eruptions with a Wireless Sensor Network*. In *Proceedings of the 2nd European Workshop on Wireless Sensor Networks (EWSN)*, pages 108–120, 2005.
- [WASW05] G. Werner-Allen, P. Swieskowski, and M. Welsh: *MoteLab: A Wireless Sensor Network Testbed*. In *Proceedings of the 4th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 483–488, 2005.
- [WC01] A. Woo and D. E. Culler: *A Transmission Control Scheme for Media Access in Sensor Networks*. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 221–235, 2001.
- [WCK02] A. Wang, A. P. Chandrakasan, and S. V. Kosonocky: *Optimal Supply and Threshold Scaling for Subthreshold CMOS Circuits*. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 5–9, 2002.

- [Wei91] M. Weiser: *The Computer for the Twenty-First Century*. Scientific American, 265(3):66–75, 1991.
- [Wei93] M. Weiser: *Some Computer Science Issues in Ubiquitous Computing*. Communications of the ACM, 36(7):75–84, 1993.
- [Wel84] T. A. Welch: *A Technique for High-Performance Data Compression*. IEEE Computer, 17(6):8–19, 1984.
- [Wes06] C. Westphal: *Layered IP Header Compression for IP-enabled Sensor Networks*. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 3542–3547, 2006.
- [Whi75] J. E. White: *High-level Framework for Network-based Resource Sharing*. Request for Comments (RFC) 707, 1975.
- [WI95] M. D. Wheeler and K. Ikeuchi: *Sensor Modeling, Probabilistic Hypothesis Generation, and Robust Localization for Object Recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(3):252–265, 1995.
- [Wiro01] Wireless Application Protocol Forum, Ltd.: *Binary XML Content Format Specification, Version 1.3*, 2001.
- [WK05] C. Westphal and R. Koodli: *Stateless IP Header Compression*. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 3236–3241, 2005.
- [WLLP01] B. Warneke, M. Last, B. Liebowitz, and K. S. J. Pister: *Smart Dust: Communicating with a Cubic-Millimeter*. IEEE Computer, 34(1):44–51, 2001.
- [WP04] B. A. Warneke and K. S. J. Pister: *An Ultra-Low Energy Microcontroller for Smart Dust Wireless Sensor Networks*. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, pages 316–317, 2004.
- [WSBC04] K. Whitehouse, C. Sharp, E. Brewer, and D. Culler: *Hood: A Neighborhood Abstraction for Sensor Networks*. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 99–110, 2004.
- [WTT⁺06] K. Whitehouse, G. Tolle, J. Taneja, C. Sharp, S. Kim, J. Jeong, J. Hui, P. Dutta, and D. Culler: *Marionette: Using RPC for Interactive Development and Debugging of Wireless Embedded Networks*. In *Proceedings of the 5th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 416–423, 2006.
- [WTVZ⁺05] J. Walnes, M. Talevi, J. van Zyl, N. Pryce, and D. North: *XStream*. Online: <http://xstream.codehaus.org/>, 2005. Last access: 14 October 2011.
- [WZL06] K. Whitehouse, F. Zhao, and J. Liu: *Semantic Streams: A Framework for Composable Semantic Interpretation of Sensor Data*. In *Proceedings of the 3rd European Workshop on Wireless Sensor Networks (EWSN)*, pages 5–20, 2006.
- [XD10] R. Xuejun and F. Dingyi: *A Normal Distribution Encoding Algorithm for Slowly-Varying Data Compression in Wireless Sensor Networks*. In *Proceedings of the 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*, pages 1–4, 2010.
- [XLC04] Z. Xiong, A. D. Liveris, and S. Cheng: *Distributed Source Coding for Sensor Networks*. IEEE Signal Processing Magazine, 21(5):80–94, 2004.
- [YD09] D. Yazar and A. Dunkels: *Efficient Application Integration in IP-based Sensor Networks*. In *Proceedings of the 1st ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings (BuildSys)*, pages 43–48, 2009.

- [YHE01] W. Ye, J. Heidemann, and D. Estrin: *An Energy-Efficient MAC protocol for Wireless Sensor Networks*. Technical Report ISI-TR-543, Information Science Institute (ISI) / University of Southern California (USC), 2001.
- [YHE04] W. Ye, J. Heidemann, and D. Estrin: *Medium Access Control With Coordinated Adaptive Sleeping for Wireless Sensor Networks*. *IEEE/ACM Transactions on Networking*, 12(3):493–506, 2004.
- [ZGo3] J. Zhao and R. Govindan: *Understanding Packet Delivery Performance In Dense Wireless Sensor Networks*. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 1–13, 2003.
- [Zim80] H. Zimmermann: *OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection*. *IEEE Transactions on Communications*, 28(4):425–432, 1980.
- [ZL77] J. Ziv and A. Lempel: *A Universal Algorithm for Sequential Data Compression*. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.
- [ZL78] J. Ziv and A. Lempel: *Compression of Individual Sequences via Variable-Rate Coding*. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.
- [ZRMS10] S. Zöllner, A. Reinhardt, M. Meyer, and R. Steinmetz: *A Concept for Cross-Layer Optimization of Wireless Sensor Networks in the Logistics Domain by Exploiting Business Knowledge*. In *Proceedings of the 5th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, pages 951–953, 2010.
- [ZSLMo4] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi: *Hardware Design Experiences in ZebraNet*. In *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 227–238, 2004.

LIST OF ACRONYMS

6LoWPAN	IPv6 over Low-power Wireless Personal Area Networks
AAL	Ambient Assisted Living
ADC	Analog-to-Digital Converter
AHC	Adaptive Huffman Coding
AmI	Ambient Intelligence
ASIC	Application-specific Integrated Circuit
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
BWT	Burrows-Wheeler Transform
CID	Context Identifier
CORBA	Common Object Request Broker Architecture
CRC	Cyclic Redundancy Check
CTP	Collection Tree Protocol
DC	Direct Current
DMA	Direct Memory Access
DRL	Data Representation Layer
DSN	Data Sequence Number
ECA	Event-Condition-Action
FCF	Frame Control Field
GPIO	General Purpose Input/Output
GPS	Global Positioning System
HC	Header Compression
HTTP	Hypertext Transfer Protocol
HVAC	Heating, Ventilation, and Air Conditioning
I ² C	Inter-Integrated Circuit
IC	Integrated Circuit
IoT	Internet of Things
IP	Internet Protocol
IPv6	Internet Protocol in version 6
IPSO	IP for Smart Objects
ISM	Industrial, Scientific, and Medical

JSON	JavaScript Object Notation
LED	Light Emitting Diode
M2M	Machine-to-Machine
MAC	Medium Access Control
MCU	Microcontroller Unit
MEMS	Micro-Electro-Mechanical System
MPDU	MAC Protocol Data Unit
MTF	Move-to-Front Coding
OSGi	Open Services Gateway initiative
PAN	Personal Area Network
PDA	Personal Digital Assistant
PER	Packed Encoding Rules
PHY	Physical Layer
PWM	Pulse-width Modulation
QoS	Quality of Service
RAM	Random Access Memory
RC	Range Coding
RFID	Radio-Frequency Identification
RLE	Run-Length Encoding
RMI	Remote Method Invocation
ROM	Read-Only Memory
RPC	Remote Procedure Call
SFD	Start-of-Frame Delimiter
SPI	Serial Peripheral Interface
S-RPC	Sensor-RPC
TCP	Transmission Control Protocol
TWiNS.KOM	Testbed for a Wireless Network of Sensors
UDP	User Datagram Protocol
USART	Universal Synchronous/Asynchronous Receiver Transmitter
WBXML	WAP Binary XML
WSN	Wireless Sensor Network
WSAN	Wireless Sensor and Actuator Network
XDR	eXternal Data Representation
XML	eXtended Markup Language

Try to learn something about everything
and everything about something.

Thomas Henry Huxley

A.1 NOTATIONS OF SQUEEZE.KOM

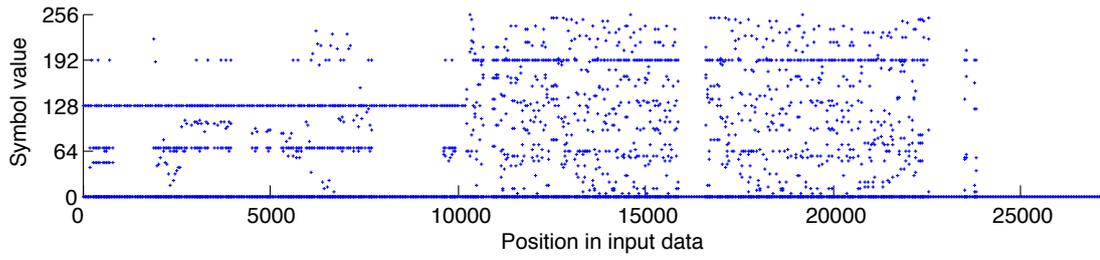
The notations used to denote different packet types in the Squeeze.KOM compression layer are listed in Table 30.

Table 30: Notations used in Squeeze.KOM

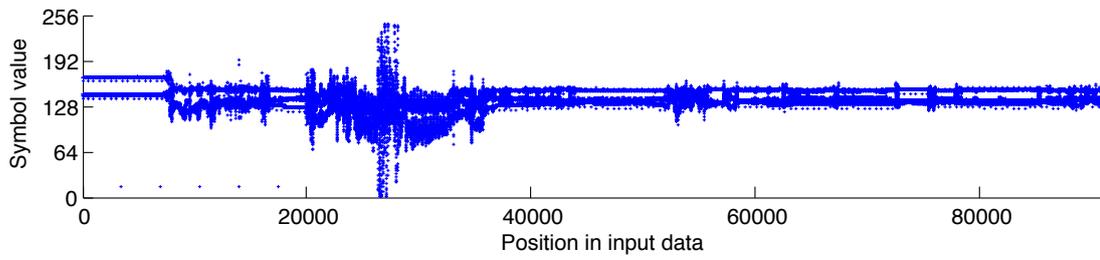
P	Packet payload
$GC(P)$	Gray code encoding of the payload P
$Hash(P)$	Hash value of the payload P
P_t	Payload of the packet transmitted at time t
P_0	Payload of the first packet of a sequence
n	Index of an entry within the local history
I_n	Index packet with entry number n in the local history
b_n	Minimum number of differing bits between P and the entries in the local history
th	Similarity metric threshold (maximum differing bits between P and the local history)
Δ_n	Bitwise difference between a payload P and the packet I_n
$\Delta_{n,enc}$	Encoded representation of Δ_n after the application of binary distance coding

A.2 ENVIRONMENTAL MONITORING AND BODY AREA SENSING DATA TRACES

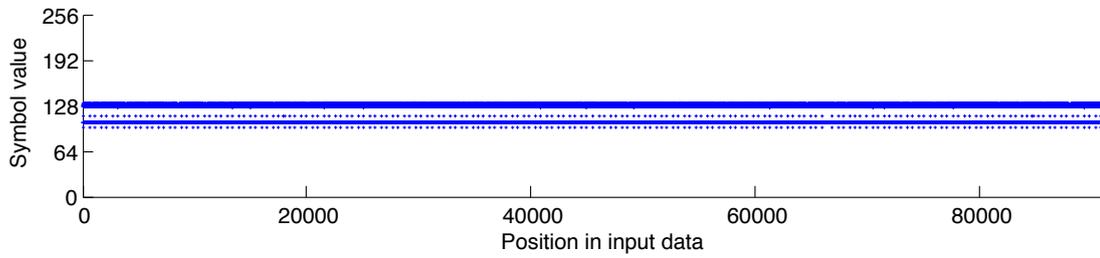
In this section, the data sets used in our evaluation of Squeeze.KOM and the AHC implementation with limited code tree sizes are visualized in their entirety. Traces of the Glacswab and Porcupine deployments are depicted in Fig. 56, while the PermaSense trace is shown in Fig. 57.



(a) Complete trace of the used Glacsweb data set



(b) Complete trace of the used Porcupine active data set



(c) Complete trace of the used Porcupine sleep data set

Figure 56: Complete traces of the used environmental and body area monitoring data sets

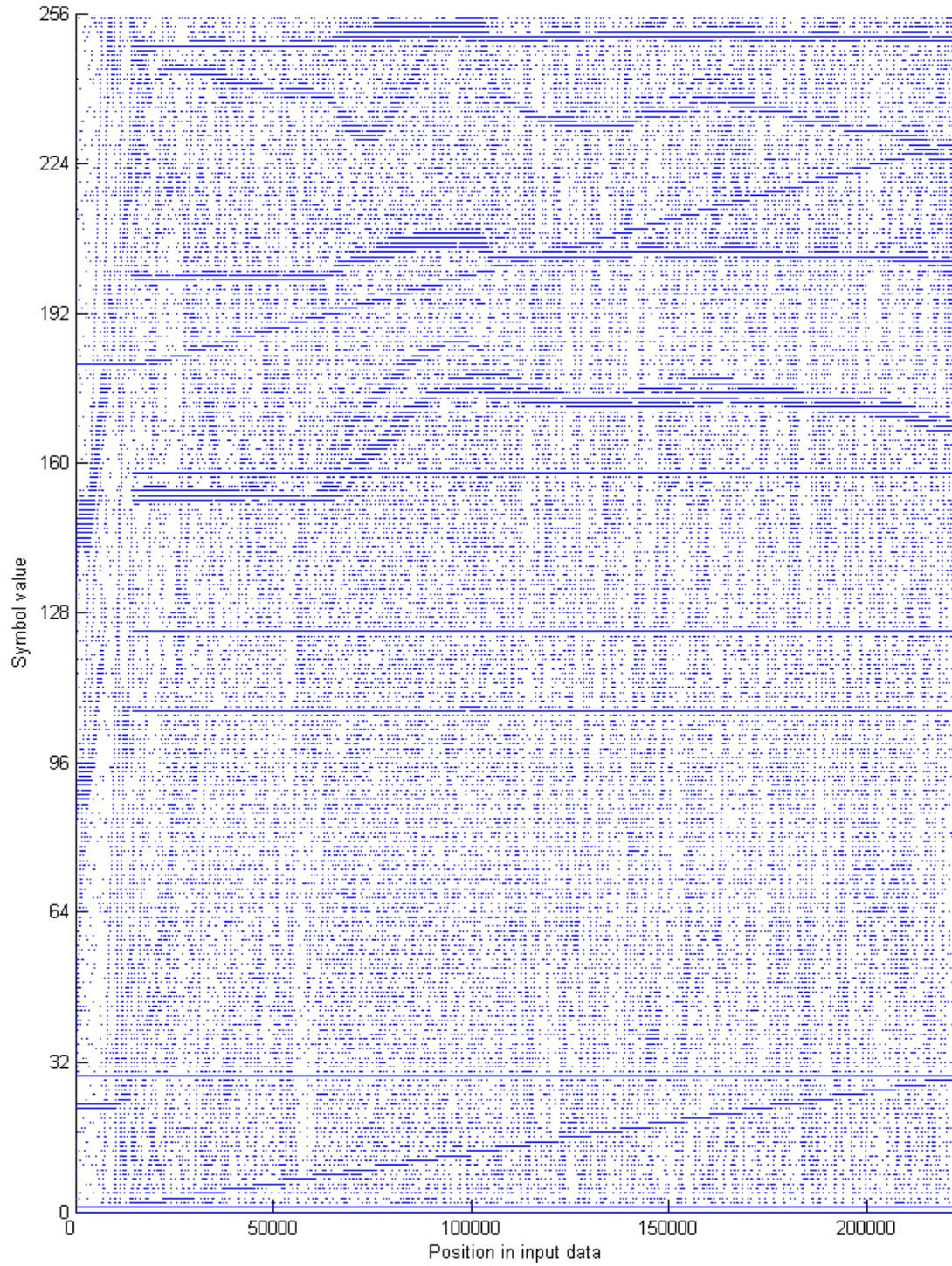
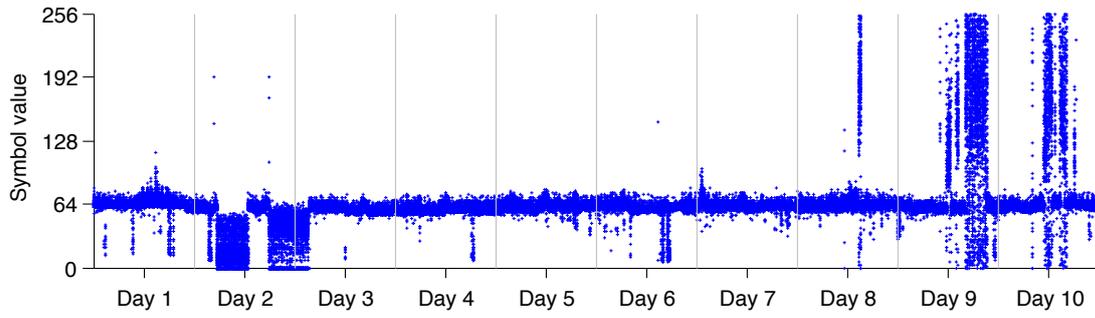


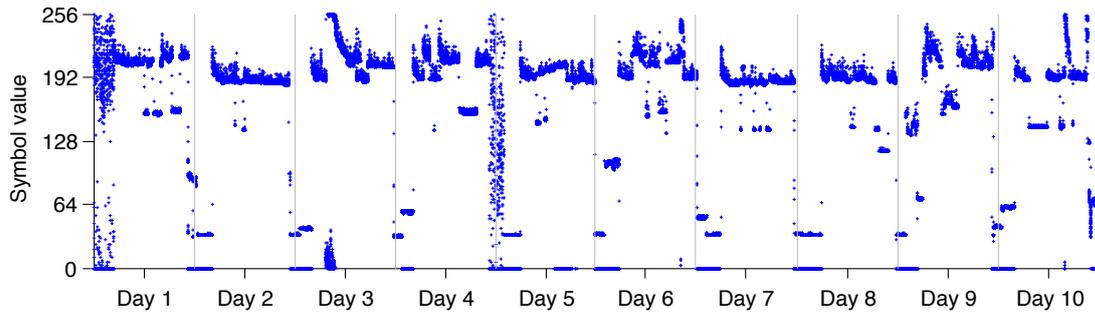
Figure 57: Complete trace of the used PermaSense data set

A.3 COMPLETE DATA TRACES COLLECTED IN AN OFFICE ENVIRONMENT

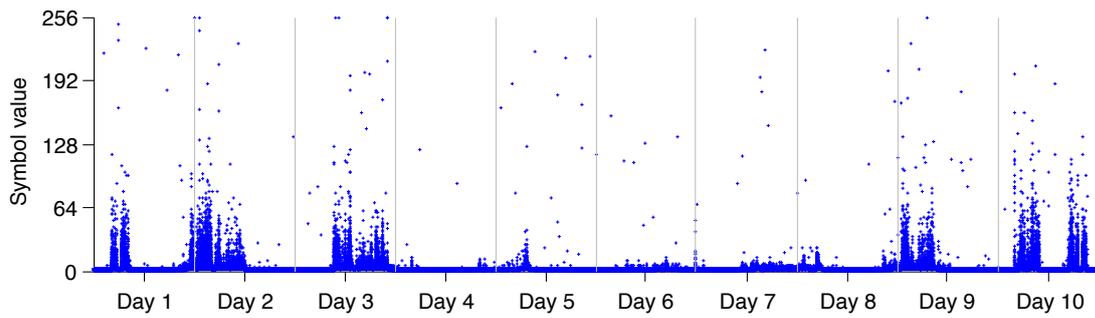
We present representative traces of the data collected in an office environment through the TWiNS.KOM testbed in Fig. 58. This data has been used in the evaluation of the presented data compression algorithms with real-world data in Sec. 5.4.



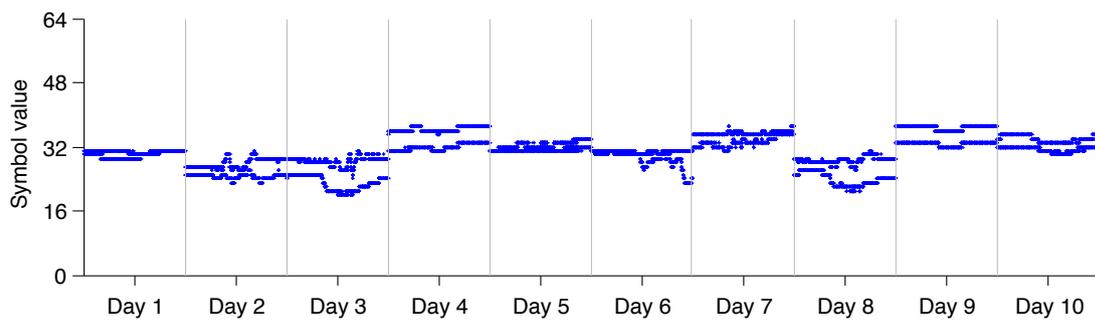
(a) Full sensor trace of the door motion sensor



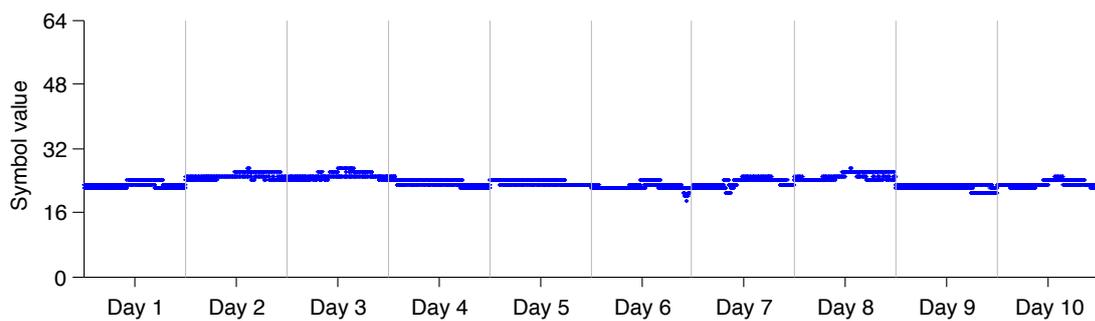
(b) Full sensor trace of the power sensor



(c) Full sensor trace of the sound level sensor



(d) Full sensor trace of the humidity sensor



(e) Full sensor trace of the temperature sensor

Figure 58: Complete traces of the used real-world data sets

A.4 SFHC.KOM PACKET TYPES AND CORRESPONDING HEADER FIELD VALUES

The complete reference to all FCF values used in SFHC.KOM is listed in Table 31.

Table 31: Complete reference to the packet types and corresponding FCF values used in SFHC.KOM

b ₀	b ₁	b ₂	b ₃	b ₄	b ₅	INTERPRETATION
0	0	1	0	1	0	Unacknowledged C-DEST data frame without AM type field
0	0	1	0	1	1	Unacknowledged C-DEST data frame with AM type field
0	0	1	1	1	1	Acknowledged C-DEST data frame without AM type field
0	0	1	1	1	0	Acknowledged C-DEST data frame with AM type field
0	0	1	1	0	0	C-DEST acknowledgment frame
1	0	1	0	1	0	Unacknowledged C-LONG data frame without AM type field
1	0	1	0	1	1	Unacknowledged C-LONG data frame with AM type field
1	0	1	1	1	1	Acknowledged C-LONG data frame without AM type field
1	0	1	1	1	0	Acknowledged C-LONG data frame with AM type field
1	0	1	1	0	0	C-LONG acknowledgment frame
0	1	1	0	0	1	C-DEST establishment request
0	1	1	0	1	1	Acceptance notification of C-DEST request
0	1	1	0	0	0	Denial notification of C-DEST request
1	1	1	0	0	0	Preparation request for C-LONG establishment
1	1	1	0	0	1	Request for CID bit vectors of neighboring nodes
1	1	1	0	1	0	C-LONG establishment request
1	1	1	0	1	1	Acceptance notification of C-LONG request
1	1	1	1	0	0	Denial notification of C-LONG request

CURRICULUM VITÆ

PERSONAL DETAILS

Name	Andreas Reinhardt
Date of Birth	4 December 1981
Place of Birth	Groß-Gerau, Germany
Nationality	German

EDUCATION

<i>since 12/2007</i>	Technische Universität Darmstadt, Germany Doctoral candidate at the Department of Electrical Engineering and Information Technology
<i>07/2006 – 11/2006</i>	The University of New South Wales, Sydney, Australia Student research project “Distributed Boundary Estimation in Wireless Sensor Networks by Applying the Right-hand Rule and Coverage Determination by Applying the Probabilistic Coverage Algorithm”
<i>10/2002 – 10/2007</i>	Technische Universität Darmstadt, Germany Studies of Electrical Engineering and Information Technology Degree: Diplom-Ingenieur (Dipl.-Ing.), M.Sc. equivalent
<i>08/1992 – 06/2001</i>	Prälat-Diehl-Schule, Gymnasium des Kreises Groß-Gerau, Germany Degree: Allgemeine Hochschulreife (Abitur)

WORK EXPERIENCE

<i>since 09/2009</i>	Freelance consultant for embedded systems design with special focus on Wireless Sensor Networks
<i>since 12/2007</i>	Technische Universität Darmstadt, Germany Doctoral candidate and research staff member at the Multimedia Communications Lab
<i>05/2006 – 07/2005</i>	Technische Universität Darmstadt, Germany Student assistant for the “Setup of a Mobile Sensor Node Testbed”
<i>04/2006 – 07/2006</i>	Technische Universität Darmstadt, Germany Student assistant for tutoring the undergraduate lab exercise “Microcontroller Programming”
<i>06/2004 – 09/2004</i>	Adams Consult GmbH & Co. KG, Groß-Gerau, Germany Internship in cable television fault clearing and web application development
<i>08/2001 – 06/2002</i>	Helen-Keller-Schule, Rüsselsheim, Germany Community service at the Helen Keller school for physically handicapped children

TEACHING ACTIVITIES

- since 12/2007* Technische Universität Darmstadt, Germany
Tutor for various diploma, Bachelor, and Master theses
- since 12/2007* Technische Universität Darmstadt, Germany
Tutor for various seminars, lab exercises, and projects
- 10/2009 – 03/2010* Technische Universität Darmstadt, Germany
Coordination of the seminar “Advanced Topics of Future Internet Research”
- since 10/2008* Technische Universität Darmstadt, Germany
Coordination of Bachelor and Master level lab exercises (“Praktikum”) and projects (“Projektseminar”)
- 12/2007 – 09/2008* Technische Universität Darmstadt, Germany
Teaching assistant for lecture and lab exercises on “Advanced Topics in Distributed Systems”

AWARDS

- 05/2008* Best M.Sc. thesis award at the 7th European Workshop on Microelectronics Education (EWME)

Darmstadt, 18 October 2011

AUTHOR'S PUBLICATIONS

MAIN PUBLICATIONS

1. Andreas Reinhardt, Johannes Schmitt, Ralf Steinmetz, Philipp Walter, and Simon Schwantzer. PROWIT – Kontext-sensitive Telekommunikation im Geschäftseinsatz. *Praxis der Informationsverarbeitung und Kommunikation (in press)*, 2011.
2. Andreas Reinhardt, Dominic Burkhardt, Parag S. Mogre, Manzil Zaheer, and Ralf Steinmetz. SmartMeter.KOM: A Low-cost Wireless Sensor for Distributed Power Metering. In *Proceedings of the 6th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, pages 1044–1051, 2011.
3. Andreas Reinhardt, Parag Mogre, and Ralf Steinmetz. Lightweight Remote Procedure Calls for Wireless Sensor and Actuator Networks. In *Workshop Proceedings of the 9th Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, pages 116–121, 2011.
4. Andreas Reinhardt, Parag Mogre, Tobias Koenig, and Ralf Steinmetz. SFHC.KOM: Stateful Header Compression for Wireless Sensor Networks. In *Proceedings of the 35th IEEE Conference on Local Computer Networks (LCN)*, pages 264–271, 2010.
5. Andreas Reinhardt, Diego Costantini, and Ralf Steinmetz. Describing Packet Payload Structures using Lightweight Semantic Data Type Annotations. In *Proceedings of the 9th GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“ (FGSN)*, pages 25–28, 2010.
6. Andreas Reinhardt, Johannes Schmitt, Farid Zaid, Parag Mogre, Matthias Kropff, and Ralf Steinmetz. Towards Seamless Binding of Context-aware Services to Ubiquitous Information Sources. In *Proceedings of the 4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*, pages 69–74, 2010.
7. Andreas Reinhardt, Delphine Christin, Matthias Hollick, Johannes Schmitt, Parag Mogre, and Ralf Steinmetz. Trimming the Tree: Tailoring Adaptive Huffman Coding to Wireless Sensor Networks. In *Proceedings of the 7th European Conference on Wireless Sensor Networks (EWSN)*, pages 33–48, 2010.
8. Andreas Reinhardt, Delphine Christin, Matthias Hollick, and Ralf Steinmetz. On the Energy Efficiency of Lossless Data Compression in Wireless Sensor Networks. In *Proceedings of the 4th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, pages 873–880, 2009.
9. Andreas Reinhardt and Ralf Steinmetz. Exploiting Platform Heterogeneity in Wireless Sensor Networks for Cooperative Data Processing. In *Proceedings of the 8th GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“ (FGSN)*, pages 21–24, 2009.
10. Andreas Reinhardt, Matthias Hollick, and Ralf Steinmetz. Stream-oriented Lossless Packet Compression in Wireless Sensor Networks. In *Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 1–9, 2009.
11. Andreas Reinhardt, Matthias Kropff, Matthias Hollick, and Ralf Steinmetz. Designing a Sensor Network Testbed for Smart Heterogeneous Applications. In *Proceedings of the 3rd IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, pages 715–722, 2008.

12. Andreas Reinhardt, Jan Hennecke, Steffen Gottwald, Matthias Kropff, Johannes Schmitt, Matthias Hollick, and Ralf Steinmetz. Tubicles: Heterogeneous Wireless Sensor Nodes – Testbed Objectives and Assembly Instructions. Technical Report TR-KOM-2008-09, Multimedia Communications Lab, Technische Universität Darmstadt, 2008.
13. Andreas Reinhardt, Matthias Hollick, and Ralf Steinmetz. An Approach towards Adaptive Payload Compression in Wireless Sensor Networks. In *Proceedings of the 7th GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“ (FGSN)*, pages 19–21, 2008.
14. Andreas Reinhardt, Heiko Hinkelmann, and Manfred Glesner. Developing a Debugging Interface for Reconfigurable Wireless Sensor Nodes. In *Proceedings of the 7th European Workshop on Microelectronics Education (EWME)*, pages 46–47, 2008. Received the Best MSc Paper Award.

PATENT APPLICATIONS

15. Räumliche Anordnung einer Mehrzahl von Kommunikationsendgeräten und Verfahren zur Bestimmung der räumlichen Position eines Gerätes. Internationale Patentanmeldung PCT/EP2010/002835.

CO-AUTHORED PUBLICATIONS

16. Sebastian Zöllner, Andreas Reinhardt, Ulrich Lampe, André Miede, and Ralf Steinmetz. Kontextbasierte eventerkennung in der logistik mit drahtloser sensornetztechnologie (accepted for publication). In *Multikonferenz der Wirtschaftsinformatik (MKWI)*, 2012.
17. Delphine Christin, Andreas Reinhardt, Salil S. Kanhere, and Matthias Hollick. A Survey on Privacy in Mobile Participatory Sensing Applications. *Journal of Systems and Software*, 84(11):1928–1946, 2011.
18. Sebastian Zöllner, Andreas Kaiser, Andreas Reinhardt, Stefan Schulte, and Ralf Steinmetz. A Multi-Threshold Approach for Efficient and User-centric Event Transmission in Logistics Wireless Sensor Networks. In *Proceedings of the 6th International Conference on Pervasive Computing and Application (ICPCA)*, pages 1–8, 2011.
19. Delphine Christin, Julien Guillemet, Andreas Reinhardt, Matthias Hollick, and Salil Kanhere. Privacy-preserving Collaborative Path Hiding for Participatory Sensing Applications. In *Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pages 341–350, 2011.
20. Sebastian Zöllner, Andreas Reinhardt, Stefan Schulte, and Ralf Steinmetz. Scoresheet-based Event Relevance Determination for Energy Efficiency in Wireless Sensor Networks. In *Proceedings of the 36th Annual IEEE Conference on Local Computer Networks (LCN)*, pages 207–210, 2011.
21. Sebastian Zöllner, Andreas Reinhardt, Heiko Guckes, Dieter Schullerwie, and Ralf Steinmetz. On the Integration of Wireless Sensor Networks and Smartphones in the Logistics Domain. In *Proceedings of the 10th GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“ (FGSN)*, pages 49–52, 2011.
22. Farid Zaid, Parag Mogre, Andreas Reinhardt, Diego Costantini, and Ralf Steinmetz. iVu.KOM: A Framework for Viewer-centric Mobile Location-based Services. *Praxis der Informationsverarbeitung und Kommunikation (PIK)*, 33(4):284–290, 2010.

23. Sebastian Zöller, Andreas Reinhardt, Marek Meyer, and Ralf Steinmetz. A Concept for Cross-Layer Optimization of Wireless Sensor Networks in the Logistics Domain by Exploiting Business Knowledge. In *Proceedings of the 5th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, pages 951–953, 2010.
24. Farid Zaid, Diego Costantini, Parag Mogre, Andreas Reinhardt, Johannes Schmitt, and Ralf Steinmetz. WBroximity: Mobile Participatory Sensing for WLAN- and Bluetooth-based Positioning. In *Proceedings of the 5th IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, pages 906–912, 2010.
25. Sebastian Zöller, Andreas Reinhardt, Marek Meyer, and Ralf Steinmetz. Deployment of Wireless Sensor Networks in Logistics – Potential, Requirements, and a Testbed. In *Proceedings of the 9th GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“ (FGSN)*, pages 67–70, 2010.
26. Diego Costantini, Andreas Reinhardt, Parag Mogre, and Ralf Steinmetz. Exploring the Applicability of Participatory Sensing in Emergency Scenarios. In *Proceedings of the 9th GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“ (FGSN)*, pages 59–62, 2010.
27. Delphine Christin, Andreas Reinhardt, Salil S. Kanhere, and Matthias Hollick. Fine-grained Access Control Enabling Privacy Support in Wireless Sensor Networks. In *Proceedings of the 9th GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“ (FGSN)*, pages 29–32, 2010.
28. Farid Zaid, Johannes Schmitt, Parag Mogre, Andreas Reinhardt, Matthias Kropff, and Ralf Steinmetz. Sorting the Wheat from the Chaff: Adaptive Sensor Selection for Context-aware Applications. In *Proceedings of the 2nd International Workshop on Information Quality and Quality of Service for Pervasive Computing (IQ2S)*, pages 87–92, 2010.
29. Delphine Christin, Andreas Reinhardt, Parag Mogre, and Ralf Steinmetz. Wireless Sensor Networks and the Internet of Things: Selected Challenges. In *Proceedings of the 8th GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“ (FGSN)*, pages 31–34, 2009.
30. Johannes Schmitt, Matthias Kropff, Andreas Reinhardt, Matthias Hollick, Christian Schäfer, Frank Remetter, and Ralf Steinmetz. An Extensible Framework for Context-aware Communication Management Using Heterogeneous Sensor Networks. Technical Report TR-KOM-2008-08, Multimedia Communications Lab, Technische Universität Darmstadt, 2008.
31. Heiko Hinkelmann, Andreas Reinhardt, and Manfred Glesner. A Methodology for Wireless Sensor Network Prototyping with Sophisticated Debugging Support. In *Proceedings of the 19th IFIP/IEEE International Symposium on Rapid System Prototyping (RSP)*, pages 82–88, 2008.
32. Heiko Hinkelmann, Andreas Reinhardt, Sameer Varyani, and Manfred Glesner. A Reconfigurable Prototyping Platform for Smart Sensor Networks. In *Proceedings of the 4th Southern Conference on Programmable Logic (SPL)*, pages 125–130, 2008.

ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe. Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 18. Oktober 2011

Dipl.-Ing. Andreas Reinhardt