THE HITCHHIKER'S GUIDE TO CHOOSING THE COMPRESSION ALGORITHM FOR YOUR SMART METER DATA

Martin Ringwelski[†], *Christian Renner*[†], *Andreas Reinhardt*[‡], *Andreas Weigel*[†], *Volker Turau*[†]

[†]Hamburg University of Technology
Institute of Telematics
Hamburg, Germany

[‡]Technische Universität Darmstadt
Multimedia Communications Lab
Darmstadt, Germany

Index Terms— smart metering, lossless data compression, DIN EN-62056-21

ABSTRACT

Smart meters are increasingly penetrating the market, resulting in enormous data volumes to be communicated. In many cases, embedded devices collect the metering data and transmit them wirelessly to achieve cheap and facile deployment. Bandwidth is yet scarce and transmission occupies the spectrum. Smart meter data should hence be compressed prior to transmission. Here, solutions for personal computers are not applicable, as they are too resource-demanding. In this paper, we propose four lossless compression algorithms for smart meters. We analyze processing time and compression gains and compare the results with five off-the-shelf compression algorithms. We show that excellent compression gains can be achieved when investing a moderate amount of memory. A discussion of the suitability of the algorithms for different kinds of metering data is presented.

1. INTRODUCTION

A massive cost pressure is entailed by the expansion of the grid in order to cope with occurring peak loads. Not managing to keep up with the customers' electricity demand bears the threat of overloading the distribution network, which can even result in blackouts. This risk can be reduced by offering flexible, time-dependent electricity tariffs, which take the current load situation into account. Customers receive monetary benefits by shifting delayable tasks to time periods during which the grid is well below its capacity limits. Running dish washers or washing machines at times with low energy prices also contributes to the stability of the grid. Likewise, smart metering helps customers to reduce their bill by identifying heavy electricity consumers, since they allow tracing consumption at fine granularity. This in turn reduces the overall load on the grid when energy-aware customers replace energy-inefficient appliances by state-of-the-art models.

However, smart metering only pays off if smart meters are inexpensive devices. As their installation location within a building's distribution board typically lacks access to communication networks, a dedicated communication infrastructure is required to communicate electricity readings to the utility. One approach is to equip smart meters with GSM or UMTS modules, which has two considerable drawbacks. Firstly, this imposes non-negligible hardware plus data transmission cost. Secondly, smart meters are frequently installed in cellars or rooms with poor GSM and UMTS connectivity. A second approach—and the one followed in this paper—is to use wireless sensor network technology, where meters form a mesh network and transport meter data to a data concentrator. This approach is generally inexpensive and facilitates wireless communication by using low-frequency radio devices. Yet, bandwidth constraints and legal requirements of channel utilization must be considered. Meeting these issues restricts the size of the network, while larger networks promise lower costs, since less data concentrators are needed.

For both approaches, reducing the network load decreases costs. The transmission of data deltas or the application of lossy compression is yet not an option due to legal requirements in many countries, such as Germany. Lossless data compression is thus needed to maintain high temporal data resolution and to not diminish the opportunities of smart metering. Although powerful compression algorithms exist for personal computers, they have too large memory footprints and computation demands for resource-constrained smart meter hardware. Exploiting data patterns for compression is tempting but comes at additional development cost. In any case, the result is likely to be a highly integrated solution, only applicable for one particular smart meter application protocol.

In this paper, we present lossless compression algorithms that are tailored to the needs of smart metering hardware but do not make assumptions about the characteristics of the collected data. They are hence compatible to a wide range of smart meter data formats. We evaluate these algorithms using two different sets of real smart meter data and compare the compression results to those achievable with off-the-shelf compression algorithms. To prove the practical merit of the presented algorithms, we analyze and compare their memory footprint and execution time on sensor node hardware.

2. RELATED WORK

The increasing distribution of smart metering devices [1] leads to the availability of power consumption traces at an unprecedented temporal resolution. One predominantly envisioned application domain is the adaptation to smart markets with highly flexible tariffs [2], since this domain strongly relies on the availability of consumption data collected in a near real-time manner. Recent research results however present further opportunities that rely on information about a household's power consumption behavior, e.g., the detection of household activities [3] or the support of ambient assisted living [4]. Although work on capturing and analyzing the power consumption of buildings and appliances has already been presented in the late 1980s ([5, 6]), the optimization of the actual communication of sensor data between the meter and a processing node has not been analyzed in detail to date. We dedicatedly address this shortcoming by presenting approaches to compress smart meter data prior to transmission.

While various data compression solutions for personal computers, such as LZ77 [7] or LZW [8], have been presented several decades ago, the field of data compression on embedded sensing systems has only recently been addressed in detail. The cardinal difference between these two worlds, however, is their objective; on personal computers, the sole target is a reduction of the file size. In contrast, embedded systems commonly target the minimization of their energy balance, i.e., they need to carefully weigh up computation and communication against each other ([9, 10]). Many smart metering devices, such as smart water meters, are likely to be battery-powered, since no other power supply may be available. Compression algorithms hence need to be sufficiently lightweight to operate even on small, resource-constrained devices. Existing algorithms have been adapted to embedded systems, e.g., the SBZIP algorithm [11], a derivative of BZIP2. Likewise, Sadler and Martonosi [12] have proposed RT-LZW, a special adaptation of the LZW algorithm. Solutions specifically tailored to embedded systems include the extraction of linearized trends in the data [13] and the use of Kalman filters for packet predictions [14]. To the best of our knowledge, the applicability of data compression on power consumption readings has only been analyzed for two application-agnostic compression algorithms in [15].

Existing projects that collect power traces and make them available for research purposes include the REDD dataset [16] and the *tracebase* project[1]. Here, readings are collected in intervals of approximately one second, resulting in around 86 400 readings per day. Under the assumption that each reading requires at least 16 bits to be represented, this equals daily traffic of between 70 and 175 kB. This figure clearly motivates the application of data compression in order to reduce the utilization of the communication channel and to remove redundancies in the input data prior to their transmission.

3. COMPRESSION ALGORITHMS

For the compression of smart meter data, we have derived four lossless data compression approaches. They have been particularly designed with regard to their applicability on embedded systems, i.e., a small footprint both in terms of program memory and RAM as well as low processing time. Besides presenting their design in more detail, we introduce the existing compression algorithms that we have used as a reference.

3.1. Adaptive Trimmed Huffman Coding

The Adaptive Trimmed Huffman (ATH) method is described in [17] and was specifically developed for its application in energy-constrained wireless sensor networks. It is an adaptive entropy coding scheme, i.e., it adapts to the symbol distribution during runtime. A restriction of the maximum Huffman code tree size is imposed in order to reduce the algorithm's memory demand. Hence, an additional prefix bit signals whether the following sequence refers to a symbol stored in the tree, or if it is an unencoded symbol.

3.2. Adaptive Markov Chain Huffman Coding

The Markov Chain Huffman (MCH) coding method is based on determining the transition probabilities of successive symbols. For each symbol in the input sequence, MCH establishes a Huffman tree [18] for all possible successor symbols. It is thus well suited for sequences in which strong interdependencies between symbols exist. For application in smart metering, we have slightly extended this concept of MCH coding, in which the underlying data must be known in advance. We term this resulting scheme, which bears the possibility to adapt to the input sequence, adaptive Markov Chain Huffman (AMCH) coding. In order to enable the Huffman trees to incorporate unknown elements during runtime (e.g., unexpected power readings), we have implemented an escape symbol analog to dynamic Huffman coding [19] in each tree. By transmitting this escape symbol, yet unknown successor symbols can be transmitted unencoded and integrated into the dynamic Huffman tree during runtime.

3.3. tiny Lempel Ziv Markov Chain Algorithm

The Lempel Ziv Markov Chain Algorithm (LZMA) relies on the combination of a dictionary-based compression mechanism with a range coding step. Each symbol in the output sequence is prefixed by a flag that determines whether it represents a reference to the dictionary or the output of the range coder. In contrast to LZMA, the tiny Lempel Ziv Markov Chain Algorithm (tLZMA) uses a constrained history window of 128 bytes for the dictionary coding step and entirely omits the range coding step. Furthermore, references to the dictionary are encoded by an optimal prefix code in order to optimize the algorithm's performance.

[1] `http://www.tracebase.org`

3.4. Lempel Ziv Markov Chain Huffman Coding

The Lempel Ziv-Markov Chain-Huffman (LZMH) method is a combination of the tLZMA and ATH methods. Symbols that cannot be resolved in tLZMA's dictionary are not transferred unencoded, but rather looked up in an adaptive Huffman tree. This Huffman tree is constructed analog to the ATH scheme, but it also includes counters for entries that are not present in the tree, such that the prefix bit could be omitted.

3.5. Reference Compression Algorithms

To compare the performance with existing desktop computer compression algorithms, we have regarded the ZLIB (1.2.6), BZIP2 (1.0.6), LZMA (9.20), LZW, and the LZSS algorithms [20]. We integrated them into our reference implementation by means of publicly available libraries in the C or C++ programming languages, or developed custom implementations of the LZW and LZSS algorithm, respectively. No specific adaptations to the resource constraints of the underlying hardware platforms were made during the implementation stage.

4. EVALUATION METHODOLOGY

We analyzed the presented compression algorithms using the datasets, methods, and metrics explained in the following.

4.1. Evaluation Data

In order to cater for a fair comparison and ensure the viability of the compared algorithms in realistic settings, evaluations need to make use of input data that has been collected by monitoring real households and appliances. For that purpose, we used two data sources for our evaluation.

To evaluate the performance of the compression algorithms in a real smart metering scenario, we obtained 3 500 ASCII-coded (EN-62056-21) datasets from a real smart meter installation. Dataset sizes range from 76 to 3 100 bytes and resemble different smart meter reading types. We refer to this dataset as *smart meter* dataset in the remainder.

The second source of data was a deployment of Plugwise Circles[2], each of which has been periodically polled to return the measured average power consumption. These traces help customers to identify individual contributors to the overall demand. The devices were sampled approximately once per second, leading to 86 400 readings per data set on average. Due to delays on the wireless link, however, some of the output files contained less values. The data has been converted to a binary format, in which all readings were concatenated. In total, we have gathered 95 datasets of different household appliances, to which we refer to as the *binary encoded daily device reports* in the following.

Algorithm	ROM (Byte)	RAM (Byte)	
		static	stack/heap
LZSS	544	129	19
LZW	550	12 416	16
ZLIB	27 960	2 690	ca. > 1 000
BZ2	28 332	1 564	ca. > 100 000
LZMA	34 442	110	ca. > 6 000 000
ATH	592	170	15
AMCH	1 680	1 820	21
tLZMA	992	133	27
LZMH	1 428	378	29

Table 1. Memory consumption of the compression algorithm implementations

4.2. Methodology and Metrics

We implemented ATH, AMCH, tLZMA, and LZMH in the C programming language and used existing implementations of the reference algorithms. Our implementations were tailored to the capabilities of low-power, low-resource devices. In particular, we did not use dynamic memory allocation, for it is rarely available on the class of expected target devices. Note that the reference algorithms make use of dynamic memory allocation.

For each data source, we compressed all datasets individually and measured the compression rate on a desktop PC. The compression rate is defined as the fraction of total size reduction and inpupt dataset size—i.e., a 10% compression rate is equivalent to decreasing the dataset size by 10% through compression. We use boxplots to depict the spreading of compression rates, where the box represents the upper and lower quartile and the line in the middle the median. The whiskers of the box indicate the minimum and maximum compression rates. We have also measured the real-world processing time for the smart meter data on an ATmega 1281 microprocessor with 8 kB RAM running at 8 MHz. This experiment could only be run for ATH, LZSS, tLZMA, and LZMH due to memory constraints (cf. Table 1). Each dataset was compressed multiple times until a per-dataset standard deviation below 1 ms (the resolution of our measurement clock) was achieved.

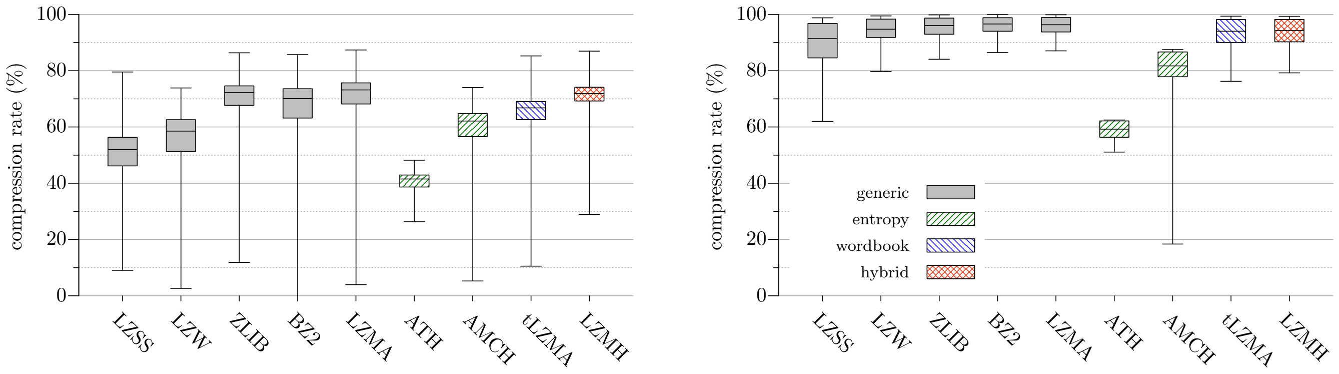
5. EVALUATION RESULTS

This section assesses the compression algorithms with particular respect to their applicability on embedded devices. We analyze their memory footprint, evaluate the achievable compression rates for real smart metering datasets and device reports, and discuss the compression time overhead in context of the achieved bandwidth savings.

5.1. Memory Consumption

The memory footprints of the compression algorithms for an ATmega 1281 are depicted in Table 1. The presented figures

[2] More details about the Plugwise system are available online at `http://www.plugwise.com`



(a) (b)

Fig. 1. Compression rates for (a) ASCII encoded smart meter datasets and (b) binary encoded daily device reports

only refer to memory demands for compression execution. Additional memory is required for storing smart meter data, and, of course, for application and network stack code.

The figures show that out-of-the-box solutions, with exception of LZSS, barely fit the capabilities of embedded devices, which in many cases offer only 64-128 kB ROM and 8-16 kB RAM. In contrast, the approaches presented in this paper comply with available resources, where particularly ATH and tLZMA are extremely lightweight.

5.2. Compression Rate

Our analysis shows that smart meter data is well compressible, particularly since all readings comprise a large fraction of digits, i.e., few symbols occur with high frequency and the data has low entropy. However, the variety of reading types and dataset sizes impacts compression performance. Figure 1a depicts the distribution of compression rates. Standard algorithms achieve average compression rates of 50-75% but exhibit large variation. This can be explained by the methods not being able to exploit the structure of datasets with sizes less than 100 bytes. One can observe this also with the more elaborate algorithms AMCH, tLZMA, and LZMA. The latter is close to the high compression rates of the heavy-weight generic compression algorithms. Note that a median compression rate of more than 70% drastically reduces bandwidth utilization by approximately the same percentage. The very simple ATH algorithm yields the lowest while least deviating compression rates. While ATH is outperforming its competitors in case of very small datasets, it does not profit from increased dataset sizes. This is evident from Fig. 2, where the index of the analyzed smart meter data set is indicated on the x-axis (the datasets were sorted by size). The figure shows that using references and wordbooks pays off as soon as smart meter readings exceed a size of 100-200 bytes. Moreover, the one-bit prefix of ATH codes diminishes its compression rates for data sets with few, frequent symbols. However, ATH is capable of cutting all data down to almost half of their size. Compared to its function logic and code size, which has been specifically tailored to embedded devices, this represents a remarkable result.

For large datasets with only small deltas between individual readings, compression performance is further increased. The results for the binary encoded daily device reports are shown in Fig. 1b. Again, ATH produces comparably low compression rates with a maximum of 63%, which is as low as the worst rate produced by its opponents excluding AMCH. The latter achieves fair results, yet suffers from significant outliers with less than 20% compression rate. All other algorithms offer a comparable performance, where LZSS exhibits the largest deviation, because it does not encode frequent references with less bits, as, e.g., LZMH.

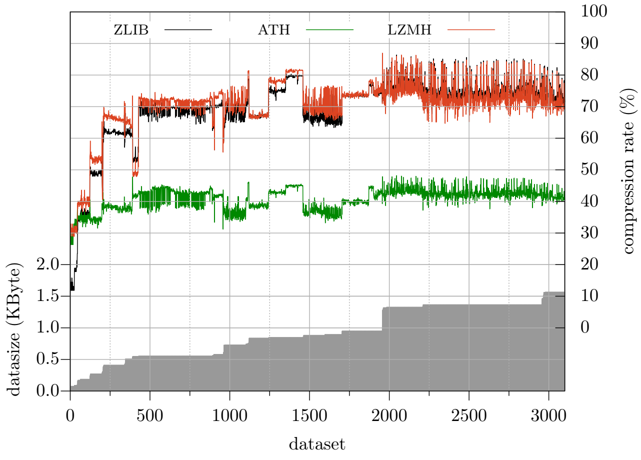


Fig. 2. Compression ratio vs. smart meter dataset size

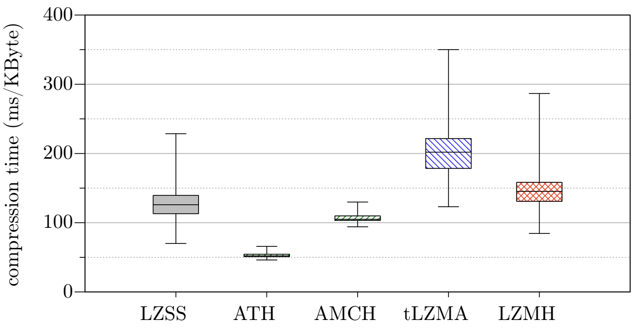


Fig. 3. Processing times for smart meter datasets

5.3. Processing Time

Compression time is a significant criterion for the selection of a compression algorithm. On the one hand, an active microprocessor consumes more energy, which is undesired if it is battery-powered. One the other hand, long compression runtime may prevent the microprocessor from concurrently performing other tasks—such as packet forwarding. Large datasets should hence be compressible within few milliseconds. This can be circumvented by preemptive tasking, which yet increases implementation effort and the chance of side-effects and runtime errors due to race conditions. Another option would be a stateful implementation approach, in which control over the microprocessor is returned to the scheduler periodically during compression. However, this requires more implementation effort and may lead to a larger memory footprint, because state variables must be persistent.

The distribution of per-byte processing times of the smart meter datasets is shown in Fig. 3. ATH has a median compression time of 52 ms per kilobyte of input data, so that compressing medium size datasets is unlikely to infer with other tasks. Moreover, energy expenditure for compression is easily compensated by the savings due to shorter radio usage. With a median of 193 ms per kilobyte of input data, tLZMA is the slowest of the tested algorithms.

Since there is a notable deviation of processing speed, we took a closer look at the dependency of processing time and dataset size. Figure 4 reveals that compression speed of ATH is not affected by dataset size. The remaining algorithms, in contrast, compress certain datasets considerably quicker or slower than others. These are the same datasets that allow for particularly high or low compression rates (cf. Fig. 2), where a high compression rate implies quicker processing time. This is caused by less complex algorithm execution due to quicker wordbook and code-word lookups.

5.4. Algorithm Selection Guidelines

Having analyzed both computational and resource demand of the compression algorithms, visible discrepancies between the analyzed algorithms could be determined. While existing compression libraries optimized for unconstrained operation achieve high compression gains, their memory consumption clearly disqualifies them for application in smart meters. Moreover, they achieve poor compression rates for small datasets. Two of the four proposed compression mechanisms, namely tLZMA and LZMH achieve compression gains comparable to these libraries, but come at significantly lower memory demand. The LZMH method offers the best compression rate while still fulfilling the constrained hardware resource requirements of embedded devices. The tLZMA algorithm provides similar compression rates, but consumes more time to compress data. It should thus only be used if LZMH consumptions are too heavy on memory and the system is not operated on a battery-powered supply. When both fast execution and a very small memory resource footprint are desired, LZSS gives fair compression results while being faster than LZMH. In case the compression time is the most critical factor, ATH is the best choice to compress smart meter data due to its significantly faster compression speed.

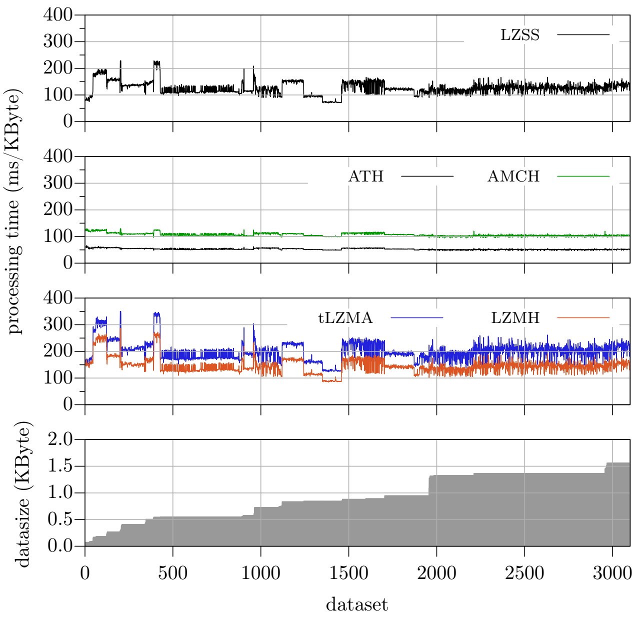


Fig. 4. Processing time vs. smart meter dataset size

6. CONCLUSION

Smart metering brings advantages for both utilities and customers by supplying fine-grained meter readings. To reduce bandwidth, energy consumption, and transmission cost while complying with legal requirements, lossless data compression is required. We have presented four compression algorithms for smart meter data that easily fit the tight resource constraints of cheap, low-power embedded meter devices.

We have evaluated their compression gains and processing time in comparison to five standard compression libraries for desktop computers. Our findings prove that a tradeoff between compression gains, processing time, and resource demands must be found. The best performing algorithm in compressing both DIN EN-62056-21 meter data as well as individual device traces from a Plugwise device was the LZMH algorithm. Its resource requirements make it an ideal extension to any smart metering device, resulting in average compression rates between 75% and 95% at comparably modest execution times. If extremely low execution time is mandatory, the ATH algorithm is the optimal choice, still achieving 40-60% compression rate at a fourfold processing speed-up.

7. REFERENCES

[1] Frost & Sullivan, "Smart Meter Market – Frost & Sullivan Forecasts 109% Growth in the UK," Online: http://www.frost.com/prod/servlet/press-release.pag?docid=238393168, 2011.

[2] J. Vasconcelos, "Survey of Regulatory and Technological Developments Concerning Smart Metering in the European Union Electricity Market," EUI RSCAS PP 2008/01, Florence School of Regulation, 2008.

[3] U. Greveler, B. Justus, and D. Loehr, "Multimedia Content Identification Through Smart Meter Power Usage Profiles," in *Proceedings of the 5th International Conference on Computers, Privacy, and Data Protection (CPDP)*, 2012.

[4] N. Zouba, F. Brémond, and M. Thonnat, "Multisensor Fusion for Monitoring Elderly Activities at Home," in *Proceedings of the 6th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2009.

[5] R. C. Kryter and H. D. Haynes, "Condition Monitoring of Machinery using Motor Current Signature Analysis," in *Proceedings of the 7th Power Plant Dynamics, Control and Testing Symposium*, 2003.

[6] G. W. Hart, "Residential Energy Monitoring and Computerized Surveillance via Utility Power Flows," *IEEE Technology and Society Magazine*, vol. 8, no. 2, 1989.

[7] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, 1977.

[8] T. A. Welch, "A Technique for High-Performance Data Compression," *IEEE Computer*, vol. 17, no. 6, pp. 8–19, 1984.

[9] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[10] K. Barr and K. Asanović, "Energy Aware Lossless Data Compression," in *Proceedings of the 1st International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2003, pp. 231–244.

[11] 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, pp. 359–367.

[12] 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)*, 2006, pp. 265–278.

[13] 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)*, 2004, pp. 516–524.

[14] 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)*, 2006, pp. 211–216.

[15] A. Reinhardt, "Designing Sensor Networks for Smart Spaces – Unified Interfacing and Energy-Efficient Communication between Wireless Sensor and Actuator Nodes," Ph.D. dissertation, Technische Universität Darmstadt, Multimedia Communications Lab, 2011.

[16] J. Z. Kolter and M. J. Johnson, "REDD: A Public Data Set for Energy Disaggregation Research," in *Proceedings of the SustKDD Workshop on Data Mining Applications in Sustainability*, 2011.

[17] 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 '10)*, no. LNCS 5970, Coimbra, Portugal, Feb. 2010.

[18] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[19] J. S. Vitter, "Design and Analysis of Dynamic Huffman Codes," *Journal of the Association for Computing Machinery*, vol. 34, no. 4, pp. 825–845, 1987.

[20] D. Salomon and G. Motta, *Handbook of Data Compression*, 5th ed. Springer, 2010.