Andreas Reinhardt, Delphine Christin, Ralf Steinmetz: Pre-Allocating Code Mappings for Energy-Efficient Data Encoding in Wireless Sensor Networks. In Proceedings of the 9th IEEE PerCom Workshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS), pp. 594-599, Mar. 2013, ISBN 978-1-4673-5076-1

Pre-Allocating Code Mappings for Energy-Efficient Data Encoding in Wireless Sensor Networks

Andreas Reinhardt*, Delphine Christin‡, Ralf Steinmetz*

*Multimedia Communications Lab, Technische Universität Darmstadt
Rundeturmstr. 10, 64283 Darmstadt, Germany
{andreas.reinhardt, ralf.steinmetz}@kom.tu-darmstadt.de

‡Secure Mobile Networking Lab, Technische Universität Darmstadt
Mornewegstr. 32, 64293 Darmstadt, Germany
delphine.christin@seemoo.tu-darmstadt.de

***Abstract*—Energy is a scarce resource on battery-powered wireless sensor nodes, and wireless communication represents the major consumer of electric energy on most current platforms. Reducing the number and size of radio transmissions thus represents a viable approach to save energy and extend a node's operational time. In the domain of pervasive computing, where a periodic reporting of data (e.g., a user's vital parameters) is often being used, packets cannot always be simply omitted from transmission. Even if the contained data have not changed, these periodically transmitted message double as beacons to indicate that the sensor node has not run out of energy. Hence, reducing the sizes of transmitted messages remains the only available solution to achieve energy savings in such sensor networks. In this paper, we show how pre-computed codebooks can be used to encode messages in an energy-efficient way and thus reduce the size of the transmitted packets. We present how we extract these code mappings from real-world data, and describe how packets are encoded prior to their transmission in order to reduce the incurred energy demand. We practically assess the energy demand on TelosB nodes and prove that up to 17.2% of energy can be saved when our approach is applied.**

I. Introduction

Wireless Sensor Networks (*WSN*s) used in pervasive computing applications are often configured to periodically report their sensor readings to a sink, where the processing of the data takes place. Therefore, the deployed devices (*mote*s) regularly wake up from a low-power sleep mode, collect readings from the attached sensors, transmit these data towards a sink, and return to their sleep state. This duty-cycling ensures a low energy demand of the systems during their state of inactivity, yet the need for wireless communication still results in a measurable overhead during the active phases. Especially in applications where the periodic status messages serve as indicators that a node has not run out of energy, and thus cannot be omitted, the only remaining option to reduce a node's energy demand is to cut down on the size of transmitted packets. Packet size reductions can be achieved by means of encoding the packet contents, with the use of data compression being a prominent example for packet encoding algorithms.

Although it has been shown that data compression can be performed on motes [1], [2], [3], a number of drawbacks limit their applicability in practical deployments. Firstly, many compression algorithms rely on a mutually established state, e.g., a dictionary of frequently used symbols. The lossy nature of wireless communication channels in WSNs, however, necessitates additional means to ensure that all updates of this state information are successfully received, which in turn introduces an additional energy overhead. Secondly, compression algorithms from the domain of desktop computing often require a large amount of memory, because their primary objective is to achieve high compression gains. The limited resources of motes often render these algorithms inapplicable, or necessitate major modifications to the code. Finally, in order to achieve high compression gains, the used algorithms need to be adapted to the characteristics of the transferred data. This puts an additional burden on application developers, who are inherently expected to be familiar with the actual value ranges of the collected readings in order to write efficient compression code.

In this paper, we hence propose to use pre-allocated code mappings (*PACmap*s) to locally encode data on deployed motes. PACmaps are lookup tables, which have been extracted based on the characteristics of sensor readings from different physical sensors, and are stored on all sensor nodes prior to their deployment. By analyzing the compressibility of outbound packets using each of the available PACmaps, the best suited code mapping can be selected and used to encode all further packets. As all required state information is pre-deployed, our approach is well applicable over lossy channels. We analyze data from existing sensor network deployments as well as own collected data with regard to their symbol distributions and derive corresponding PACmaps (cf. Sec. II). We then introduce our system architecture in Sec. III. The real-world applicability of our implementation is proven in Sec. IV, along with analyses of its achievable compression gains and the energy demand of each operation measured in an experimental setting. We summarize related work in Sec. V and conclude this paper in Sec. VI.

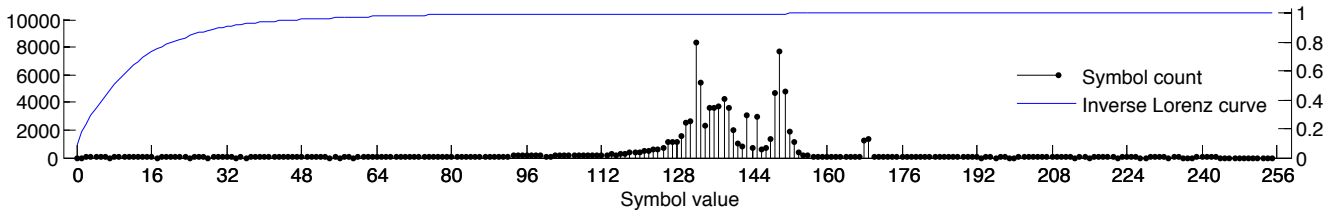

Figure 1. Symbol distribution of the Porcupine data set

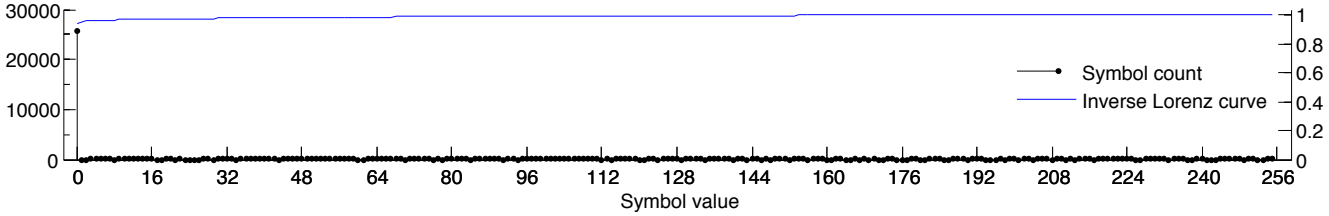

Figure 2. Symbol distribution of the Glacsweb data set

II. Generation of PACmaps

Because PACmaps are pre-installed on all motes before the deployment, their definition needs to be done prior to their actual usage. To this end, we have collected several data traces from existing and self-deployed WSNs, and analyze their symbol distributions in this section. In a subsequent step, we describe how the PACmaps are extracted from the input sequences.

A. Data Set Characteristics

For the definition of PACmaps, we have analyzed the symbol distributions of three existing real-world data sets, namely the PermaSense [4], Glacsweb [5], and Porcupine [6] deployments. While the former two deployments are focused on monitoring environmental parameters in permafrost and glacier areas, the latter project is based on wrist-worn acceleration sensors to detect human activities. Besides using the data sets from these three existing deployments, we have deployed a WSN in our pervasive computing lab that was configured to collect motion, sound, temperature, and humidity readings in an office space, as well as a sensor for the power consumption of a desktop PC and two LCD monitors. In order to get an impression of the data characteristics, we show the symbol distributions of the Porcupine data set in Fig. 1 and Glacsweb in Fig. 2. All possible 8-bit input values are interpreted as unsigned values and shown on the x-axis, and the black stem plot indicates the absolute occurrence count of each symbol with regard to the left y-axis. In addition to the occurrence frequency of each symbol in the input trace, we show the inverse Lorenz curve, which can be interpreted as the cumulative distribution function of the sorted occurrence frequency list, on the right y-axis. The deviation of the inverse Lorenz curve from the line through the origin can be seen as a visual indicator of the deviation from a uniform distribution and thus as a direct indicator for the compressibility of the data.

In Table I, we show the actual number of used symbols in each of the data sets, along with its Shannon entropy, which is calculated as

$$H = -\sum_{i=0}^{255} P(x_i) log_2 P(x_i) \qquad \text{(in } bits\text{)}$$

It becomes clear from the table that all considered traces deviate from a uniform distribution of symbols, in which the entropy value would be 8 bits per symbol. Some of the analyzed traces have shown entropy values as low as 0.65 bits per symbol, despite the fact the 8 bits are required to represent them in the radio packet, whereas the PermaSense data is significantly less predictable, but still features an entropy of only 6.73 bits per symbol.

B. Code Mapping Extraction

Having observed that many data sets have an entropy value of less than eight bits per symbol, a compression of the data by means of entropy coding, e.g., arithmetic coding [7] or Huffman codes [8], can be expected to lead to significant size reductions. Entropy coders analyze the distribution of symbols in the input sequence and allocate shorter codes to more frequently occurring symbols and vice versa.

We have used a Huffman coder to generate code mappings, i.e., PACmaps, from the data traces in a process identical to the establishment of a Huffman tree. Each PACmap is comprised of two elements, the first one being

Table I
Statistics of the used input traces

Data set	# of used symbols	Entropy (bits)
Glacsweb	185	0.65
PermaSense	256	6.73
Porcupine	220	5.19
Humidity	9	2.13
Temperature	4	1.58
Noise level	76	2.68
Power consumption	146	5.36
Motion	254	4.88

Table II
SIZE AND DISTRIBUTIONS OF THE CODE MAPPINGS

Input data	PACmap size	Code length (bits)		
		Min	Avg.	Max.
Glacsweb	836 bytes	1	10.11	12
PermaSense	799 bytes	4	8.96	10
Porcupine	909 bytes	3	12.39	16
Humidity	988 bytes	1	14.88	16
Temperature	861 bytes	1	10.88	11
Noise level	892 bytes	2	11.86	13
Power consumption	839 bytes	3	10.23	12
Motion	837 bytes	3	10.15	13

a bit array `uint8_t codes[]`, in which all Huffman codes are concatenated to achieve a small memory footprint. Accordingly, a supplementary list `uint16_t offsets[256]` is required which maps each of the possible input symbols to the corresponding offset in the `codes` bit array. A separate length definition is not necessary, as the length of the code is bounded by the starting offset of the subsequent symbol. In order to ensure that all symbols are contained in the list, the occurrence counter for unused symbols is set to a value of 1 prior to the extraction of the PACmap.

The resulting codebook sizes are shown in Table II, which also highlights the minimum, average, and maximum code lengths. The size comparison shows that less than one kilobyte of memory is required to store each of the PACmaps, which is especially important in the domain of pervasive computing, where tight resource constraints are often present. From the table, it becomes clear that the resulting code lengths strongly depend on the distribution of the input data. For example, the high number of occurrences of the '0' symbol in the Glacsweb trace (cf. Fig. 2) leads to the assignment of an output code of only one bit in length.

In WSN deployments, data are continually being collected, and no a priori knowledge about the exact characteristics of the data to be collected in the future exists (otherwise all future packets would be completely redundant). The code mappings are thus extracted prior to the actual operation, and hence only reflect the average characteristics of the input data. While this decreases their efficiency when characteristics of the sensor data change over time, no updates to the code tables are required. Especially as the wireless communication channel in WSNs is often lossy [9], suitable transport protocols would otherwise have to be employed to ensure that code information is up-to-date at both sender and receiver. Such transport protocols often introduce a large overhead in terms of energy expenditure, which may even exceed the energy savings introduced by encoding the data. The presented PACmaps are static, such that the impact of lossy channels is effectively mitigated.

III. DESIGN OF THE DATA ENCODING FRAMEWORK

Traditional sensor network applications commonly define packet structures statically at compile time due to efficiency reasons [10]. In such applications, transmitted packets thus always have the same size and thus the same energy demand for their transmission, despite possible redundancies in their content. As our analysis in the preceding section has shown that redundancies exist in all collected data sets, our framework precedes the wireless communication with a data encoding step. Our approach encodes outbound sensor readings using PACmaps, which have been deployed onto motes prior to their installation in the field.

A. System Architecture and Algorithm Flow

We have designed our encoder as a TinyOS module [10], which can be easily integrated into existing applications. Its general control flow is visualized in Fig. 3, and can be summarized in the following two steps.

- In the initialization phase as well as when degraded encoding gains are noticed during its operation, our framework performs the *PACmap selection* step.
- Once a PACmap has been selected to encode all further data, the framework caters to the *packet encoding step* and returns the compressed packet along with an identifier of the used PACmap to the invoking application.

Its interface is simple, as its core method takes an outbound radio packet as its input, and returns an encoded data packet as well as an indicator whether the result has been compressed and which code mapping has been used. The system first checks whether a PACmap has already been selected in a prior selection process (cf. Sec. III-B). If this is the case, the given map is used to compress the data, otherwise the selection process is triggered and all available code mappings are evaluated with regard to

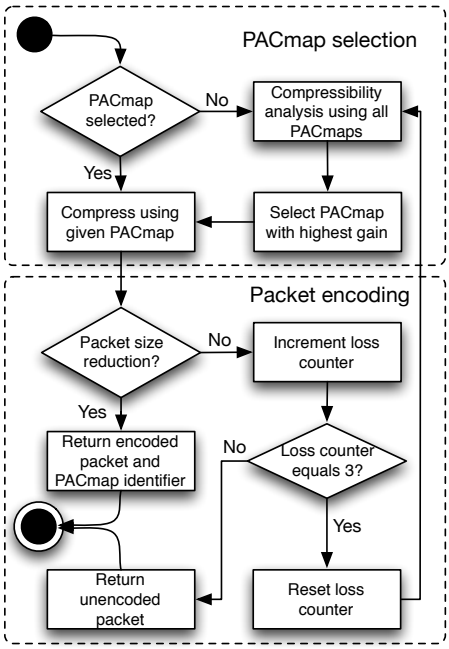

Figure 3. Control flow in the PACmap encoding architecture

the achieved compression gains. Following the compression step, a gain monitor checks whether positive compression gains are achieved and triggers the selection process again in case the packet size increases after the encoding step. The proposed solution uses a lossless encoding mechanism and is agnostic to the actual type of application data.

B. Optimal Map Selection

All nodes that run PACmap come with a pre-defined set of code mappings that have been extracted from data from existing deployments. These code mappings have already been introduced in Sec. II, in which we have collected and analyzed data from multiple sources and extracted corresponding PACmaps. Depending on the actual types of sensors and the expected application domain, this set can be extended or adapted to reflect the anticipated data types.

When several code mappings are available, the best PACmap, i.e., the one that yields highest compression gains, needs to be found. This is done by compressing the first packet after the node's reboot using all available mappings. The PACmap with the highest average compression gain is selected as the codebook for all subsequent transmissions. In case changes to the physical phenomenon have occurred, the selected map becomes less efficient with regard to its encoding gain. An encoding gain monitor thus constantly checks whether the encoded packet is smaller than its unencoded representation. When size increments have been recognized for three packets, the PACmap selection process is repeated.

C. Packet Transmission

In order to successfully decode a packet, information about whether it is present in plain or encoded representation is required. In our solution, we have decided to use the active message ID field of the TinyOS operating system in order to distinguish between regular (i.e., uncompressed) and encoded transmissions. This way, no additional byte in the payload is required to identify uncompressed transmissions. All encoded packets are prefixed by a flag that identifies the PACmap that has been used in the encoding step. Due to the limited reliability of WSN channels, upon which we have designed our solution, the stateless PACmap requires the identifier of the used map to be transferred in every packet. Hence, a single additional byte prefixes each encoded packet and informs the recipient how to decode the data.

IV. Performance Evaluation

In this section, we evaluate the achievable encoding gains when our proposed PACmap-based data encoding architecture is used. Subsequently, the energy demand of our implementation is analyzed using a high-resolution energy profiling tool and compared to an operation without data encoding.

A. Evaluation Setup

In order to assess the energy gains under realistic conditions, we have chosen the widely used TelosB platform [11] as the basis for our implementation. For the analysis of real-world energy demands, we have used a Hitex PowerScale unit [12] with ACM probe, which can measure currents at a resolution of 0.2μA. The sampling rate of the PowerScale base unit is 100kHz, such that all operations can be analyzed up to a resolution of 10μs. A trigger pin of the PowerScale was used to mark the beginning and end of individual operation phases. Because our PACmap approach focuses on the optimization of a node's local energy demand, only a single transmitting node has been regarded in our evaluation. The node has been programmed with all of the eight PACmaps that were presented in Sec. II.

B. Encoding Gains

In order to assess the output sizes of the encoder, we have segmented each of the input data traces into chunks of 30 bytes size, which represents an average value of the packet sizes used in WSN deployments. The resulting packets were compressed with each of the PACmaps presented in Sec. II. The achieved results are visualized in Fig. 4, which shows the average output size as well as the encountered minimum and maximum values. From the figure, it becomes clear that each PACmap manages to achieve positive encoding gains when applied to the matching input data. Despite the fact that an additional byte is required in each encoded packet to indicate which PACmap has been used during the encoding step, size reductions by up to 83.3% (i.e., a packet size reduction from 30 bytes to a mere 4 bytes plus 1 byte for the prefix) have been achieved. In fact, the PermaSense PACmap also shows positive encoding gains when applied to the Glacsweb data. In all other cases, however, losses are encountered in the average case, with encoded packet sizes of up to 60 bytes, i.e., twice the size of the unencoded data. As a result, PACmaps necessarily need to be adapted to the symbol distributions of the anticipated data streams.

C. Compression Time and Energy Consumption

In order to yield energy savings, the energy required for local computation must not exceed the energy saved by transmitting shorter packets over the wireless link. We thus analyze this tradeoff between local computation and wireless transmission next. As introduced in Sec. III, the system operation is composed of two major steps, namely the selection of the best suited PACmap, followed by the actual encoding of outbound packets. In the previous size gain analysis, we have omitted the first step and hard-coded the application to use a particular codebook.

In this evaluation, we thus investigate the energy overhead introduced by the PACmap selection process. Therefore, we have used ten packets of input data for the assessment of the best-suited PACmap, and quantified the average energy

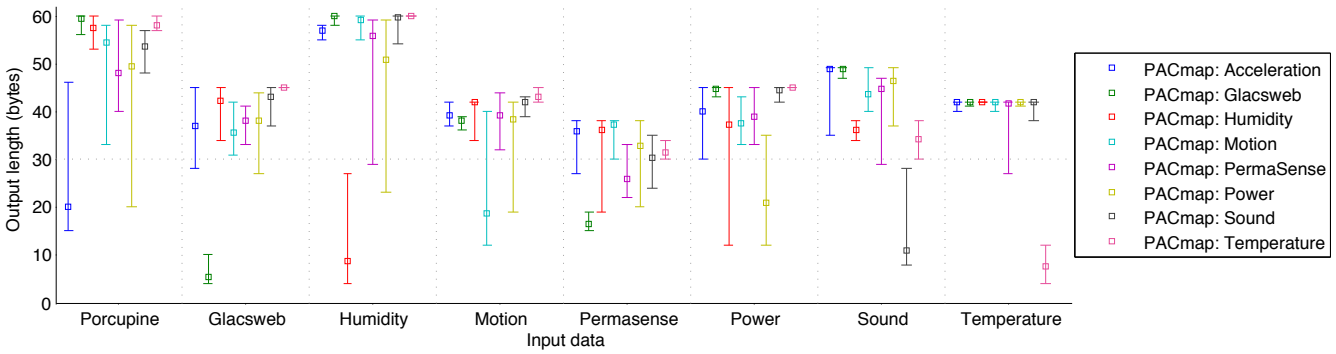

Figure 4. Compression gains for the combinations of input data sets and PACmaps (the input packet length was set to 30 bytes)

required to conduct this analysis by means of the Hitex PowerScale. A typical trace of the TelosB's power consumption is shown in Fig. 5, in which the three *U1* triggers indicate the PACmap selection, data encoding, and wireless transmission steps, respectively. The end of the transmission is signaled by the *D1* trigger. The actual PACmap selection process has been determined to consume approximately 800μJ of energy. Of these, 735μJ can be attributed to the compression of each data set (accurate figures of the processing overhead for each type of input data are shown in Table III), and an additional 65μJ are required by the actual selection process.

In a subsequent experiment, we have analyzed the average energy consumption for the encoding of each data trace using the corresponding PACmap. Therefore, we have activated the radio transceiver upon the completion of the encoding step, transmitted the packet, and put the transceiver to sleep mode immediately afterwards. The resulting average energy budgets for the transmission of encoded packets are also shown in Table III and compared to the unencoded reference transmission. The correlation between the average output size (cf. Fig. 4) and the energy demand for the transmission of packets becomes clear from the table. Based on the observation that energy savings of up to 17.2% are given when applying the PACmap approach, the initial energy overhead for the PACmap selection is easily compensated. In fact, in case of the Glacsweb data set, the energy expenditure for the PACmap selection is amortized after just two packet transmissions, whilst approximately 80 packet transmissions are required for the less compressible PermaSense data set.

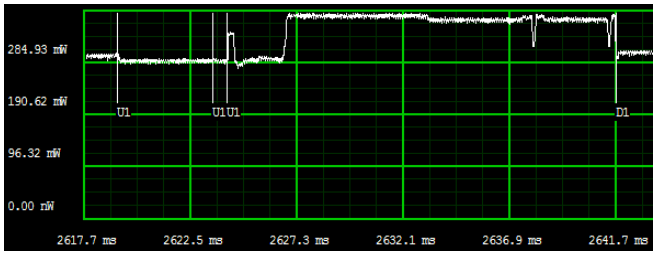

Figure 5. Excerpt of the power consumption during typical operation

V. Related Work

Many research groups have addressed the challenge of optimizing the tradeoff between local computation and wireless transmission. In [1], Sadler and Martonosi have shown that the transmission of a single byte over one hop consumes an amount of energy that is equivalent to performing several thousand instructions on a Texas Instruments MSP430 microcontroller. In their paper, they hence propose an adapted LZW compression algorithm, which uses retransmissions in order to ensure that updates to the code dictionary are conducted simultaneously at both the sender and receiver. Guitton et al. [13] have also confirmed the need for data compression and applied a modified version of the adaptive Huffman coding algorithm on motes. Again, in order to cope with the unreliable wireless channel character, retransmissions and block acknowledgments were added such that losses can be recovered from.

In [2], an approach to compress firmware updates prior to their wireless distribution has been presented by Tsiftes et al. The resulting SBZIP algorithm is a derivative of BZIP2 and adapted to the resource constraints of motes. However, the implementation of SBZIP addresses an opposite challenge, as only the decompression step is implemented on motes, whereas the data compression takes place at the sink. Means to reduce the network's total energy consumption by applying the Slepian-Wolf coding theorem in a low-complexity implementation have been presented by Chou et

Table III
Energy demand for encoding individual packets

Input data	Energy demand		Energy gain
	Processing	Sending	
Reference	–	4.49 mJ	0%
Glacsweb	50 μJ	3.68 mJ	17.2%
PermaSense	140 μJ	4.39 mJ	2.2%
Porcupine	100 μJ	4.17 mJ	7.1%
Humidity	80 μJ	3.81 mJ	15.1%
Temperature	75 μJ	3.77 mJ	16.0%
Noise level	100 μJ	3.88 mJ	13.6%
Power consumption	100 μJ	4.17 mJ	7.1%
Motion	90 μJ	4.15 mJ	7.6%

al. in [14]. In their implementation, no communication overhead between nodes is introduced as long as the correlation between the collected sensor data is known. However, the major focus of this approach is to reduce the overall number of packet transmissions. The approach is hence orthogonal to our concept of reducing the sizes of packets and can be used supplementary. In our previous work [15], we have presented the stateful Squeeze.KOM compression layer as an architectural element for sensor network nodes, which applies differential coding to outbound packets. Similarly, an adapted version of adaptive Huffman coding was presented in [3] with a special focus on the low resource consumption and the energy gains of the implementation. Due to their stateful character, however, both approaches require additional means to recover from packet losses.

In summary, most of the existing approaches rely on stateful solutions whose performances are severely impacted when packets are lost. Often, supplementary mechanisms are used to recover from packet losses, which however introduce an additional energy demand. We are not aware of any previous work that discusses the energy efficiency of stateless packet encoding techniques in detail.

VI. Conclusion

We have presented a stateless data encoding scheme based on pre-allocated code mappings as a lightweight approach to reduce the size of data packets prior to their transmission. In contrast to existing stateful approaches, no supplementary mechanisms to ensure the synchronous exchange of state information are required, leading to an improved energy efficiency. Our PACmap approach has been specifically designed for resource-constrained node platforms, and uses code mappings that have been extracted from the characteristics of real-world data. When applied, only a slight overhead to assess the best PACmap is introduced, which is easily compensated by the subsequent shorter transmission durations of the encoded packets. In total, energy savings of up to 17.2% could be achieved when our collected real-world data traces were compressed using the PACmap approach. Especially in pervasive computing applications, in which periodic status messages serve as indicators for the availability of a mote, the use of PACmaps is a viable approach to preserve energy and thus extend the network's lifetime.

References

[1] 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.

[2] 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.

[3] 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)*, 2010.

[4] 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 under Extreme Conditions," in *Proceedings of the 8th ACM/IEEE Intl Conf. on Information Processing in Sensor Networks (IPSN/SPOTS)*, 2009.

[5] K. Martinez, R. Ong, and J. Hart, "Glacsweb: A Sensor Network for Hostile Environments," in *Proceedings of the 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, 2004.

[6] K. V. Laerhoven, D. Kilian, and B. Schiele, "Using Rhythm Awareness in Long-Term Activity Recognition," in *Proceedings of the 11th International Symposium on Wearable Computers (ISWC)*, 2009.

[7] J. Rissanen and G. G. Langdon, "Arithmetic Coding," *IBM Journal of Research and Development*, vol. 23, no. 2, 1979.

[8] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, vol. 40, no. 9, 1952.

[9] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis, "An Empirical Study of Low-power Wireless," *ACM Transactions on Sensor Networks*, vol. 6, no. 2, 2010.

[10] 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.

[11] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling Ultra-Low Power Wireless Research," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005.

[12] Hitex Development Tools, "PowerScale with ACM Technology," Online: http://www.hitex.com/index.php?id=powerscale, 2012.

[13] 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.

[14] 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)*, 2003.

[15] 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)*, 2009.