Andreas Reinhardt, Delphine Christin, Matthias Hollick, Ralf Steinmetz:
On the Energy Efficiency of Lossless Data Compression in Wireless Sensor Networks. In:
Proceedings of the Fourth IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2009), October 2009

On the Energy Efficiency of Lossless Data Compression in Wireless Sensor Networks

Andreas Reinhardt*, Delphine Christin*, Matthias Hollick†, Ralf Steinmetz*

* Multimedia Communications Lab, Technische Universität Darmstadt
Merckstr. 25, 64283 Darmstadt, Germany
{*andreas.reinhardt, delphine.christin, ralf.steinmetz*}*@kom.tu-darmstadt.de*

† Departamento de Ingeniería Telemática, Universidad Carlos III de Madrid
30 Avenida de la Universidad, Leganés, 28912 Madrid, Spain
matthias.hollick@uc3m.es

***Abstract*—In wireless sensor networks, energy is commonly a scarce resource, which should be used as sparingly as possible to allow for long node lifetimes. It is therefore mandatory to put a focus on the development of energy-efficient applications. In this paper, we analyze the achievable energy gains when packet payloads are compressed prior to their transmission. As the radio transceiver chips are the predominant power consumers on most current sensor node platforms, we present how local compression of data can be successfully employed to preserve energy. We compare two lossless mechanisms to eliminate redundancies in the packets with regard to the overall energy savings. The results prove that data compression is a viable approach to reduce a platform's energy consumption, as it can reduce the radio transmission durations of packets and thus shorten the duty cycles of the radio device.**

I. Introduction

Commonly, nodes in wireless sensor networks (*WSNs*) are limited in their available energy budget due to their battery-powered operation [1]. Unfortunately, this limitation is only addressed in an insufficient manner in most of today's sensor network applications, although a variety of approaches towards energy-aware sensor networking exist (cf. [2]). The power consumption of the communication unit of a sensor network node (*mote*) often dominates its energy demand in both transmission and reception mode; minimizing the duty cycles of the radio transceivers is therefore a viable approach towards preserving energy. While energy-aware MAC protocols, such as S-MAC [3] or B-MAC [4], target to reduce the activity periods of the radio unit, data compression can be used to either reduce the transmission durations, or to increase the information density within a payload of the same length.

In this paper, we assume a sensor network scenario where data needs to be transmitted to a base station at a regular interval. With regard to this constraint, being present in various application domains, like homecare or medical sensor networks ([5], [6]), we do not investigate network-level approaches towards energy reduction, such as data aggregation [7]. Instead, we put the focus of this paper on the energy gains achievable by the local compression of packet payloads. Similarly, header compression mechanisms such as defined in RFC 4944 [8] are beyond the scope of this paper.

Theoretical considerations prove it possible to achieve energy savings by applying data compression, as the shorter payload results in a shorter transmission time (e.g. [1], [9]). However, compressing the data requires additional microcontroller operations, which in effect increase the microcontroller energy consumption. It is therefore necessary to find a trade-off which achieves a global energy reduction. In this paper, we analyze the energy gains achievable by data compression mechanisms in simulation, and provide estimates for their real-world applicability. To achieve further size reductions and thus energy savings, data compression should ideally be supplemented by header compression algorithms and energy-efficient MAC layer protocols for a maximum node lifetime.

We perform evaluations using *lossless* compression algorithms only. The analysis would have been possible for *lossy* compression algorithms as well, although we expect the results to be somewhat similar. When losses in precision are tolerated, the achievable savings might be even greater due to the higher compressibility of the data, and the resulting smaller packets.

The key contribution of this paper is the investigation of the applicability of data compression algorithms and their effects on the sensor node energy budgets. We use representative data sets from the Porcupine sensor platform, which performs person activity tracking through readings of on-board accelerometers [10]. Two versions of run-length encoding as well as an implementation of adaptive Huffman coding are evaluated with a focus on their energy consumption. Using the Avrora simulator [11], our simulations particularly focus on the sensor node microcontroller and radio energy consumptions. Simulations are initially performed with synthetic data in order to evaluate the achievable best and worst case results. In a second step, the algorithms are evaluated with data sets from the real deployment.

We present related work on data compression in WSNs in Section II, and provide details on the simulation setup in Section III, including the simulation tools, the mote platform, and the used data set. Obtained simulation results focused on the microcontroller, radio and global energy gains are presented and discussed in Section IV, and we conclude this paper in Section V.

The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

II. Related work

The area of data compression in WSNs has received some attention in previous literature. Although focusing on wireless ad-hoc networks, Barr and Asanović show in [9] that the energy required for transmitting a bit can be equivalent to the energy consumption of a thousand microcontroller operations. However, their results were acquired from a Compaq Personal Server handheld, which features 32 megabytes of RAM and 16 kilobytes of cache. This significantly exceeds the resource constraints on many of today's mote platforms.

Sadler and Martonosi present a variation of the lossless LZW algorithm in [12]. Specially designed for common sensor platforms with a few kilobytes of memory, their version compresses data blocks with a length of 528 bytes at a time. Tested with data from real sensor networks, the S-LZW algorithm shows energy savings up to a factor of more than 1.5x locally, and over 2.5x for the overall network. However, the system was evaluated in a delay-tolerant network setting, where data was buffered before being transmitted.

Tsiftes et al. compare mechanisms to compress code updates to remotely reconfigure nodes in [13]. They present the SBZIP algorithm, which combines multiple preprocessing and coding steps, while maintaining the characteristics that the decoder part can still be run on memory-constrained sensor platforms. Results show that network-wide energy savings of to 67% can be achieved when compressing the code updates using GZIP.

Ju and Cui presented the EasiPC packet compression mechanism in [14]. Each packet field needs to be analyzed and classified according to its changing frequency in advance. Different compression methods are associated with each category; while randomly changing fields are always transferred uncompressed, sequence number fields are e.g. encoded by difference coding. In addition, the sensor readings are encoded by either difference coding or length-variable coding. Removing redundant information from the packet, the developed mechanism allows for compression gains of up to 50% as well as the corresponding reduced transmission delays. The results were however not verified by an energy analysis.

In previous work, we have investigated a differential encoding scheme exploiting the temporal resemblance of sensor network packets [15]. The application-agnostic solution provides a generic framework for data compression with size reductions of up to 35% for a real data set, although a thorough energy analysis has been identified as future work.

As none of the presented publications discusses the achievable energy savings when sensor readings are compressed at the originating node, we provide a detailed investigation of the corresponding energy gains in this paper.

III. Simulation Setup

The goal of this simulation study is the assessment of the achievable energy gains when data compression is applied prior to packet transmissions. We analyze a single-hop scenario, where a sensor node priodically delivers data to a sink. Starting with an analysis of the data set and its characteristics, we present our selection of the simulation tool, the considered mote platform, and brief descriptions of the used compression algorithms and their resource demands.

A. Data Set

In many application fields for wireless sensor networks, such as home care or patient monitoring in healthcare scenarios, collected data needs to be transferred to a base station periodically with low latency. The available data should be up-to-date to avoid ambiguous interpretation possibilities and allow for the correct action to be taken in case of emergencies. Although in a regular state of operation, successive readings are not expected to significantly differ from each other, they need to be transferred to the base station at regular intervals.

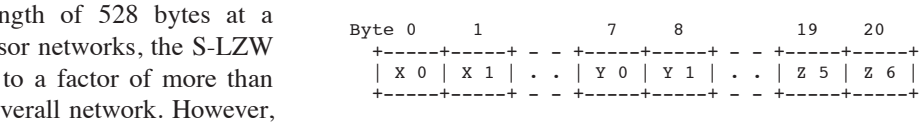


Fig. 1. Packet structure used in the Porcupine project

The wearable Porcupine nodes have been designed to determine a person's activity from accelerometer readings, which is a common scenario for wearable sensor networks. Transmitted packets follow the structure shown in Fig. 1; the payload consists of seven accelerometer readings in X, Y, and Z dimension of one byte each. An exemplary plot of the data is given in Fig. 2, showing a 10 minute excerpt of the data where a phase of almost no movement is followed by a series of quickly changing readings. All acceleration readings are shown as unsigned byte values. A closer look on the data set reveals that it is dominated by long runs of similar accelerometer readings when there is no movement of the wearable sensor, while during periods of high activity, frequent and fast changes of the sensor readings are observed.

B. Simulator Selection

Simulating application behavior before the actual deployment is a useful tool to identify problems and debug them. Moreover, the profiling capabilities of simulators allow evaluating general properties of applications without deploying them on real motes. The obtained simulation results can

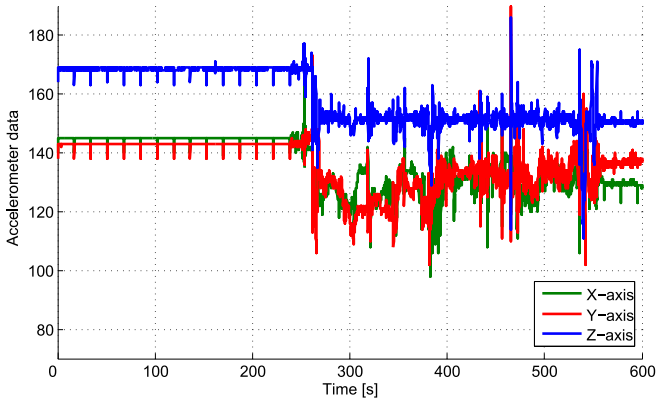


Fig. 2. Example Porcupine accelerometer data

be used to modify applications quickly, without requiring a time-consuming reinstallation on motes. Although a set of instruction-level simulators exist (cf. [16]), we have only considered simulators compatible with TinyOS which additionally feature energy profiling. As a result, we have selected the Avrora simulator for our analyses [11], combined with the AEON energy modelling extension [17]. Avrora is implemented in Java and its source code is available to easily perform modifications to the code, e.g. to adapt power consumption values or to increase the available RAM size.

All following simulations were conducted using Avrora with the AEON energy extension. The Avrora simulator offers the possibility to specify an input file containing sensor readings in order to simulate the data collection process through a sensor interface. We have inserted the data sets into the mote using this approach, but disregarded the corresponding energy demand in the simulation results, as it is an inherent hardware characteristic and presents a static energy consumption.

C. Mote Platform Selection

We have performed all our implementations in TinyOS [18] and compiled the applications for the Mica2 platform, as a fine-grained energy model is already implemented within the Avrora simulator. The Porcupine node intended to perform the data compression is however based on different hardware components with different specifications, and Avrora does not ship with a corresponding energy model. To estimate its power consumption, we thus compare its energy profile to values from the Mica2 data sheet in Table I. The figures for the Porcupine were hereby extracted from the data sheets of its components.

The numbers confirm that the energy consumptions of both platforms increase dramatically when the radio is configured in active, i.e. in transmission (*TX*), reception (*RX*), or idle modes. However, it is also obvious that the overall consumptions of the platforms are similar in these modes. The Porcupine only requires significantly more energy than the Mica2 when the microcontroller unit (*MCU*) is active while the radio transceiver is powered down. This state is present while data is collected from the accelerometer, and when data processing, such as compression, takes place. We have allowed the operating system to put the platform into sleep mode in between data sampling operations to save energy. All following simulations use the implementation of the B-MAC protocol [4] provided in TinyOS.

Concluding from the energy consumption values, we anticipate our simulation results to represent good – although possibly pessimistic, as we disregard the smaller sleep mode energy demand – estimates for the Porcupines, when weighting the MCU energy by a factor of $\frac{51mW}{36mW} = 1.416$.

In the following sections, Flash memory and RAM consumption values are stated according to the output of the TinyOS compiler. To determine the increase in resource consumption, we have implemented reference applications without any compression code for all of our simulations, and show all results in comparison to these reference values.

TABLE I
POWER CONSUMPTION OF MICA2 AND PORCUPINE MOTES

		Mica2 [19]	Porcupine
Specs	Microcontroller	ATmega128L	PIC18F4550
	Flash memory	128 kilobytes	32 kilobytes
	RAM	4 kilobytes	2 kilobytes
	Radio unit	Chipcon CC1000	Chipcon CC2420
Energy	MCU sleep, Radio off	0.054 mW	0.014 mW
	MCU active, Radio off	36 mW	51 mW
	MCU active, Radio RX	117 mW	120 mW
	MCU active, Radio TX	117 mW	115 mW

D. Algorithm Selection

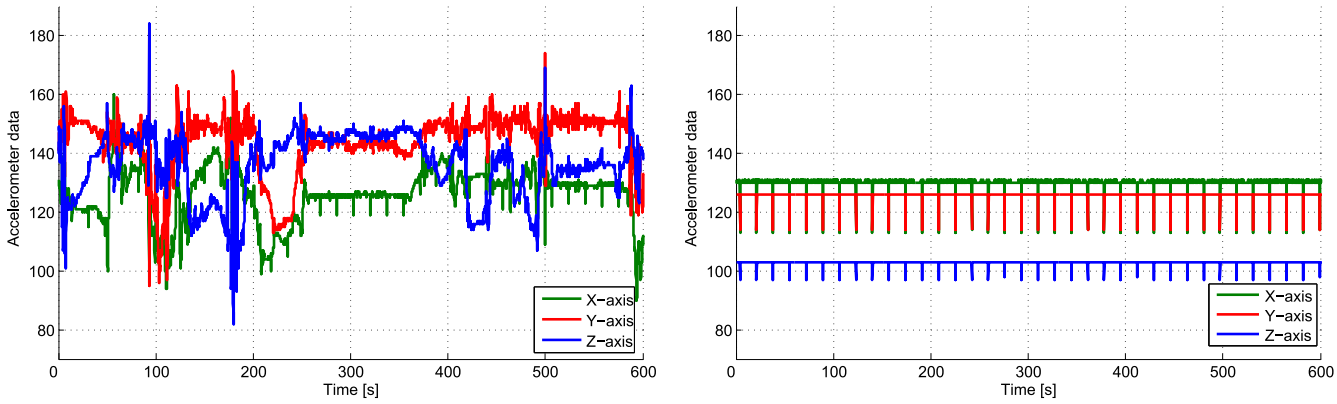
Due to the determined nature of many successive bytes within the input data stream, we have selected compression algorithms which deliberately exploit these data structures. Our second criterion was the demand to achieve good compression gains while compiling to sizes within the resources available on current mote platforms. While run-length encoding (*RLE*) replaces multiple occurrences of an input symbol by a repetition count field, the adaptive Huffman coding (*AHC*) algorithm automatically adapts to the symbol occurrence frequencies within the input data and adjusts its code table accordingly. Frequent symbols are assigned short binary codes, while the codes for less frequent symbols are longer.

We have added a one byte status field preceding the packet payload in our implementations, which contains information about whether the following payload contains compressed or uncompressed data, and which algorithm has been used. Although the compression step is always performed, data is only sent compressed if the corresponding output size is smaller than the uncompressed data. In case of incompressible data, the packet size to be transmitted thus increases by one byte.

1) Run-Length Encoding Algorithms: We have implemented two variants of RLE, which differ in the way how the number of symbol repetitions is detected. While the first variant proceeds on a byte-by-byte basis, the second variant fixes the first occurrence position of a symbol in the input data and steps through the further input data until a different character is determined. Both use a threshold value of two, i.e. only sequences of two or more identical symbols are replaced by two symbol repetitions, followed by the repetition count field. As the encoded output from both algorithm variants is identical, the same decoding algorithm can be used.

The compiled version of the first variant of the RLE algorithm requires 308 bytes of Flash memory and 58 bytes of RAM. This represents a very small fraction of the resources available on the motes. The resource consumption of the second variant is 490 bytes of Flash memory and 58 bytes of RAM, also being less than 2% of the available resources. As RLE requires very low memory, it can be easily combined with existing applications on motes.

2) Adaptive Huffman Coding: We have founded our evaluation of the adaptive Huffman coding algorithm on an implementation provided by A. Guitton, as presented in [20]. In addition to the normal operation, the version of AHC was



(a) Activity period

(b) Sleep period

Fig. 3. The Porcupine sensor data sets

extended to be fault-tolerant, i.e. packet losses which would result in a false synchronization of the Huffman code tree can be recovered from. Contrary to the static Huffman algorithm, the adaptive variant does not require a prior knowledge of the symbol occurrence frequencies, so code tables need not be transmitted in advance. Instead, in the adaptive Huffman algorithm, both encoder and decoder manage their own tree construction. This complex process must be synchronized and symmetric in order to avoid different correspondences between symbols and codes, which may result in incorrect decoding.

The memory requirements of the AHC implementation on both sender and receiver were evaluated, and yielded a demand for 30,686 bytes of Flash memory and 6,462 bytes of RAM. It is obvious that the Flash memory consumption is just within the limits of the Porcupine platform, but the available RAM is significantly exceeded. To allow an analysis of the AHC algorithm, we have thus modified the source code of the Avrora simulator to assume an increased RAM size of 10 kilobytes for the Mica2 platform, so our simulations could be performed successfully and energy estimates extracted.

IV. Simulation Results

We have performed simulations in two successive steps; in the first series of simulations, a preliminary estimation for best and worst case results was performed for both algorithms by using previously defined data structures. In a second stage, we evaluate the algorithms with sampled Porcupine data sets.

A. Simulation Scenario

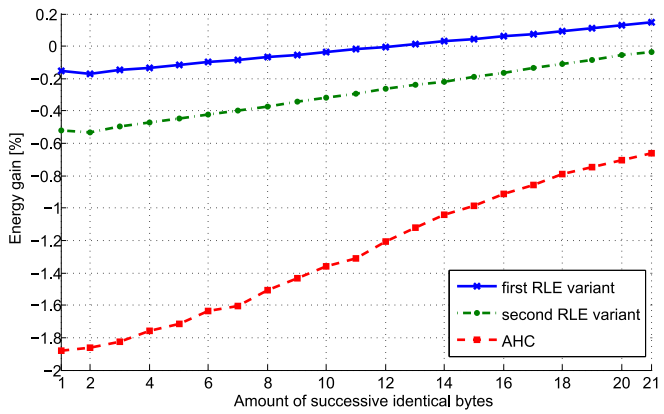
To assess the energy gains achievable by applying data compression at the sender node, we have assumed an ideal (i.e. lossless) one-hop communication channel and used the original Porcupine application, which samples and stores seven accelerometer values per axis. Data is sampled at an interval of 60 millisecondes between successive readings. Once all 21 readings have been taken, the radio is powered up and the packet sent. When the packet has been sent, the radio is immediately stopped for energy saving reasons.

The following energy results are presented in comparison to the reference implementation, which directly forwards the packets to the radio transceiver instead of applying the data compression algorithm first. By taking the difference of both obtained energy results, the energy gains or losses caused by the compression algorithm can be directly compared.

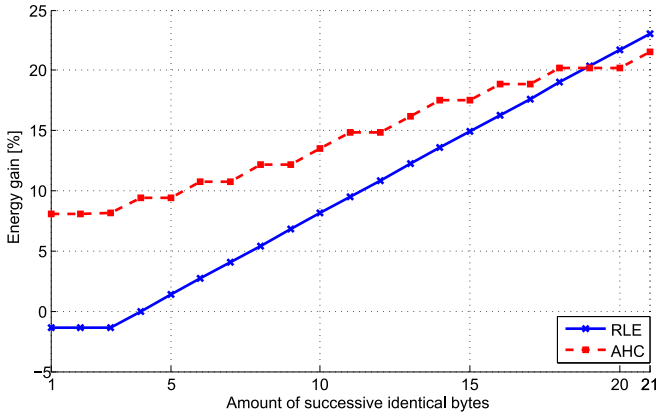
In a first step, we have evaluated the algorithms with a synthetic workload to assess best case and worst case performances. Concisely, for the best case evaluation, we have used input sequences of 21 bytes in length with a defined number of identical characters followed by differing symbols. For the worst case, we have used different specific input sequences for RLE and AHC. In the RLE case, we have used input data with two repetitions of each symbol (`aabbcc...`), so the threshold is met every time, but the repetition count field will contain a value of zero. For the adaptive Huffman coding mechanism, we have used payloads with an equal distribution of all possible input byte values to evaluate the worst case.

To analyze the energy gains under realistic conditions, we have selected two representative sequences of sensor readings from the Porcupine project [10], which are depicted in Fig. 3 and were kindly provided by K. Van Laerhoven. The first selected period is presented in Fig. 3(a), and termed as the *activity* phase in the following sections. With a profile composed of multiple varying acceleration readings, it allows evaluating the algorithms during a period of high wearer activity. The second data set (shown in Fig. 3(b)) represents readings from a phase when the wearer was asleep, and is thus termed the *sleep* phase. It is well suited to obtain energy results for the common case when the wearer's motions are limited and the accelerometer data may thus show a great number of consecutive identical bytes.

We have selected the Porcupine data to evaluate the compression algorithms, as it contains real sensor data which expose good compressibility through the natural characteristic of body movements. In total, 30,316 samples for each of the three dimensions were used in our simulations, equalling a number of 90,948 individual acceleration values. We have segmented the data into 26 chunks of 1,166 readings per axis each, corresponding to 166 sent packets, and analyze the compression gain for each chunk individually. After having



(a) Microcontroller energy gain



(b) Radio energy gain

Fig. 4. Resulting best case energy gains

successively simulated each chunk of data, we show the corresponding mean energy gain in the following figures.

B. Best/Worst Case Energy Results

To assess the best and worst case energy gains, we have configured the simulator to send 166 packets with identical payload and compare the resulting energy gain values.

1) Microcontroller Energy: The microcontroller energy results are depicted in Fig. 4(a) in function of the number of identical symbols within the input. The obtained results show that applying compression to the data leads to MCU energy losses in almost all cases. Only when 13 or more identical symbols are present within the input sequence, the first RLE variant shows a slight energy gain. The energy loss observed in all other cases can be interpreted as such that the energy required by the microcontroller operations for compressing the data is greater than energy saved by not transferring the bytes to the radio transceiver over the SPI bus. With energy losses between 0.2% and 0.54%, RLE requires less microcontroller energy than the AHC coder, which has almost 2% of energy loss in the worst case. We attribute this difference to the complex tree management, which requires a high number of additional microcontroller operations.

Worst case results for the algorithms have been simulated as well. They reach energy losses of 1.3% in the first RLE variant, 2.5% in the second alternative, and 6.8% in the AHC case. As it becomes clear that the first RLE variant consumes less microcontroller energy than the second at all times while producing equally sized output data, we disregard the second variant of our RLE implementation in the following simulations.

2) Radio Energy: In addition to the microcontroller energy estimation, we have used Avrora to evaluate the radio energy consumption of both algorithms. Again, the results obtained after sending 166 packets with the defined payload are illustrated in Fig. 4(b). As both compression algorithms demonstrate energy gains for each simulation, the results confirm the hypothesis that shorter packets require less time on the radio channel, and thus less energy to be sent.

Analyzing both algorithms, the obtained results show that more than 21% of radio energy can be saved by sending compressed packets. Even though this best case, corresponding to 21 successive identical bytes, might be unlikely, the remaining cases show that the achievable energy savings expose a linear dependency on the amount of successive identical bytes at the input. In the RLE case, energy savings are possible when at least four identical bytes are present within the input, because the algorithm requires at least two bytes to indicate the repetition and another one for the repetition count. With less than three identical successive bytes, there is a slight loss in energy as the data are prefixed by the status byte, but remain uncompressed otherwise. In contrast to wrapping the number of symbol repetitions into a repetition count field, AHC replaces each input symbol by the corresponding code, which requires at least a single bit, hence the compression gain curve exposes a smaller slope. The additional status flag byte and a second field indicating the number of bits in the payload add up to the size requirement of AHC encoded packets.

By examining the energy gain of the adaptive Huffman algorithm, it becomes clear that greater savings than for RLE are possible for sequences with a smaller number of symbol

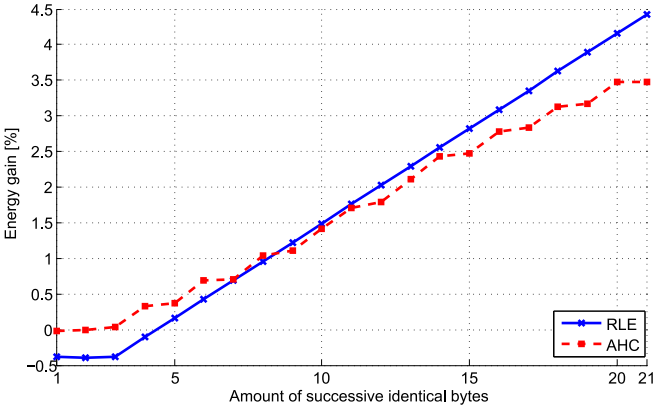
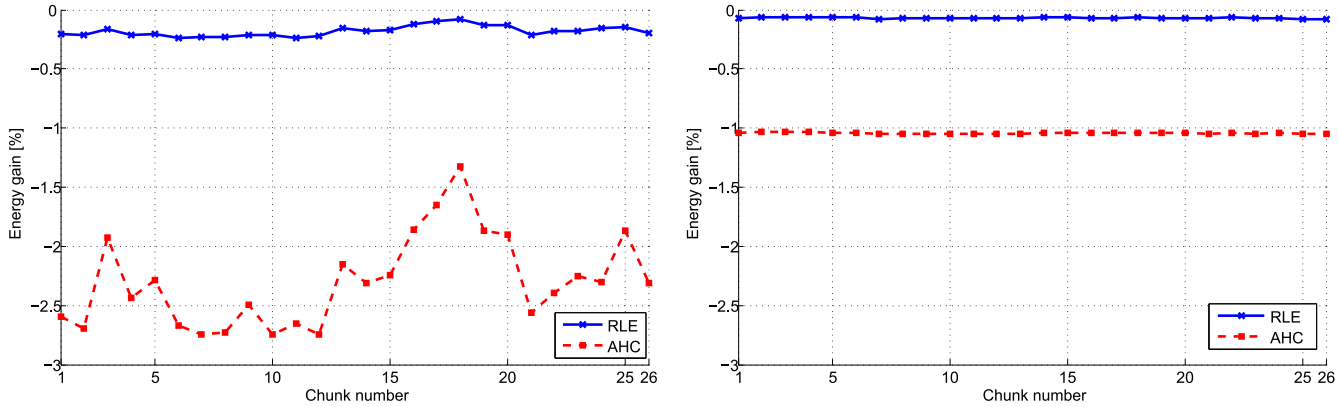
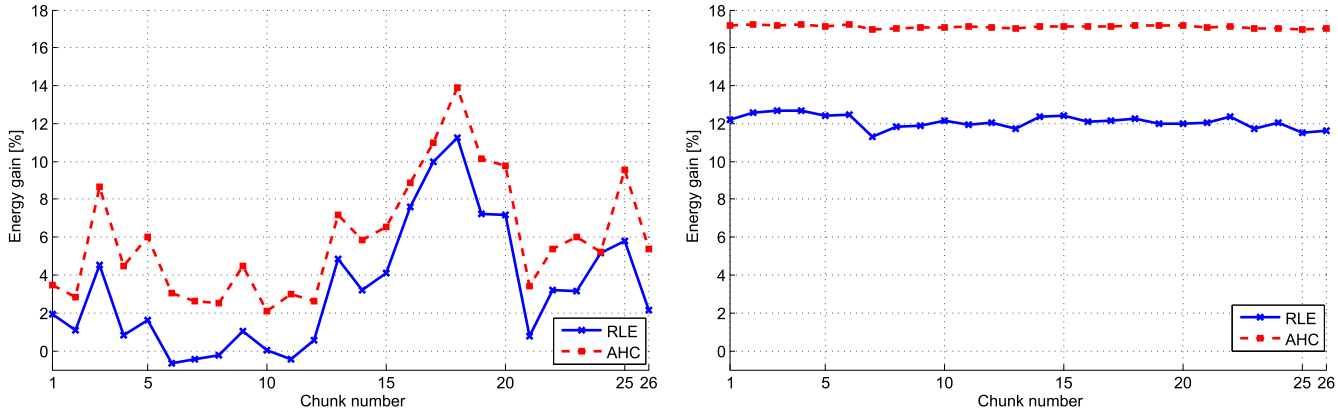


Fig. 5. Overall best case energy gain



(a) Microcontroller energy gain for real data, activity phase

(b) Microcontroller energy gain for real data, sleep phase



(c) Radio energy gain for real data, activity phase

(d) Radio energy gain for real data, sleep phase

Fig. 6. Energy gains for Porcupine data

repetitions. The resulting curve exposes some discrete steps, which result from the bitwise operation of AHC, as it requires terminating padding bits. In summary, each additional symbol repetition in the input data can save 1.3% of the radio energy in the RLE case, and 0.6% when AHC is applied.

3) Overall Energy: The global energy consumption of both algorithms and best/worst case data is illustrated by Fig. 5. As the radio transceiver and MCU are the major power consumers on the mote platform, the resulting curves resemble the addition of both previously determined graphs, weighted by the corresponding duty cycles.

Global energy savings reach up to 4.5% in the RLE case, and up to 3.5% for AHC, respectively. Evaluated with previously determined best and worst case data sets, the results allow establishing that data compression leads to energy savings. After having proved the feasibility of our motivating scenario, we evaluate the implemented algorithms with real sensor data in the following step.

C. Real Data Energy Results

In order to evaluate both Run Length Encoding and the adaptive Huffman algorithm on the microcontroller energy when real data sets are given, we have used the real data from the Porcupine project, and analyzed the algorithms for both the activity and the sleep phases.

1) Microcontroller Energy: By considering the activity period first, the microcontroller energy results illustrated in Fig. 6(a) are obtained. It is obvious that this time energy losses are present with both algorithms due to the highly dynamic and thus less compressible nature of the sensor data. Again, RLE losses range around 0.2%, whereas the losses by the AHC algorithm are more than ten times as high, with average losses of 2.3% and peaks of up to 2.75%. Illustrated by Fig. 6(b), the microcontroller energy consumption during the sleep phase data set follows the same trends: Applying the compression algorithms requires microcontroller energy for both algorithms. However, the improved compressibility of the data results in losses for the RLE algorithm around 0.1%, while AHC leads to around 1% of energy loss.

According to our expectations, the microcontroller energy gain depends on the activity intensity, and is greater in case the data presents a high amount of consecutive identical bytes. Such sequences allow for very small losses, while intense activity of the wearer leads to a greater energy demand, as the accelerometer readings are varying quickly, and fewer byte reductions are hence possible.

2) Radio Energy: To complete the compression algorithm evaluation with real data, their effects on the radio energy consumption were also examined. Depicted in Fig. 6(c), the

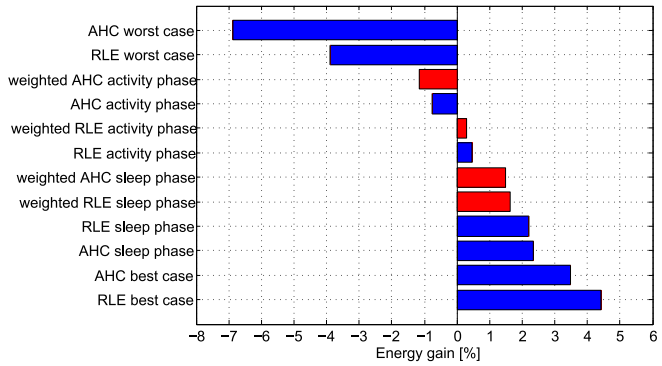


Fig. 7. Overall energy gain for real data

radio energy consumptions confirm the previous observations when using the activity phase data set. With an energy gain of 3.3% on average, the RLE algorithm is outperformed by the AHC, which achieves mean energy gains of 5.9%.

The results for the sleep phase are presented in Fig. 6(d), and confirm the expectations that the radio energy gain is greater during the sleep period than during the activity phase. More precisely, applying the RLE algorithm allows to save 12.1% of energy on average, and the application of the adaptive Huffman algorithm even leads to around 17.1% of radio energy savings, confirming that compressing data allows to reduce the radio transceiver's energy consumption significantly.

3) Overall Energy: The overall simulation results, including radio and microcontroller gains as well as the previously determined best and worst case results, are compared to the reference implementation in Fig. 7. With a mean energy loss of 0.76% during the activity phase and a gain of 2.35% during the sleep period, the AHC algorithm has shown that energy losses can occur when real sensor data is compressed using AHC. In contrast, the RLE algorithm exposes average energy gains of 0.47% in the activity phase and 2.2% in the sleep phase, confirming that RLE allows to globally save energy when applied to Porcupine sensor data.

D. Transferring the Results

To allow for an estimation of the real-world energy savings, we have weighted the MCU energy demands by the factor of 1.416, as determined for the Porcupine platform in our hardware comparison in Section III-C. The corresponding results are shown in Fig. 7. They indicate that the achievable energy gains decrease due to the higher cost of microcontroller operations. In the AHC case, this results in energy losses of 1.14% in the activity phase, while a gain of 1.49% is present in the sleep phase. When RLE is applied, savings are achieved in both phases and range from 0.3% in the activity phase to 1.63% during sleep. In summary, positive energy gains can still be observed for all RLE cases and the sleep phase AHC.

E. Summary

We have compared the energy gain results for all data sets and determined that up to 17.1% of radio energy could be saved when applying AHC on the real Porcupine sensor data, a result close to the best case achieved in simulations with synthetic data. However, these significant savings are counterbalanced by the microcontroller energy demands to maintain the complex Huffman code tree and to transcode the input symbols to a binary output sequence, resulting in overall energy gains ranging from -0.76% to 2.35% on Mica2 hardware. In contrast, the more lightweight RLE implementation requires much less operations to encode symbols on a per-byte basis and always leads to positive energy gains in the range of 0.47% to 2.2% for the real data set.

V. Conclusion and Outlook

In this paper, we have analyzed the achievable energy gains when applying data compression in WSNs. Our simulation results prove that applying data compression may allow saving energy, even if additional microcontroller operations are required, as the corresponding MCU energy demand can be counterbalanced by the radio energy gain, which results from transmitting smaller packets.

We have evaluated the applicability of both run-length encoding and adaptive Huffman coding on sensor network nodes. While the RLE implementation consumes a very small fraction of the available RAM and Flash memory, the adaptive Huffman algorithm requires a major amount of Flash memory and even exceeds the RAM available on the platform. The gains achievable by the AHC algorithm did however not significantly exceed the energy preserved when applying RLE.

Although having confined our discussion to a single application with two data sets only, our results indicate that data compression is a feasible way to reduce a WSN node's energy consumption, and thus extend its lifetime. However, our findings also show that adequate solutions designed for sensor network node hardware are necessary.

When considering a whole sensor network, transmitting compressed packets also has a beneficial effect on the overall network, as less traffic volume needs to be forwarded in a multi-hop WSN setting, leading to an extended network lifetime as well. We anticipate even greater energy savings when larger payload sizes are used, as the fixed-size packet headers present a smaller fraction of the overall packet size in such cases.

A. Outlook

We have performed our simulations using the Avrora simulator, as the energy profile of the Mica2's MCU (weighted by a constant factor) presented a good estimate for the Microchip MCU on the Porcupine platform. To however analyze the effects of data compression on different platforms, we target to perform additional simulations using different tools, such as MSPsim [21]. To verify our findings on real hardware, we also target to complement our simulations by real experiments.

Besides, the characteristics of the B-MAC protocol, especially the long preamble when sending a packet [4], also have an impact on the achievable energy savings. We therefore aim to perform further analyses in conjunction with different MAC layer protocols.

ACKNOWLEDGMENT

Our thanks go to Kristof Van Laerhoven for supplying us the Porcupine data sets, as well as to Alexandre Guitton, who kindly supported our analysis by providing his implementation of the fault-tolerant AHC algorithm. Furthermore, we would like to thank Parag S. Mogre for the fruitful discussions.

REFERENCES

[1] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors," *Communications of the ACM*, vol. 43, no. 5, 2000.

[2] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "Energy-Aware Wireless Microsensor Networks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, 2002.

[3] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.

[4] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[5] V. Shnayder, B.-R. Chen, K. Lorincz, T. R. F. Fulford-Jones, and M. Welsh, "Sensor Networks for Medical Care," Diversion of Engineering and Applied Sciences, Harvard University, Tech. Rep. TR-08-05, 2005.

[6] J. A. Stankovic, Q. Cao, T. Doan, L. Fang, Z. He, R. Kiran, S. Lin, S. Son, R. Stoleru, and A. Wood, "Wireless Sensor Networks for In-Home Healthcare: Potential and Challenges," in *High Confidence Medical Device Software and Systems (HCMDSS) Workshop*, 2005.

[7] B. Krishnamachari, D. Estrin, and S. Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks," in *Proceedings of the International Workshop on Distributed Event-Based Systems (DEBS)*, 2002.

[8] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944 (Proposed Standard), 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4944.txt

[9] K. Barr and K. Asanović, "Energy Aware Lossless Data Compression," in *Proceedings of the First International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2003.

[10] K. V. Laerhoven, H.-W. Gellersen, and Y. G. Malliaris, "Long-Term Activity Monitoring with a Wearable Sensor Node," in *Workshop on Wearable and Implantable Body Sensor Networks (BSN)*, 2006.

[11] B. L. Titzer, D. K. Lee, and J. Palsberg, "Avrora: Scalable Sensor Network Simulation with Precise Timing," in *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN)*, 2005.

[12] C. M. Sadler and M. Martonosi, "Data Compression Algorithms for Energy-Constrained Devices in Delay Tolerant Networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[13] 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)*, 2008.

[14] 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)*, 2005.

[15] A. Reinhardt, M. Hollick, and R. Steinmetz, "Stream-oriented Lossless Packet Compression in Wireless Sensor Networks," in *Proceedings of the Sixth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2009.

[16] O. Landsiedel, K. Wehrle, B. L. Titzer, and J. Palsberg, "Enabling Detailed Modeling and Analysis of Sensor Networks," *Praxis der Informationsverarbeitung und Kommunikation*, vol. 28, no. 2, 2005.

[17] O. Landsiedel, K. Wehrle, and S. Götz, "Accurate Prediction of Power Consumption in Sensor Networks," in *Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors (EmNetS-II)*, 2005.

[18] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System Architecture Directions for Network Sensors," in *Proceedings of the 10th Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2000.

[19] *MICA2 Datasheet*, Crossbow Technology, http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf.

[20] 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)*, 2008.

[21] 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 Workshop on Wireless Sensor Networks (EWSN)*, 2009.