Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 672921, 11 pages
http://dx.doi.org/10.1155/2014/672921

Research Article

Lightweight Data Compression in Wireless Sensor Networks Using Huffman Coding

Henry Ponti Medeiros,[1] Marcos Costa Maciel,[2] Richard Demo Souza,[3] and Marcelo Eduardo Pellenz[4]

[1] *Purdue University, 465 Northwestern Avenue, West Lafayette, IN 47907-2035, USA*
[2] *Federal Institute of Education, Science and Technology of Amazonas (IFAM), Campus Manaus Industrial District, Avenida Danilo Areosa, 1672, 69075-351 Manaus, AM, Brazil*
[3] *Federal University of Technology-Paraná (UTFPR), Avenida Sete de Setembro, 3165, 80230-901 Curitiba, PR, Brazil*
[4] *Pontifical Catholic University-Paraná (PUC-PR), R. Imaculada Conceição, 1155, 80215-901 Curitiba, PR, Brazil*

Correspondence should be addressed to Henry Ponti Medeiros; henryponti@gmail.com

Received 5 August 2013; Revised 22 November 2013; Accepted 10 December 2013; Published 23 January 2014

Academic Editor: Wan-Young Chung

This paper presents a lightweight data compression method for wireless sensor networks monitoring environmental parameters with low resolution sensors. Instead of attempting to devise novel ad hoc algorithms, we show that, given general knowledge of the parameters that must be monitored, it is possible to efficiently employ conventional Huffman coding to represent the same parameter when measured at different locations and time periods. When the data collected by the sensor nodes consists of integer measurements, the Huffman dictionary computed using statistics inferred from public datasets often approaches the entropy of the data. Results using temperature and relative humidity measurements show that even when the proposed method does not approach the theoretical limit, it outperforms popular compression mechanisms designed specifically for wireless sensor networks.

1. Introduction

One of the greatest challenges to the construction of large scale wireless sensor networks (WSNs) with practical applicability is the development of mechanisms that allow the network to operate for prolonged periods of time relying solely on the limited amounts of energy that can be stored in or harvested by wireless sensor nodes. Since data communication is generally the main factor responsible for draining the energy reserves of the network, techniques to reduce the amount of information transmitted by the sensor nodes are of great interest. One effective approach to reduce data communication in the network is to compress the information locally before it is transmitted.

Although data compression is a well-established research area, despite the extraordinary advances in the computational capability of embedded devices, most existing algorithms still cannot be directly ported to wireless sensor nodes because of the limited hardware resources available, particularly program and data memory [1]. Even though many of the time-honored compression algorithms could be executed in modern wireless sensor nodes, they would leave few resources available for the nodes to carry out other tasks such as sensing and communication. More importantly, these nodes would have significantly fewer opportunities to enter deep sleep modes and attain the energy efficiency that motivated the use of a compression algorithm in the first place. Therefore, a number of data compression methods specifically designed for WSNs have been proposed in the past few years [2–11]. What many of these methods have in common is the fact that they make use of the correlation of the data acquired by the sensor nodes in order to achieve high compression ratios while employing computationally inexpensive algorithms.

However, WSNs are generally deployed with the purpose of monitoring a particular phenomenon of interest [12]. Therefore, we show that, if the statistics of this phenomenon are known beforehand from general datasets, and if the data collected by the sensor nodes presents relatively low resolution, by employing simple Huffman encoding, it is possible to achieve compression ratios higher than those obtained by state-of-the-art algorithms such as those presented in [2–4]. More specifically, we show that by constructing a fixed Huffman dictionary to encode the differences between two consecutive samples from a large general dataset, the compression ratio obtained on test datasets of the same phenomenon at different locations and periods is very close to what would be achieved if a specific dictionary was constructed for each test dataset.

The rest of this paper is organized as follows. In Section 2, we briefly explain some of the recent contributions on data compression for WSNs. We then describe the proposed lightweight compression method in Section 3. In Section 4, we show the results obtained with our approach and compare them with those obtained when employing the methods in [2–4]. Finally, Section 5 concludes the paper.

2. Related Work

In the literature on compression methods for WSNs both lossy and lossless approaches that exploit the high temporal correlation of the sensor node data can be found. One of the first lossy methods for data compression in WSN, lightweight temporal compression (LTC) [5], approximates the data collected by each sensor node in a WSN by a set of lines. In [6] a variation of the run length encoding (RLE) method for data compression in WSN known as K-RLE approximates a string of N measurements with values in the range $[K-d, K+d]$ as the pair (N, d), where K defines the precision of the method.

Although lossy compression methods can generally achieve high compression ratios at the expense of moderate accuracy losses, in many WSN applications it may not be clear before data collection how much information can be disregarded without compromising the overall purpose of the system. Event-based communication approaches attempt to resolve this problem by limiting the transmission of sensor data to responses to user queries [13]. However, in many cases, the user may not be able to formulate queries without observing the raw sensor data beforehand. As a consequence, a number of lossless compression methods for WSNs have been proposed. S-LZW [7] is an adaptation of the celebrated Lempel-Ziv-Welch (LZW) algorithm [14] for resource-constrained wireless sensor nodes. Alternatively, in [8] sensor measurements are coded using adaptive Huffman [15], but in order to save memory the number of symbols present in the Huffman tree is limited to the measurements that happen most frequently. In an interesting attempt to facilitate the application of data compression algorithms in real WSN deployments, a middleware layer is proposed in [9] in which only the dissimilarity between the packet to be transmitted and a previously transmitted reference (or index) packet is compressed using variable length coding.

While a number of additional works on data compression for WSNs such as [10, 11] attempt to employ distributed source coding techniques [19] to exploit the spatial correlation in the data acquired by the sensor nodes, we are particularly interested in methods that do not make any assumptions about the spatial structure of the WSN. Furthermore, in the context of data aggregation, that is, when nodes along routing paths collaborate to reduce the dimensionality of the data collected by multiple nodes, one recent and extremely promising technique is compressed sensing [20]. In this work, however, we consider approaches that attempt to achieve efficient lossless data compression by leveraging solely on the temporal correlation of the data collected by each sensor node and performing all the computations locally, without relying on information from other nodes. Two of the most recent and effective approaches in this category are Marcelloni and Vecchio's lossless entropy compression (LEC) [2, 3] and Kiely et al.'s adaptive linear filtering compression (ALFC) [4].

LEC computes the differences of consecutive sensor measurements and divides them into groups whose sizes increase exponentially. Each group corresponds to the number of bits required to represent the measurement differences. These groups are then entropy coded using a fixed compression table based on the baseline JPEG algorithm to compress the DC coefficients of an image. The compressed symbols are formed by concatenating the group number and the index of the element within the group. The authors reported high compression ratios for actual environmental data collected by WSNs.

In ALFC, an adaptive linear filter is used to predict the future M samples of the dataset and the prediction errors are compressed using an entropy encoder. In order to account for the limited computational capabilities of wireless sensor nodes, the method employs a quantization mechanism. Adaptive prediction avoids the requirement of defining the filtering coefficients *a priori* while still allowing the system to adjust to dynamic changes in the source. The authors showed that ALFC achieves higher compression ratios than previous methods while requiring significantly fewer hardware resources.

Although essentially all of the methods described above rely on the temporal correlation of the data collected by WSNs to achieve high compression ratios, they do not take into consideration the fact that the statistics of the phenomena to be monitored by a particular WSN are usually relatively easy to estimate before the deployment of the sensors. Furthermore, the state-of-the-art algorithms perform well when the resolution of the data collected by the sensor nodes is very high, but when the data resolution is limited to integer measurements they suffer significant performance penalties. In this work, we leverage on these facts to achieve even higher compression ratios while resorting only to traditional entropy-based compression methods with extremely modest computational requirements.

3. Lightweight Compression of Environmental Data

In this section, we define the problem of data compression in WSNs and present a simple compression approach, which takes into consideration the characteristics of the measurements acquired by the sensor nodes so that algorithmic complexity can be reduced without sacrificing compression ratios.

Table 1: Main characteristics of the temperature datasets, including location, temperature range, number of samples, date range when the measurements were taken, and sampling interval.

	Location (lat., lon.)	Range (°C)	Samples	Date (mm/dd/yy)	Sampling interval
Set 1	Hagerstown, MD, USA (39.711, −77.722) [16]	−16 to +37	26,843	01/01/09 to 07/08/11	10 min
Set 2	Manaus, AM, Brazil (−3.145, −59.986) [16]	+21 to +36	3,676	07/01/11 to 11/30/11	60 min
Set 3	Jonesboro, AR, USA (35.834, −90.649) [16]	−1 to +41	3,738	07/01/11 to 11/20/11	60 min
Set 4	Le Génépi, Switzerland (46.025, 7.044) [17]	−11 to +16	42,141	08/28/07 to 10/31/07	2 min
Set 5	Morges, Switzerland (46.494, 6.472) [17]	+5 to +29	14,527	08/06/07 to 09/02/07	2 min
Set 6	Bern, Switzerland (46.948, 7.444) [17]	−14 to +6	4,851	03/13/07 to 03/15/07	0.5 min
Set 7	Pas du Chat, Switzerland (46.029, 7.408) [17]	−10 to +8	3,041	04/16/08 to 04/20/08	2 min
Set 8	Matterhorn, Switzerland/Italy (45.976, 7.658) [18]	−18 to +25	243,665	01/01/12 to 12/31/12	~2 min
Set 9	Chamonix, France (45.879, 6.887) [18]	−5 to +31	61,746	10/01/12 to 12/29/12	~2 min

3.1. Problem Definition. We consider a sensor node monitoring environmental data. Let the data acquired by the sensor at time instant t be represented, after analog to digital conversion, by $x(t) \in \mathcal{X}$, where $\mathcal{X} \subset \mathbb{Z}$. The set $\mathcal{X} = \{x_0, x_1, \ldots, x_{N-1}\}$ is said to be the source alphabet. A particular source encoder represents each symbol $x_i \in \mathcal{X}$ with an l_i bits long codeword, so that the average number of bits used to represent each source symbol is given by $L = \sum_{i=0}^{N-1} p_i l_i$, where p_i is the probability that $x(t) = x_i$. When a source encoder is absent, it is typical to represent all source symbols with codewords of equal length, so that the symbol length is $L_u = \lceil \log_2 N \rceil$ bits/symbol. Moreover, the theoretical limit for the minimum number of bits/symbol for a discrete source is the source entropy [21]:

$$H(\mathcal{X}) = \sum_{i=0}^{N-1} p_i I_i = -\sum_{i=0}^{N-1} p_i \log_2(p_i), \tag{1}$$

where $I_i = -\log_2(p_i)$ is the information measure of source symbol x_i.

The efficiency of a compression algorithm can be measured by comparing the average symbol length after compression to the source entropy. For instance, consider the case of a set of integer temperature measurements denoted as Set 1 in Table 1. As the measured integer temperature values range between −16°C and +37°C, without compression we have to use $L_u = 6$ bits/symbol in order to represent the 54 different source symbols in that alphabet. The source entropy in this case is $H = 5.29$ bits/symbol, which can be approached by simple Huffman coding. Indeed, after designing the Huffman code for this particular source, only $L = 5.31$ bits/symbol are required after compression. However, because the probability distribution of the temperature values is somewhat uniform, the reduction in the average symbol length is only 0.69 bits or 11.5%.

One can do much better by considering the differences of consecutive temperature measurements, so that the data to be transmitted is $d_i = x_i - x_{i-1}$. For instance, in the case of Set 1 the entropy of the difference of consecutive temperature measurements is only $H_d = 2.13$ bits/symbol, which is a reduction of 59.7% with respect to the entropy of the temperatures in Set 1. Such a reduction is due to the strong correlation between consecutive temperature samples, making the probability distribution of the differences strongly nonuniform. Thus, it is much more promising to consider the compression of the differences of consecutive temperatures than the compression of the temperatures themselves. Such a fact has been exploited in [2], where the authors compress the differences by using a scheme similar to the JPEG compression of the DC coefficients of a digital image.

By applying the method in [2] to the temperatures in Set 1 we obtain $L = 3.24$ bits/symbol, a remarkable reduction of 46.0% with respect to the original 6 bits/symbol required by Set 1, but still 18.5% more than the theoretical limit given by the entropy of the difference of the temperatures ($H_d = 2.13$ bits/symbol). The theoretical limit can be approached by Huffman coding, yielding $L = 2.16$ bits/symbol, a reduction of 64.0% with respect to the original 6 bits/symbol. Nevertheless, note that the scheme proposed in [2] uses a fixed dictionary, which can be applied to any source. An entropy coding technique such as Huffman coding has to be matched to the source distribution in order to achieve optimal performance [21]. Hence, there is a causality issue, as the exact distribution may not be known *a priori*. Moreover, the Huffman alphabet designed for a given source may perform poorly if used to compress another source with a different distribution. In order to circumvent these problems one can make use of an adaptive (or dynamic) Huffman coding technique [15]. Although in this kind of scenario in which there is little variation of symbol probability, adaptive Huffman would tend to perform well; the main drawback of this approach is that it requires maintaining one dictionary for each connection between a pair of neighboring nodes [8]. Given the severe memory constraints of wireless sensor nodes, this method is utterly impractical.

3.2. Proposed Scheme. Our objective is to devise a simple compression method which approaches the performance of optimal entropy coding while relying on a fixed dictionary. After comparing the probability distribution of the temperatures and of the differences of consecutive temperatures for many datasets of measurements carried out at different locations, we noticed that the distributions of the differences are quite similar for all datasets, even though the distributions of the temperature values vary significantly. This can be observed in Figure 1, which shows the probability distribution of the differences between consecutive measurements for each of the datasets in Table 1. As the figure shows,

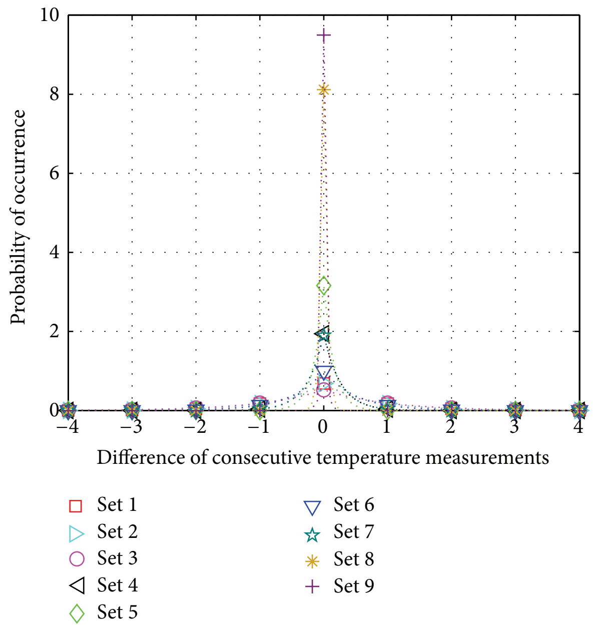

Figure 1: Probability distributions of the differences between consecutive measurements for each of the temperature datasets.

all the distributions are approximately Laplacian with zero mean [22]. Since the temperature differences only assume integer values, the distributions are actually discrete and the continuous approximations are shown simply to facilitate the visualization. More importantly, note that for all datasets, if we list the differences from the most likely to the least likely, the result is $(0, \pm1, \pm2, \pm3, \pm4, \dots)$. Hence, in principle, it should be possible to use a fixed Huffman alphabet to compress different measurement sets if we consider the difference of the temperatures, as all sets have very similar behavior and the optimal Huffman alphabet for each set tends to be similar.

Thus, in this paper we propose to construct a fixed alphabet obtained by the application of the Huffman algorithm to a large dataset of temperature measurements. We consider Set 1 as our reference dataset, without any particular reason other than the fact that both the number of samples and the measured temperature range are quite large. Unlike LEC, which always uses the same alphabet, our approach uses a reference dataset to generate a dictionary for a particular parameter under observation (e.g., temperature). We compute the frequencies of each of the symbols available in the reference dataset and use them to construct the Huffman tree that represents the compression alphabet [23]. This alphabet, shown in Table 2, is then used to compress different temperature datasets. As the alphabet is fixed, the complexity of the proposed approach is rather low, being no more complex than that in [2]. For instance, an implementation of the Huffman encoding and decoding for AVR microcontrollers, widely used in sensor nodes, utilizes only 468 bytes of program memory [24]. In this work, we utilize Huffman coding due to its utmost simplicity; however, other entropy coding approaches such as arithmetic coding would likely produce similar results [25].

Table 2: Proposed fixed Huffman dictionary for temperature datasets.

d_i	Codeword
−10	0010101000101110
−9	00101010001010
−8	001010100010110
−7	001010100011
−6	001010100111
−5	00101010010
−4	001010101
−3	0010100
−2	00100
−1	01
0	1
+1	000
+2	0011
+3	001011
+4	00101011
+5	00101010000
+6	001010100110
+7	00101010001001
+8	00101010001000
δ	0010101000101111

In the proposed compression scheme, as in any dictionary-based differential compression approach, two special cases must be considered: (i) in the beginning of data collection, the first sample, x_0, must be transmitted uncompressed since there is no previous measurement to compute the difference d_0. From the second sample on, the differences $d_i = x_i - x_{i-1}$ can be properly computed and compressed; (ii) in addition, the compression table, for example, Table 2, covers a limited range of difference values, according to the data available in the reference dataset. However, the probability of occurrence of a symbol not present in the dictionary is extremely low (see Figure 1). Hence, its value can be sent uncompressed and identified by the presence of a special marker in the Huffman dictionary. That is, a codeword which is not part of the original dictionary and whose presence can be unambiguously detected may be transmitted to signify that the next symbol corresponds to an uncompressed value, which can then be transmitted using a subsequent codeword of a fixed previously defined length. In Table 2 the special marker is the 16 bits symbol δ.

4. Results

In this section we investigate the performance of the proposed scheme when the fixed Huffman alphabet in Table 2 is used to compress different datasets. First we consider the temperature datasets in Table 1. It is important to note that the test datasets, Set 2 to Set 9, were collected at different locations and times than those of the reference dataset Set 1, which was used to construct the alphabet in Table 2. The performance of the

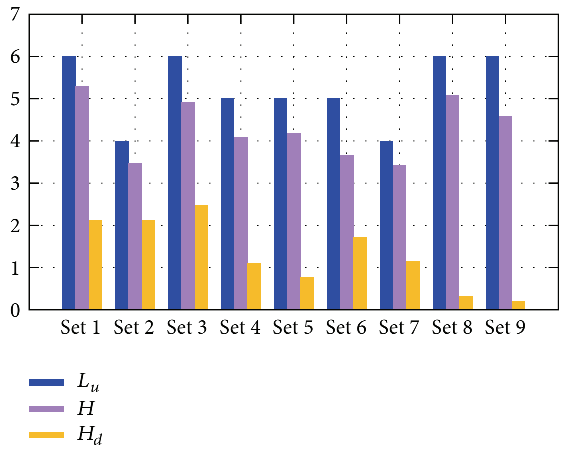

FIGURE 2: Average symbol length (L_u) for the uncompressed case, entropy of temperature measurements (H), and entropy of the differences of consecutive measurements (H_d).

proposed scheme is compared to the theoretical limit given by the source entropy (considering both the temperatures and the differences of the temperatures) and to the performance of the LEC [2] and ALFC [4] algorithms. In order to meet the assumptions of the proposed method, whenever a dataset contains measurements with resolutions higher than 1°C, the data is rounded before being processed by any of the algorithms under consideration. To further validate our approach, we also carry out experiments using six datasets of relative humidity measurements. Again, a resolution of 1% is assumed, and the data is rounded whenever that is not the case (although it is possible to easily adapt the proposed method for higher resolution data, preliminary experiments showed that as the resolution increases, more subtle phenomena such as sensor noise cause the distributions at different sites to differ to a larger extent, penalizing the performance of the approach).

4.1. Comparison with Lossless Entropy Compression. Figure 2 shows the average uncompressed symbol length (L_u), the entropy of the temperature measurements (H), and the entropy of the differences between consecutive temperature measurements (H_d) for each of the datasets shown in Table 1. Figure 3 shows the average symbol length (L) after compression, the compression ratio (C_r), and the code efficiency (η) when using the proposed scheme as well as LEC. The compression ratio is computed as

$$C_r = 100 \times \left(1 - \frac{L}{L_u}\right)\%, \qquad (2)$$

while the code efficiency with respect to the theoretical limit is given by

$$\eta = 100 \times \left(\frac{H_d}{L}\right)\%. \qquad (3)$$

As the results demonstrate, the proposed fixed dictionary method outperforms LEC, always achieving a larger compression ratio. Moreover, the code efficiency η for the proposed scheme approaches 100% for some of the test datasets (Sets 2, 3, and 6). Note also that the code efficiency for the proposed scheme is considerably larger than that obtained with LEC (in [3], the authors also compare the performance of LEC with that of a semiadaptive Huffman system in which the dictionary is generated based on an initial set of samples and then used to compress the entire dataset. Because that approach fails to account for longer-term variations in the measurements collected by the WSN, the improvement over LEC is at most 4.6% for temperature datasets and 6.2% for relative humidity datasets, whereas the proposed approach presents much higher compression ratio gains with respect to LEC). Even in the case of the datasets for which the performance of the proposed scheme does not approach the theoretical limit (notably in datasets of extremely low entropy, such as Sets 8 and 9), the reduction in symbol length is almost 50% with respect to LEC and the corresponding compression rates are above 80%. In fact, because those datasets tend to contain long sequences of measurements that convey very little information, none of the methods under consideration can achieve high efficiency for sets of extremely low entropy (ALFC achieves approximately 10% lower efficiency than the proposed approach for Sets 8 and 9). In the case of Set 9, for example, the average length of a sequence consisting exclusively of zeros is 38 measurements. In order to achieve higher efficiency in datasets of very low entropy, a method that encodes long sequences of measurement differences as a single symbol would be needed. Although the proposed method could be adapted for that purpose, doing so might jeopardize its versatility.

4.2. Comparison with Adaptive Linear Filtering Compression. ALFC assumes that the measurements are transmitted in fixed-length packets which contain enough information so that the measurements within the packet can be decoded regardless of communication failures that may have caused previous packet losses. In order to achieve comparable results, we modify our approach so that measurements are also assumed to be contained in fixed-length packets, and the first measurement in each packet is not compressed. That allows every measurement within a packet to be decoded independently of the information contained in previous packets and adds robustness to packet losses similar to that obtained by ALFC.

We compare the performance of our approach to that of ALFC using the same set of parameters employed in the experimental evaluation presented in [4]. That is, we used the quantization parameters $A = 15$, $B = 8$, and $R = 14$ and the order of the filter was defined to be $M = 3$. As for the number of bits required to represent the measurements without compression, we used $b = 6$, since this is the minimum number of bits required to represent the measurements in our datasets. As suggested in [4], for each packet we computed the optimal value of the variable length coding parameter k, which has a major impact on the length of the corresponding coding symbols (ALFC uses mth Golomb codes [26], with the restriction that $m = 2^k$).

Figure 4 shows the average symbol length (L) after compression using ALFC and our proposed approach. Despite the current popularity of the 802.15.4 protocol, we did not

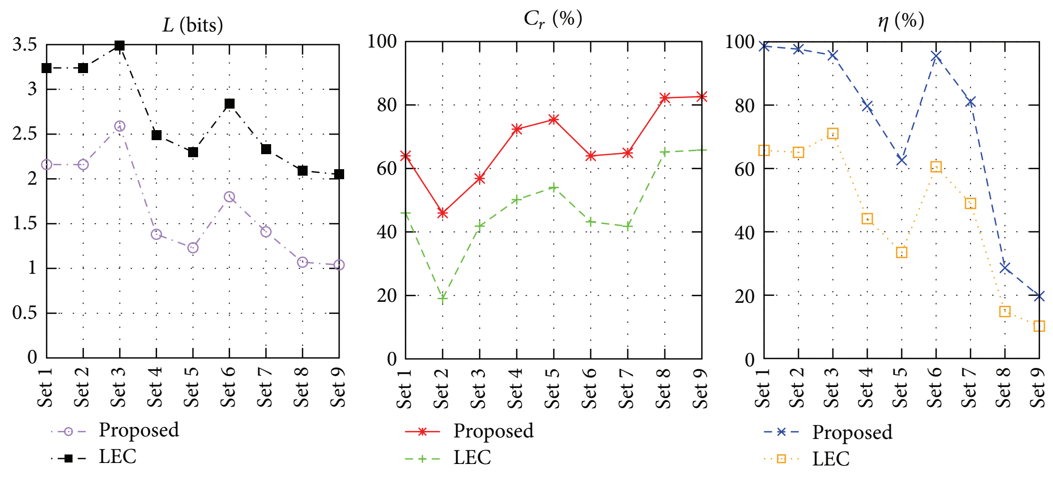

FIGURE 3: Average symbol length after compression (L), compression ratio (C_r), and code efficiency (η) for different temperature datasets.

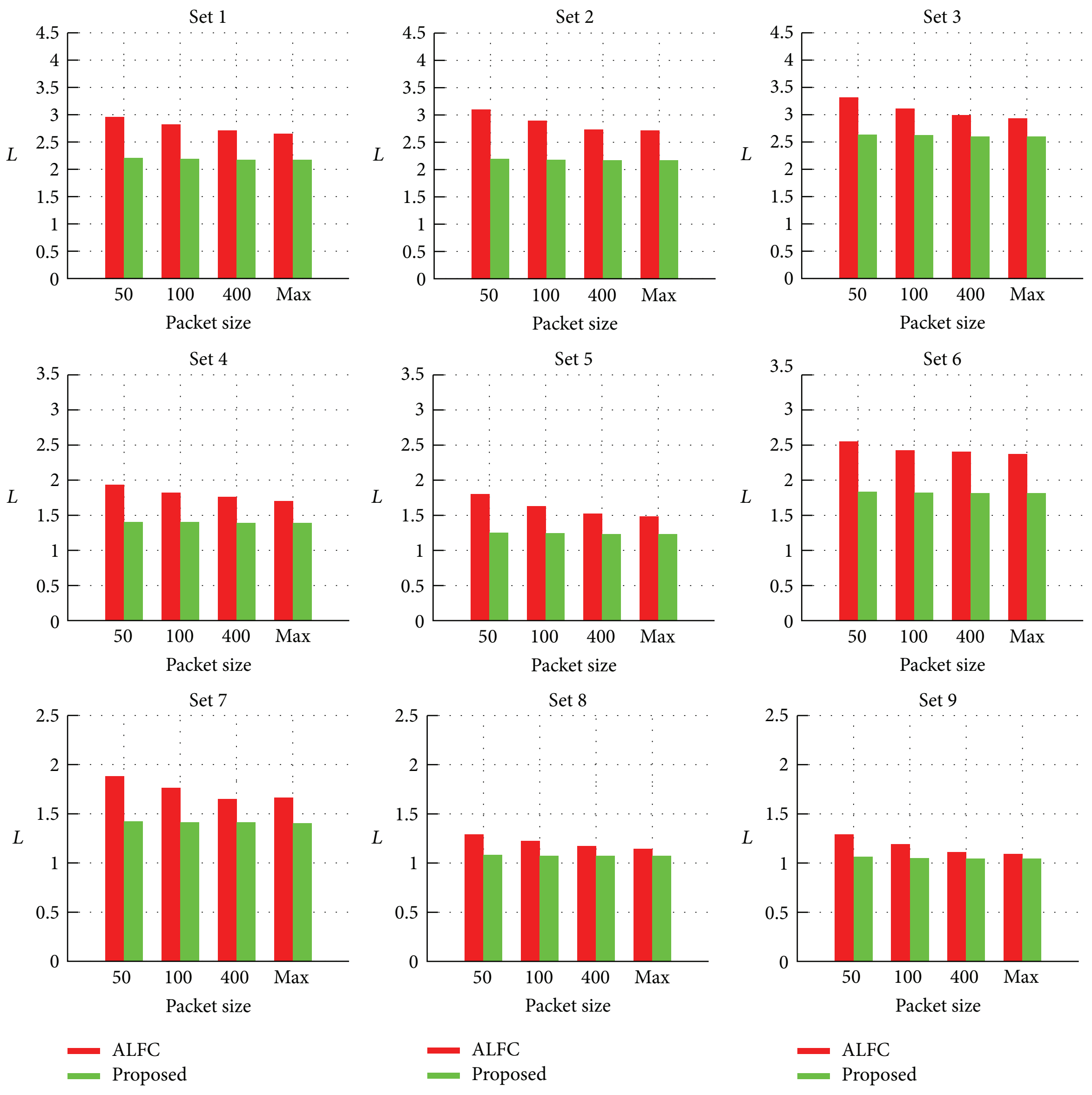

FIGURE 4: Average symbol length (L) (bits/sample) after compression using ALFC [4] and the proposed method for different packet sizes.

Table 3: Average symbol length (L) when measurements are compressed using dictionaries generated by other sets. Each column corresponds to the values obtained when the set on the top row is used to generate the dictionary.

	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9	σ
Set 1	2.16	2.17	2.31	2.21	2.27	2.20	2.21	2.36	2.22	0.07
Set 2	2.16	2.15	2.31	2.21	2.23	2.19	2.18	2.29	2.24	0.06
Set 3	2.59	2.59	2.56	2.70	2.76	2.65	2.66	2.93	2.68	0.11
Set 4	1.38	1.38	2.02	1.38	1.38	1.38	1.39	1.39	1.39	0.21
Set 5	1.23	1.23	2.00	1.22	1.22	1.22	1.23	1.23	1.23	0.26
Set 6	1.80	1.80	2.19	1.81	1.83	1.80	1.81	1.88	1.83	0.13
Set 7	1.40	1.40	2.04	1.40	1.40	1.40	1.40	1.41	1.42	0.21
Set 8	1.07	1.07	2.00	1.07	1.07	1.07	1.07	1.07	1.07	0.31
Set 9	1.04	1.04	2.00	1.04	1.04	1.04	1.04	1.04	1.04	0.32
Avg.	1.65	1.65	2.16	1.67	1.69	1.66	1.67	1.73	1.68	—

want to constrain our analysis exclusively to protocols that only allow very small payloads; hence we show the results for packet sizes of 50, 100, and 400 bytes. For comparative purposes, we also show the average symbol length when the entire dataset is compressed simultaneously (*max*). For each packet size, the average symbol length was computed over all packets. The results show that the average symbol lengths L obtained using the proposed approach are lower than those obtained using ALFC for any dataset or packet length. When the entire datasets are compressed simultaneously, we achieve an average symbol length reduction with respect to ALFC between 4.8% and 30.9%, whereas for packets of 50 bytes, the average reduction is between 16.3% and 30.6%. In addition, unlike ALFC, the proposed method does not require the user to adjust several different operation parameters, allowing broader practical applicability, besides being less complex.

4.3. Impact of Encoding Symbols Not Present in the Dictionary. To evaluate the impact of transmitting uncompressed symbols not present in the dictionary, we augmented each of the measurement datasets in Table 1 by inserting pairs of uncompressed symbols along with the corresponding special marker and computed the corresponding compression ratio C_r. Figure 5 shows the impact of varying the percentage of uncompressed symbols between 0.1% and 3.5% of the total size of each dataset. In the evaluation, we used the special 16 bit symbol δ shown in Table 2, and each uncompressed symbol was represented using L_u bits, according to Figure 2.

As Figure 5 shows, even for a relatively high percentage of uncompressed symbols, the compression ratio difference is below 15% in the worst case (Set 2) and close to 12% on average. It is important to note that if a high percentage of the measurements in a given dataset must be encoded by symbols which are not present in the original dictionary, then the dataset clearly violates the basic assumption that new measurements can be modeled based on the statistics of previous measurements. In that case, an alternative method should be employed. In most WSN applications, however, that situation is rather uncommon. In the nine temperature measurement datasets presented in Table 1, for example, there were no symbols that were not represented in the original dictionary shown in Table 2. As shown in the next section,

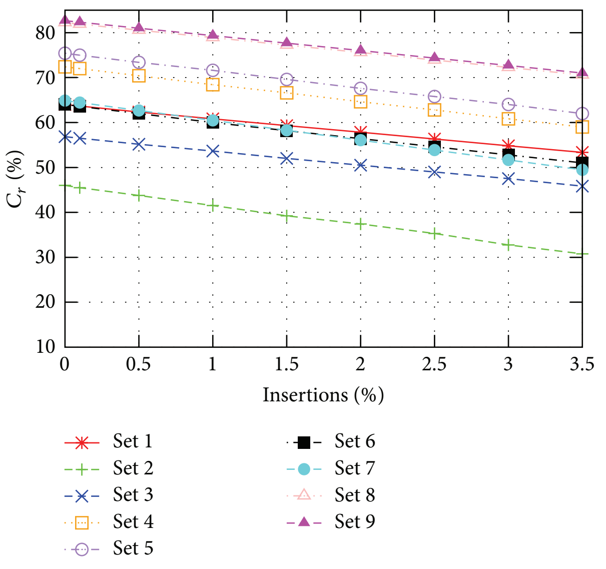

Figure 5: Impact of the insertion of symbols not present in the dictionary in the compression ratio C_r (%).

even when datasets which contain fewer symbols are used to build the dictionary, the percentage of uncompressed symbols remains quite low.

4.1. Comparison with Lossless Entropy Compression. With the purpose of demonstrating the generality of the proposed approach, we evaluated the performance of our method when datasets other than Set 1 were used to generate the dictionary. Table 3 shows the results of this evaluation. In the table, each row shows the average symbol length (L) obtained when the dataset on the left is compressed using the dictionary generated by the dataset on the top. The last row shows the average symbol length over all datasets. We can see that the average symbol length over all sets is reasonably consistent, except for that obtained using the dictionary generated by Set 3, which is higher than the average. The last column shows the standard deviation (σ) of the symbol length. We can see that the variation is generally quite small.

Table 4: Percentage of symbols not present in the dictionaries generated by the different datasets.

	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9
Set 1	0.00	0.03	0.03	0.79	1.77	0.03	0.23	2.05	0.06
Set 2	0.00	0.00	0.03	0.98	1.17	0.03	0.46	1.28	0.24
Set 3	0.00	0.03	0.00	2.84	3.18	0.03	1.02	4.39	0.19
Set 4	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.00
Set 5	0.00	0.00	0.00	0.04	0.00	0.00	0.02	0.01	0.01
Set 6	0.00	0.00	0.00	0.80	1.05	0.00	0.25	1.36	0.08
Set 7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.00
Set 8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Set 9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 5: Main characteristics of the relative humidity datasets.

	Location (lat., lon.)	Range (%)	Samples	Date (mm/dd/yy)	Sampling interval
Set 10	Morges, Switzerland (46.494, 6.472) [17]	44 to 95	15,362	08/06/07 to 09/02/07	2 min
Set 11	Pas du Chat, Switzerland (46.029, 7.408) [17]	23 to 89	3,041	04/16/08 to 04/20/08	2 min
Set 12	Bern, Switzerland (46.948, 7.444) [17]	25 to 91	4,851	03/13/07 to 03/15/07	0.5 min
Set 13	Le Génépi, Switzerland (46.025, 7.044) [17]	6 to 93	42,141	08/28/07 to 10/31/07	2 min
Set 14	Matterhorn, Switzerland/Italy (45.976, 7.658) [18]	25 to 32	42,758	11/03/12 to 12/29/12	~2 min
Set 15	Chamonix, France (45.879, 6.887) [18]	32 to 44	66,365	10/01/12 to 12/29/12	~2 min

Despite the longer symbol lengths obtained when Set 3 is used, the proposed method performed better that LEC in all the cases presented in Table 4. It also outperformed ALFC's best case compression in every scenario, except when the dictionary generated by Set 3 is used to compress the symbols in sets with very low entropy. Again, one must take into consideration the simplicity of the proposed method.

Table 4 shows the percentage of symbols present in each dataset which cannot be represented by the dictionary generated using the dataset on the top row. The results show that the percentage of symbols not present in the dictionary is very close to zero in most cases. The percentage of symbols outside the dictionary is at most 4.39%, which would still allow the proposed method to perform quite well. It is interesting to notice that the highest percentages of symbols not present occur exactly when sets which contain a very limited number of symbols are used to generate the Huffman dictionaries (i.e., Sets 4, 5, and 8). However, for these very same sets, the maximum symbol length is significantly shorter (because fewer symbols are present), and as a consequence so is the length of the special symbol δ, which mitigates the impact of the insertion of additional uncompressed symbols. This can be verified in Table 3, where we can see that the average symbol lengths obtained using the dictionaries generated by sets that caused the highest percentages of symbols not present (especially Set 8) are not significantly higher than those obtained using other sets.

4.5. Evaluation Using Relative Humidity Measurements. The proposed approach can be applied to other environmental datasets. In order to demonstrate that, we consider a set of relative humidity measurements, whose characteristics are

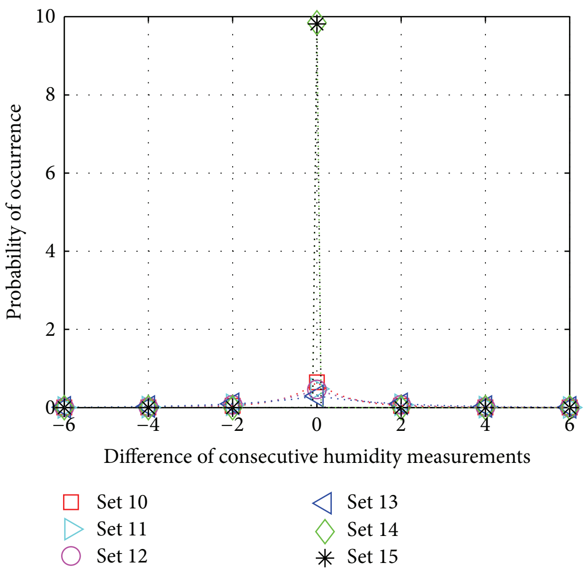

Figure 6: Probability distributions of the differences between consecutive measurements for each of the relative humidity datasets.

listed in Table 5. Figure 6 shows the probability distribution of the differences between consecutive relative humidity measurements for each of the datasets. Figure 7 shows the average symbol length for the uncompressed case (L_u), the entropy of the relative humidity measurements (H), and the entropy of the difference of consecutive relative humidity measurements (H_d).

The results concerning the utilization of the proposed scheme and of LEC are shown in Figure 8. In this case,

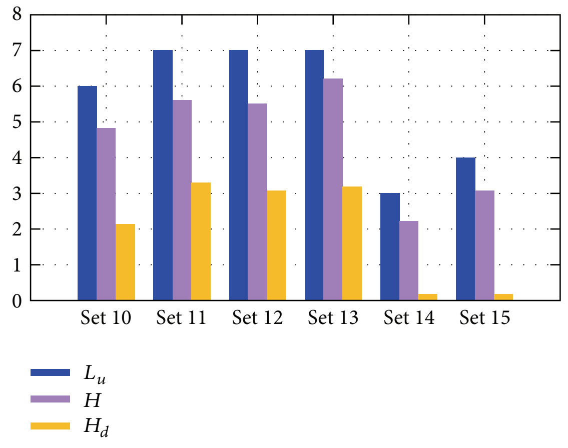

Figure 7: Average symbol length (L_u) for the uncompressed case, entropy of relative humidity measurements (H), and entropy of the differences of consecutive measurements (H_d).

Table 6: Proposed fixed Huffman dictionary for relative humidity datasets.

d_i	Codeword
−15	111111001000110
−14	1111110010001111111
−13	11111100100010
−12	1111110010000
−11	1111110010001111110
−10	11111100101
−9	11111100110
−8	1111000000
−7	111100001
−6	11110001
−5	1111001
−4	111110
−3	11001
−2	1101
−1	101
0	0
+1	100
+2	1110
+3	11000
+4	111101
+5	1111111
+6	11111101
+7	111111000
+8	11110000011
+9	11110000010
+10	1111110011101
+11	1111110011100
+12	111111001000111110
+13	111111001001
+14	1111110011111
+15	11111100100011110
+16	1111110011110
δ	1111110010001110

a Huffman dictionary (Table 6) was generated based on the measurements in Set 10 and used to compress the measurements in Sets 11 to 15 (if a temperature dataset, such as Set 1, is used to generate the dictionary, a degradation of up to 17.6% can be seen when compressing relative humidity datasets). The conclusions are very similar to those obtained for the temperature datasets. Once again the proposed scheme performs best, while approaching the theoretical limit for some of the datasets.

As we did for the temperature datasets, we also compared the performance of our approach to that of ALFC using the same set of parameters used in Section 4.2. Figure 9 shows the average symbol length (L) after compression using ALFC and our proposed approach for the six relative humidity datasets. Again, the proposed approach outperforms ALFC for any dataset or packet length.

5. Conclusions

This paper presents a lightweight compression mechanism for low resolution sensor nodes based on fixed Huffman dictionaries. Since the proposed scheme presents very modest computational and memory requirements, it can be easily employed in practical wireless sensor nodes. In order to evaluate the method, we computed the compression ratio obtained in several real datasets containing temperature and relative humidity measurements collected at different locations and during distinct periods of time. The compression ratios obtained using our approach vary between 46% and 82%. The code efficiency results also demonstrate that in some cases the proposed method closely approaches the theoretical limit. Finally, the proposed scheme, although extremely simple, outperforms LEC [2] for all the considered datasets and ALFC [4] in the vast majority of cases.

The most promising direction we envision for the future is to improve our ability of understanding the reference measurement datasets so that we can compensate for deficiencies in the dataset during dictionary generation. That is, we would like, for example, to analyze the relationship between measurement range and sample rate in order to adjust for any discrepancies before generating the dictionary. This would mitigate the impact in the performance of the method seen, for example, when Set 3 was used to generate the dictionary. In fact, taking one step further, this approach might allow us to establish synthetic measurement distributions for different environmental variables (e.g., temperature or relative humidity) which would allow sensor nodes to generate measurement dictionaries on-the-fly without the necessity of referring to reference datasets. For example, the experiments with temperature measurements in Section 4 showed that very high compression ratios are achieved when Set 1 is used to generate the Huffman dictionary. As Figure 1 shows, the distribution of the measurement differences in Set 1 can be obtained by sampling a Laplacian distribution with mean 0 and scale 1 at integer points. The dictionary in

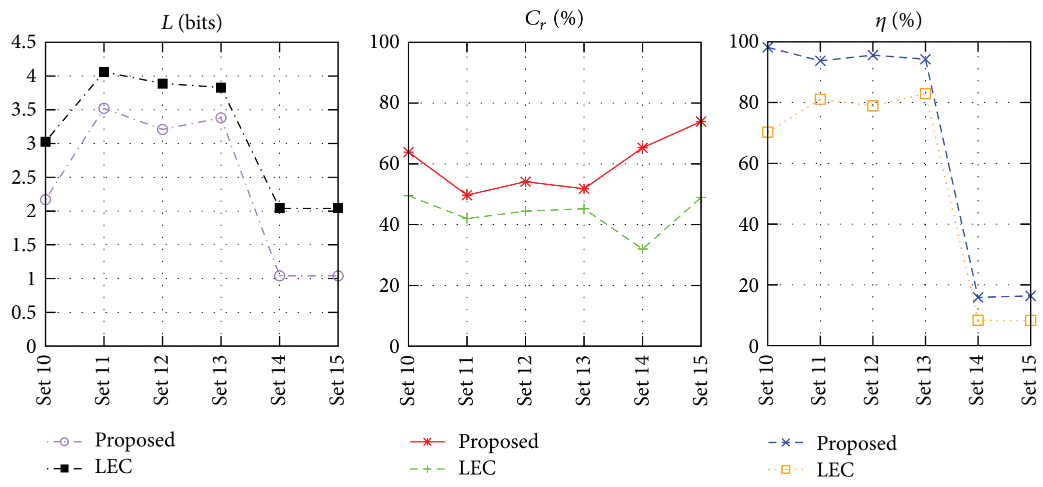

FIGURE 8: Average symbol length after compression (L), compression ratio (C_r), and code efficiency (η) for different relative humidity datasets.

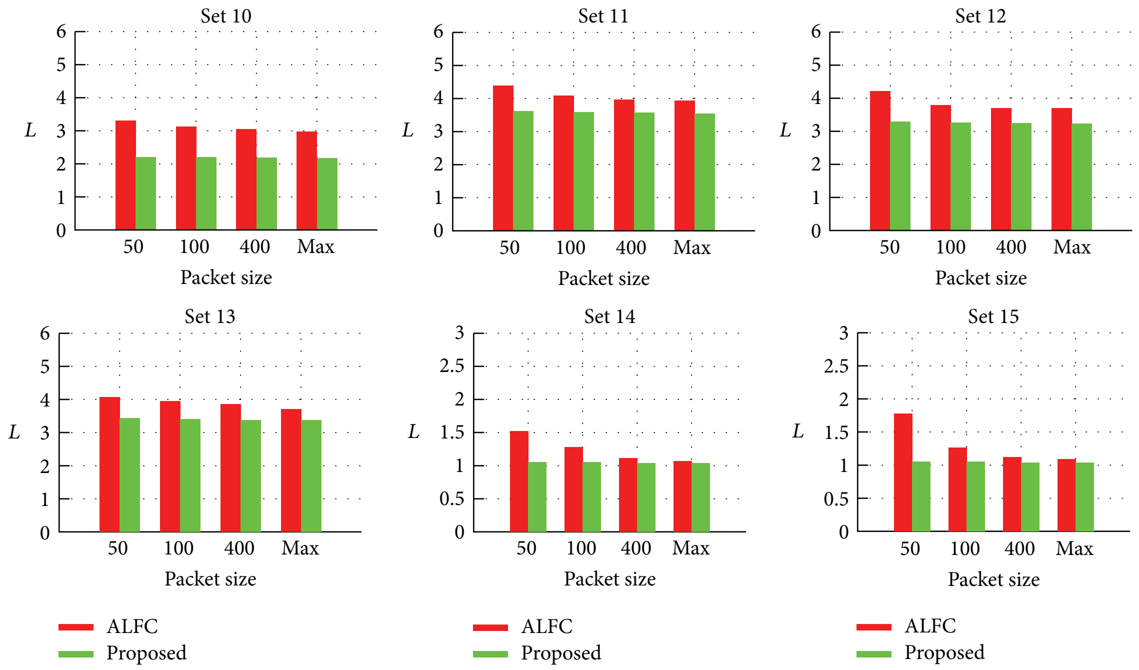

FIGURE 9: Average symbol length (L) (bits/sample) after compression using ALFC [4] and the proposed method for different packet sizes for relative humidity datasets.

Table 2 can then be generated by constructing a Huffman tree based on the probabilities of the points $[-8, \ldots, +10]$. This entire procedure could be carried out by a wireless sensor node based exclusively on the knowledge of the median and variance of the distribution and on the range of points that must be sampled without any additional information about the original dataset.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was partially supported by CNPq, CAPES, and FAPEAM (Brazil). Part of this work was presented (in Portuguese) in the 31st Brazilian Telecommunications Symposium.

References

[1] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki, "Practical data compression in wireless sensor networks: a survey," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 37–59, 2012.

[2] F. Marcelloni and M. Vecchio, "A simple algorithm for data compression in wireless sensor networks," *IEEE Communications Letters*, vol. 12, no. 6, pp. 411–413, 2008.

[3] F. Marcelloni and M. Vecchio, "An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks," *Computer Journal*, vol. 52, no. 8, pp. 969–987, 2009.

[4] A. Kiely, M. Xu, W.-Z. Song, R. Huang, and B. Shirazi, "Adaptive linear filtering compression on realtime sensor networks," *Computer Journal*, vol. 53, no. 10, pp. 1606–1620, 2010.

[5] T. Schoellhammer, E. Osterweil, B. Greenstein, M. Wimbrow, and D. Estrin, "Lightweight temporal compression of microclimate datasets," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN '04)*, pp. 516–524, November 2004.

[6] E. P. Capo-Chichi, H. Guyennet, and J.-M. Friedt, "K-RLE: a new data compression algorithm for wireless sensor network," in *Proceedings of the 3rd International Conference on Sensor Technologies and Applications (SENSORCOMM '09)*, pp. 502–507, Athens, Greece, June 2009.

[7] 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 '06)*, pp. 265–278, ACM, November 2006.

[8] A. Reinhardt, D. Christin, M. Hollick, J. Schmitt, P. S. Mogre, and R. Steinmetz, "Trimming the tree: tailoring adaptive Huffman coding to wireless sensor networks," *Wireless Sensor Networks*, vol. 5970, pp. 33–48, 2010.

[9] 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 '09)*, pp. 1–9, Rome, Italy, June 2009.

[10] J. Barcelo-Llado, A. Perez, and G. Seco-Granados, "Enhanced correlation estimators for distributed source coding in large wireless sensor networks," *IEEE Sensors Journal*, vol. 12, pp. 2799–2806, 2012.

[11] H. Arjmandi, M. Taki, and F. Lahouti, "Lifetime maximized data gathering in wireless sensor networks using limited-order distributed source coding," *Signal Processing*, vol. 91, no. 11, pp. 2661–2666, 2011.

[12] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–105, 2002.

[13] K. Romer, "Discovery of frequent distributed event patterns in sensor networks," in *Proceedings of the 5th European Conference on Wireless Sensor Networks (EWSN '08)*, pp. 106–124, Bologna, Italy, 2008.

[14] T. A. Welch, "A technique for high-performance data compression," *Computer*, vol. 17, no. 6, pp. 8–19, 1984.

[15] J. S. Vitter, "Design and analysis of dynamic huffman codes," *Journal of the ACM*, vol. 34, no. 4, pp. 825–845, 1987.

[16] Weather Underground, 2012, http://www.wunderground.com.

[17] Sensorscope, 2012, http://sensorscope.epfl.ch.

[18] Permasense, 2013, http://www.permasense.ch.

[19] S. S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): design and construction," *IEEE Transactions on Information Theory*, vol. 49, no. 3, pp. 626–643, 2003.

[20] J. Luo, L. Xiang, and C. Rosenberg, "Does compressed sensing improve the throughput of wireless sensor networks?" in *Proceedings of the IEEE International Conference on Communications (ICC '10)*, pp. 1–6, May 2010.

[21] K. Sayood, *Introduction to Data Compression*, Morgan Kaufman, 3rd edition, 2005.

[22] Y. Liang and W. Peng, "Minimizing energy consumptions in wireless sensor networks via two-modal transmission," *SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 12–18, 2010.

[23] D. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[24] D. Otto, "Avr-huffman," 2012, http://www.das-labor.org/wiki/AVR-Huffman/en.

[25] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[26] W. Golomb, "Run-length encodings," *IEEE Transactions on Information Theory*, vol. 12, no. 3, pp. 399–401, 1966.