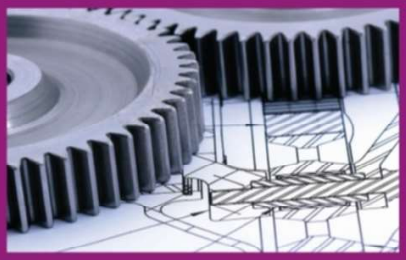
Volume 2 No. 1 - 2015

# NIGERIA JOURNAL OF ENGINEERING AND APPLIED SCIENCES (NJEAS)

# CONTENTS

# An Efficient and Robust Lossless Compression Scheme for Wireless Sensor Networks

Jonathan Gana Kolo[1], S. Anandan Shanmugam[2], Taliha Abiodun Folorunso[1], James Agajo[1] Abraham Usman Usman[3]
[1]Department of Computer Engineering, Federal University of Technology, PMB 65 Minna, Niger State, Nigeria
[2]Department of Electrical and Electronics Engineering, The University of Nottingham Malaysia Campus, Jalan Broga, 43500 Semenyih, Selangor Darul Ehsan, Malaysia
[3]Department of Electrical and Electronics Engineering, Federal University of Technology, PMB 65 Minna, Niger State, Nigeria

**Abstract**
In wireless sensor networks (WSNs), a large number of tiny, inexpensive and computable sensor nodes are usually deployed randomly to monitor one or more physical phenomena. The sensor nodes collect and process the sensed data and send the data to the sink wirelessly. However, WSNs have limitations such as tight energy budgets, limited radio bandwidth, limited memory, limited computational capability, limited packet size and high packet loss rates. These constrains are important issues when designing compression schemes for WSNs. Data compression is one important tool that can maximize data return over unreliable and low rate radio links. Thus, due to the unreliable nature of the radio links in WSNs that result in packet losses, it is therefore very critical to propose a data compression scheme that is very robust to packet losses. In this paper, we propose a block based approach which allows each block of source data to be encoded independently to ensure unique decodability at the sink, thus leading to an efficient and robust lossless compression scheme for WSNs. Simulation results using various real-world sensor datasets show that a maximum percentage energy saving of 29.29% was achieved by our proposed scheme. In addition, although the compression performance of our proposed scheme is comparable with those of LEC, it is however 200% as efficient as S-LZW.

**Keywords:** Energy Efficiency, Huffman Coding, Lossless Compression, Signal Processing, Wireless Sensor Networks.

## 1. Introduction

Wireless sensor networks (WSNs) is a cooperative network of small sized, inexpensive, computable and battery-operated wireless sensor nodes which are usually deployed randomly in large numbers to monitor one or more phenomena (Liao & Yang, 2012 Villas et al, 2012).

The nodes monitor their surroundings for local data and forward the gathered data to the sink node using direct or multi-hop communication. The sink node then processes all the received data from several source nodes and reports them to a monitoring facility. This network architecture allows a number of applications in areas such as industrial, environmental, military and medical (Kolo, Shanmugam, Lim, Ang, & Seng, 2013; Villas et al., 2012).

WSNs are highly energy-constrained because the sensor nodes are battery-operated. Other limitations of WSNs include limited memory, limited radio bandwidth, limited computational capability, high packet loss rates and limited packet size. Data communication among sensor

nodes is the main source of energy consumption in WSNs. To achieve energy conservation, the data rate should be aggressively reduced. The amount of energy required for transmission can be greatly minimized if the volume of data to be transmitted is reduced. The reduction in data size will further lead to considerable savings in power. Research shows, saving a byte of data through data compression has been shown to worth spending between four thousand (Chipcon CC2420) to two million (MaxStream XTend) cycles of computation (Sadler & Martonosi, 2006.Therefore, compressing data before transmission is one of the major strategies for energy-efficient WSNs and this saving would directly translate into lifetime extension for the network nodes (Schoellhammer, Greenstein, Osterweil, Wimbrow, & Estrin, 2004. In addition, the reduction in data size will further lead to reduce network load which lead to fewer collisions and retransmissions.

In general, data compression techniques can be divided into two main categories, lossless and lossy. In the lossless compression techniques, original data can be retained exactly without any loss but on the expense of compression ratio and system complexity. Examples of lossless data compression techniques are Run Length Encoding (RLE), Huffman coding, arithmetic coding and Lempel–Ziv (LZ) coding. On the other hand, lossy data compression permits a certain degree of inaccuracy in the reconstructed data but the compression ratio will be significantly high. Examples of lossy compression techniques are prediction, quantization, fitting, transform coding and compressive sensing. Some compression techniques may use combination of two or more techniques such as encoders of multi media data (Alsalaet & Ali, 2015; Wu, Tan, & Xiong, 2016.

Furthermore, many of the data compression schemes proposed in the literature do not take packet losses into consideration. Hence, those compression schemes cannot perform well due to the unreliable nature of the radio links in WSNs. It is therefore very critical to propose a data compression scheme that is very robust to packet losses. In this paper, we propose An Efficient and Robust Lossless Compression Scheme (AERLCS) based on the basic LEC (F. Marcelloni & Vecchin; 2009) for WSNs. AERLCS is block-based; as such each block of source data is independently encoded to ensure unique decoding at the sink. The AERLCS is simple and lightweight and very suitable for resource-constrained WSNs.

The remaining part of this paper is divided into the following sections: A review of past and related works to the subject matter is presented in section 2 while in section 3, a detailed description of our proposed AERLCS WSNs is presented. In section 4, we present a detailed performance evaluation of AERLCS by analyzing the obtained simulation results as well as the results of comparison with the LEC and S-LZW schemes. Finally, in section 5, we present our conclusions.

**Related Work**

There exist two main categories of data compression scheme in literature namely; (i) distributed data compression schemes, and (ii) local data compression schemes.

Characteristically, the distributed data compression approach exploits the high spatial correlation in data from fixed sensors node in dense networks with some of the main techniques under this approach include distributed source coding (DSC) (Chou, Petrovic, & Ramchandran, 2003; Pradhan, Kusuma, & Ramchandran, 2002), distributed transform coding (DTC) (A Ciancio & Ortega, 2005; Alexandre Ciancio, Patem, Ortega, & Krishnamachari, 2006), distributed source modeling (DSM) (Predd, Kulkarni, & Poor, 2006; Xiao, Cui, Luo, & Goldsmith, 2006) and compressed sensing (CS) (Donoho, 2006). The distributed compression approach however conserves energy at the expense of information loss in the source data and for this reason will not be considered.

Conversely, the local data compression approach which takes advantage of the temporal correlation that exist in sampled sensor data to perform its compression locally on each sensor node will be considered. Some of the proposed local data compression algorithms based on temporal correlation in WSNs include: lossy algorithms (Lightweight Temporal Compression (LTC) (Schoellhamme et al., 2004), K-RLE (Capo-Chichi, Guyennet, & Friedt, 2009), Differential Pulse Code Modulation-based Optimization (DPCM-Optimization) (Francesco Marcelloni & Vecchio, 2010); lossless algorithms (Sensor node LZW (S-LZW) (Sadler & Martonosi, 2006), Lossless Entropy Compression (LEC) (F. Marcelloni & Vecchio, 2009) Modified Adaptive Huffman Compression Scheme, (Tharini & Randan, 2009), Two-Modal Transmission (TMT) (Liang & Peng, 2010), modified Lossless Entropy Compression (mLEC) (Kolo, Ang, Seng & Prabaharan, 2013). The precision required by some application domains demands sensor nodes with very high accuracy that cannot tolerate measured data being corrupted by the compression process. Thus, this research is directed to focus on lossless local data compression algorithms. (Sadler & Martonosi, 2006) introduced a lossless compression algorithm called S-LZW which is an adapted version of LZW (Welch, 1984) designed specifically for resource constrained sensor nodes. S-LZW is a dictionary-based low complexity lossless compression algorithm that divides the uncompressed input bit streams into fixed size blocks of 528 bytes (two flash pages) and compresses each block separately. The dictionary structure allows the algorithm to adapt to changes in the input and to take advantage of repetition in the sensed data. However, the algorithm suffers from the growing dictionary problem and its compression efficiency still needs to be improved. Also, it is less robust to packet losses because it encodes 528 bytes of source data into 10 or more dependent packets. Hence, once a packet is lost in transmission, the remaining packets following in the same group cannot be decoded. (F. Marce;;pmi & Vecchio, 2009), introduced the Huffman coding into wireless sensor nodes. Their simple lossless entropy compression (LEC) algorithm which was based on static Huffman coding, exploring the temporal correlation that exist in sensor data to compute a compressed version using an

extended coding table that was used in the baseline JPEG algorithm for compressing the DC coefficients (Pennebaker & Mitchell, 1992). The algorithm was particularly suitable for computational and memory resource constrained sensor nodes. The algorithm gives good compression performance. However, one fundamental drawback of the LEC scheme is the interdependence of one packet on the other. As such, once a packet is lost in transmission, all other packets following cannot be decoded. Furthermore, a modified version of the classical adaptive Huffman coding was proposed in (Tharini & Ranjan, 2009). The modified algorithm does not require prior knowledge of the statistics of the source data and compression is performed adaptively based on the temporal correlation that exists in the source data. The drawback of this algorithm is that it is computationally intensive and it is not robust to packet losses. In lieu of the aforementioned draws backs in the earlier algorithms, the proposed AERLCS offers solutions to the raised shortcomings. Being a modified version of LEC, the proposed algorithm requires low computational resource and gives good compression performance. The AERLCS also encodes each block of source data independently, thereby being very robust to packet losses.

*The AERLCS Algorithm*

The proposed efficient and robust lossless compression scheme (AERLCS) is described in details in this section. The scheme is formed from the basic LEC (F. Marcelloni & Vecchio, 2009) coding method for WSNs. Although the LEC scheme is simple, its compression performance is good and far better than earlier schemes like S-LZW (Sadler & Martonosi, 2006) among others. However, the interdependence of packets on themselves is one of the main fundamental drawbacks of the LEC. As such, once a packet is lost in transmission, other packets following cannot be decoded. Thus, given the nature of unreliable wireless communication channels of WSNs, the proposed modified version of the LEC provides the flexibility of encoding blocks of source data independently making it significantly more robust to packet losses than LEC. Since wireless sensor nodes transmit data in packets, AERLCS explores this inherent transmission mode by compression source data in blocks. AERLCS also enjoys the high temporal correlation that typically exists between consecutive sensed samples to lossless compress data.

*Prediction*

The proposed scheme depends on the principle of entropy coding of exponentially distributed source. In practice, most of the sources encountered are not exponentially distributed; as such the original sources have to be preprocessed to ensure that they fit exponential distribution. The principle of predictive coding is applied owing to the fact that predictive errors are often modeled using exponential distribution. To ensure a corresponding low level computation to the relatively to the low computational power WSN, we have adopted the use of a linear prediction model that is limited to taking the differences between consecutive sampled data. For our intended

application which is the compression of environmental data such as temperature and relative humidity, the prediction model proves to be simple and efficient. In addition, it's also ensures that the computational complexity of our compression scheme is as low as possible. Thus, the predicted sample

$$\hat{x}_i = x_{i-1} \quad (1)$$

That is, the predicted sample is equal to the last observed sample. The error term (residue) is then calculated by subtracting the predicted sample from the current sample. Hence, the residue $d_i$ is the difference.

$$d_i = x_i - \hat{x}_i \quad (2)$$

Where $i = 1, 2, 3 \ldots$

In other to compute the first residue we assume that

$$x_0 = 0 \quad (3)$$

Equation (3) ensures that the first sample of each block of residues is the reference sample (original sample value). The residues $d_i$ are now exponentially distributed.

*Entropy Coding*

In order to achieve energy conservation and also to maximize data return over unreliable and low rate radio links, we propose to implement an efficient and robust lossless compression algorithm that compresses blocks of sampled data independently at a time using the basic LEC encoder. Our proposed algorithm uses the basic entropy encoder proposed in Marcelloni & Vecchio, (2009) with its coding table optimized for WSNs sensed data to encode each block independently and can be applied to multiple data types. The encoder performs compression losslessly by encoding differences $d_i$ more compactly in accordance with the pseudo-code described in Figure 1. Each $d_i$ is represented as a bit sequence $c_i$ composed of two parts $h_i$ and $l_i$ ($c_i = h_i * l_i$).

$$l_i = (Index)|_b \quad (4)$$

Where

$$b_i = \lceil \log_2(|d_i|) \rceil \quad (5)$$

$$Index = \begin{cases} d_i & d_i \geq 0 \\ (2^{b_i} - 1) + d_i & d_i < 0 \end{cases} \quad (6)$$

Equation (6) returns the index position of each $d_i$ within its group. $(\text{Index})|_{bi}$ denotes the binary representation of *Index* over $b_i$ bits. $b_i$ is the category (group number) of $d_i$. It is also the number of lower order bits needed to encode the value of $d_i$. Note that if $d_i = 0$, $l_i$ is not represented. Thus, at that instance, $c_i = h_i$. Once $c_i$ is generated, it is appended to the bit stream which forms the compressed version of the source data.

To efficiently and independently encode block of n source samples $x_i$ at a time using our proposed algorithm the under listed procedural steps is followed;

1. Using equation (2) and (3), convert the block of n source samples $x_i$ to blocks of n residue samples $d_i$ Note that the first residual sample is the reference sample.
2. Encode the blocks of n residue samples using the basic LEC entropy encoder with the optimized coding table (Table 1).
3. Send the encoded bit-stream to the sink.

Although our modification is simple, it ensures that each packet is independently decodable. That is, it provides significant robustness to packets losses.

```
encode(di, TABLE)
// di is the current residue value
// TABLE is the variable length Huffman codes used in encoding
// bi is the category (group number) of di
// bi is also the number of lower order bits needed to encode the value of di
// ci is the encoded bitstream of di
// hi is the variable-length Huffman code that codifies the category (group) of di
// Ii is the variable-length integer code that codifies the index position of di
//    within its group (category)
// * denotes concatenation
// (Index)|bi denotes the binary representation of index over bi bits

// compute di category
IF di = 0 THEN
   SET bi TO 0
ELSE
   SET bi TO Glog2(|di|)?
ENDIF
// extract hi the variable length Huffman code from TABLE
SET hi TO TABLE [bi]
// build ci
IF bi = 0 THEN
   // Ii is not needed
   SET ci TO hi
ELSE
   // build Ii
   SET Ii TO (Index)|bi
   // build ci
   SET ci TO hi * Ii
ENDIF
RETURN ci
```

**Figure 1** The pseudo-code of the encode algorithm

**Table 1** Optimized Huffman Coding Table

| $b_i$ | $h_i$ | $d_i$ |
|---|---|---|
| 0 | 100 | 0 |
| 1 | 110 | - 1,+1 |
| 2 | 00 | - 3,- 2,+2,+3 |
| 3 | 111 | - 7, . . . ,- 4,+4, . . .,+7 |
| 4 | 101 | - 15, . . . ,- 8,+8, . . .,+15 |
| 5 | 010 | - 31, . . . ,- 16,+16, . . .,+31 |
| 6 | 0111 | - 63, . . . ,- 32,+32, . . .,+63 |
| 7 | 01101 | - 127, . . . ,- 64,+64, . . .,+127 |
| 8 | 011001 | - 255, . . . ,- 128,+128, . . .,+255 |
| 9 | 0110001 | - 511, . . . ,- 256,+256, . . .,+511 |
| 10 | 01100001 | - 1023, . . . ,- 512,+512, . . .,+1023 |
| 11 | 011000001 | - 2047, . . . ,- 1024,+1024, . . .,+2047 |
| 12 | 01100000000 | - 4095, . . . ,- 2048,+2048, . . .,+4095 |
| 13 | 01100000001 | - 8191, . . . ,- 4096,+4096, . . .,+8191 |
| 14 | 01100000010 | - 16383, . . . ,- 8192,+8192, . . .,+16383 |

## Simulations and Analysis

To demonstrate the effectiveness of our proposed algorithm was tested against various real-world environmental datasets (Temperature and relative humidity datasets were considered). The performance evaluation of the algorithm is computed by using compression rate (*cr*) which is defined as the ratio of the number of bits to represent the compressed data to the number of samples of the original data. The compression rate (Cr) is defined in Equation 7 and the smaller the value of the compression rate, the better the compression performance of the compression algorithm.

$$cr = \frac{CS}{NS} \text{ bits per sample} \qquad (7)$$

Where *CS* is the compressed data size in bits and *NS* is the number of samples of the original data

*Data Sets*

Real-world environmental monitoring WSN data sets from Sensor Scope ("Sensor Scope deployments homepage." 2011) were used in our simulations. The relative humidity and temperature measurements from three Sensor Scope deployments: Le Gènèpi Deployment, HES-SO Fish Net Deployment and LUCE Deployment were used. Publicly accessible data sets were used to make the comparison as fair as possible. These deployments use a TinyNode node ("TinyNode homepage,") which comprises of a TI MSP430 microcontroller, a Xemics XE1205 radio and a Sensirion SHT75 sensor module ("Sensirion homepage," 2011).

Both the relative humidity and temperature sensors are connected to a 14 bit analog to digital converter (ADC). The default measurement resolution for raw relative humidity (*raw_h*) and raw temperature (*raw_t*) is 12 bits and 14 bits respectively. Each ADC output *raw_h* and *raw_t* are converted into

measure $h$ and $t$ in percentage and degree Celsius respectively as described in ("Sensirion homepage," 2011). The data sets that are published on SensorScope deployments correspond to physical measures $h$ and $t$. But the compression algorithms work on *raw_h* and *raw_t*. Therefore, before applying the compression algorithm, the physical measures $h$ and $t$ are converted to *raw_h* and *raw_t* by using the inverted versions of the conversion functions in ("Sensirion homepage," 2011). Table 2 summarizes the main characteristics of the datasets. We compute the information entropy $H$ of the original data sets and the information entropy $H_d$ of the preprocessed samples of each data sets using equation (8) and (9) respectively. $R$ is the number of possible values of $x_i$ (or $d_i$ ) and $p(x_i)$ (or $p(d_i)$ is the probability mass function of $x_i$ (or $d_i$ ) These are all recorded in Table 3. By preprocessing the data sets as can be seen from Table 3, the compressibility of the data sets have been enhanced since information entropy represents an absolute limit on the best possible lossless compression of any source, under certain constraints (i.e. under the assumption that the source is a memoryless source).

$$H = -\sum_{i=1}^{R} p(x_i).\log_2 p(x_i) \qquad (8)$$

$$H_d = -\sum_{i=1}^{R} p(d_i).\log_2 p(d_i) \qquad (9)$$

**Table 2** Main characteristic of the datasets

| Deployment name | Node ID | Symbolic name | Number of samples | Time interval | |
|---|---|---|---|---|---|
| | | | | From day | To day |
| LUCE | 84 | LU84 | 64913 | 23-Nov-06 | 17-Dec-06 |
| HES-SO FishNet | 101 | FN101 | 12652 | 09-Aug-07 | 31-Aug-07 |
| Le Génépi | 20 | LG20 | 21523 | 04-Sep-07 | 03-Oct-07 |

**Table 3** Information entropy of the original and the preprocessed samples of each data set

| DATASET | $H$ | $H_d$ |
|---|---|---|
| LU84 TEMP | 10.0677 | 4.0471 |
| FN101TEMP | 10.2609 | 5.0965 |
| LG20 TEMP | 10.2492 | 6.8178 |
| LU84 RH | 9.9156 | 5.7032 |
| FN101RH | 9.6070 | 5.7313 |
| LG20 RH | 10.7634 | 7.6052 |

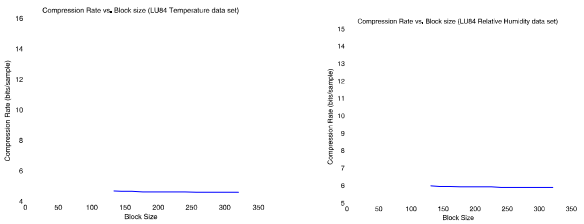
*Compression performance of our proposed algorithm*

In this section, the compression performance of our proposed scheme using the six real-world data sets discussed in section 4.1. For each data set, using the improved coding table, the compression performance of AERLCS is computed for different value of $n$ (block size) using (7). Figure 2 shows the compression rate vs. block size achieved by AERLCS for the six real-world data sets. As evident from Figure 2, the compression rate of AERLCS for each of the six data sets decreases with respect to the increase in the block size. For very small values of $n$, high compression rates were obtained due to high reference sample overhead cost incurred that over weigh the compression benefits. In fact, data expansion results at block size of 2 for all the six test data sets. However, the compressed bit rates obtained were significantly low for block sizes in the range $n$ = 48, this is because the smaller the value of the compression rate, the better the compression performance of the compression algorithm. Values of $n$ (block size) beyond 96 results in less improvement to the compression rate as evident from Figure 2. Thus, for real-time and near real-time operation using AERLCS, the block size $n$

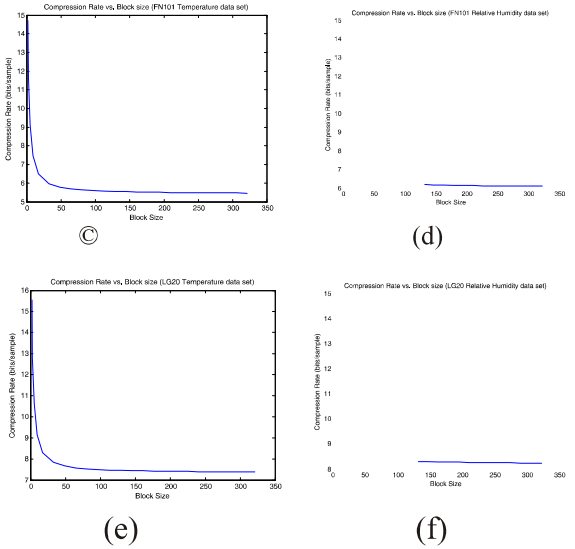
should be in the range $48 = n = 96$ for optimum performance. It is observed that at the limits (when the block size equals the size of each data source), the compression performance of AERLCS is better than that obtained by the basic LEC algorithm. In addition, while the basic LEC algorithm is not robust to packet losses, our proposed scheme is very robust to packet losses since it encodes each data block independently.

*Performance comparison between LEC coding and our improved coding*

To demonstrate the effectiveness of our improved coding table, in this section compare the compression performances obtained by AERLCS with those of the LEC coding table. For ease of comparison, the corresponding figures of all the six data sets are on the same plot as shown in Figure 3 (a-f). From the plots, it can be seen that the compression rate performance achieved using the AERLCS (Improved) coding table is better than those achieved using the LEC coding table for five out of the six test data sets. The improved performance is as a result of our optimization of the coding table for environmental monitoring data sets. The LEC coding table is optimized for the compression of the DC coefficients of a digital image (Pennebaker & Mitchell, 1992) which are highly correlated.



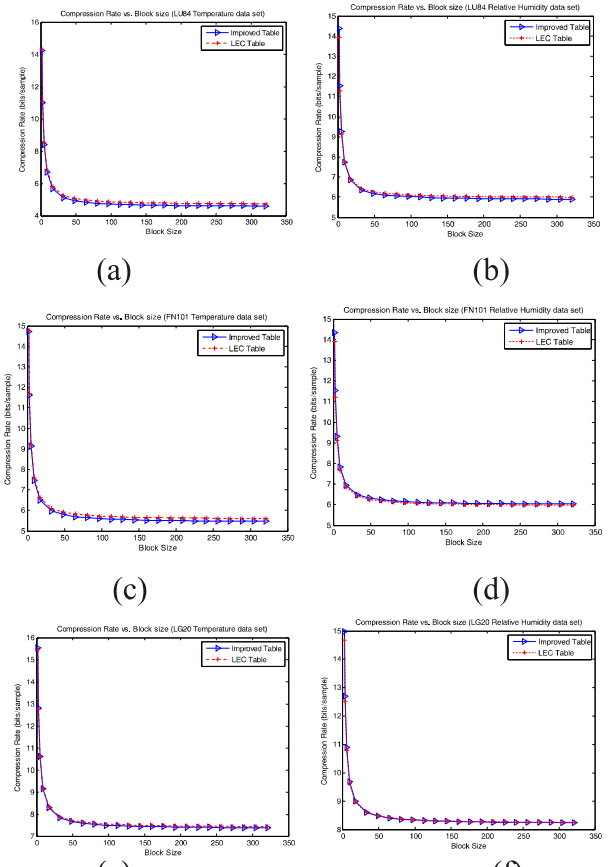

(a) (b)




© (d)

(e) (f)

**Figure 2** Compression rate vs. block size achieved by our proposed algorithm for the six test datasets




(a) (b)

(c) (d)

(e) (f)

**Figure 3** Performance comparisons between our improved coding Table and the LEC coding Table for the six test datasets

The improved (AERLCS) compression rate performance achieved as a result of using our improved coding table have help to offset to some extend the performance degradation that results due to reference sample overhead cost. Thus, for real-time and near real-time operation, the compression performance of AERLCS is comparable with the basic LEC compression performance. For delay-tolerant operation, our proposed algorithm outperforms the basic LEC algorithm.

*Performance comparison with S-LZW Algorithm*

In this section, the evaluation of the compression performance of AERLCS with S-LZW algorithm (Sadler & Martonosi, 2006) for WSNs is discussed. S-LZW is a dictionary-based low complexity lossless compression algorithm that divides the uncompressed input bit streams into fixed size blocks of 528 bytes (two flash pages) and compresses each block separately. From S-LZW coding mechanism, it is much less robust to packet losses than AERLCS. Table 4 gives the compressed bit rates that was achieved on all the six data sets by the S-LZW algorithm with the following fixed parameters: Mini-Cache Entries=32, Max Dictionary Entries=512, Block Size=528 bytes, and Dictionary Strategy=Frozen (Sadler & Martonoso, 2006). From this fixed parameters, it can be seen that S-LZW uses significantly more memory space for its dictionary entries and mini-cache entries than our proposed algorithm. Also, the block size of 528 bytes suggests that S-LZW can only be applied in delay-tolerant applications while AERLCS can be applied in both real-time and delay-tolerant applications. To make the compression performance comparison between AERLCS and S-LZW as fair as possible, we compress temperature and relative humidity data sets at block size of 302 ($14\text{bits} \times 302 \approx 528\text{bytes}$) and 352 ($12\text{bits} \times 352 = 528\text{bytes}$) respectively using AERLCS. These compression performances are also recorded in Table 4. From the compression results in Table 4, AERLCS is 151% to 203% as efficient as S-LZW. The worst compression performance of S-LZW is recorded when it is used to compress the LG20RH data set. The samples of the LG20RH data set are originally represented by 12-bit and after compression by S-LZW, the resulting compressed bit rate is 12.4915 bits/sample (i.e. data expansion results). In addition, while S-LZW algorithm suffers from the growing dictionary problem AERLCS on the other hand requires only small dictionary size (the size of the ADC) for its coding operation.

**Table 4.** Performance comparison between S-LZW and AERLCS in bits/sample at block size of 528 bytes

| DATASET | *S-LZW* | *AERLCS* |
|---|---|---|
| LU84 TEMP | 8.1628 | 4.5940 |
| FN101TEMP | 11.1472 | 5.4742 |
| LG20 TEMP | 12.477 | 7.3847 |
| LU84 RH | 11.0012 | 5.8953 |
| FN101RH | 10.1956 | 6.1036 |
| LG20 RH | 12.4915 | 8.2401 |

*Complexity and Energy saving*

In terms of algorithm complexity, AERLCS is simple. When compared to the basic LEC algorithm, AERLCS takes the same amount of time to compress data and requires the same

amount of memory. When compared to the S-LZW, AERLCS requires much less memory. In , Marcelloni & Vecchio, (2009), in other to assess the complexity of the basic LEC algorithm, the authors execute the LEC algorithm on SimIt-Arm simulator ("Simit-ARM Simulator," 2011) to compress the first block of each data set that consist of 528 bytes. They observed that on average, the LEC algorithm executes: (i) 30549.25 instructions per 528 bytes for the temperature data sets. (ii) 11.35 instructions for each saved bit of temperature data set. Thus, the LEC algorithm and by extension AERLCS is simple and lightweight. Exploiting this amount of work that has already been done as stated above and since our proposed scheme has the same complexity (in time and space) with the LEC algorithm, we therefore assume that AERLCS also executes approximately 30549.25 instructions per 528 bytes for the temperature data sets (which corresponds to the execution of on average 57.8584 instructions per bytes of the original temperature data set).

For the energy saving calculations, we considered the LU84 temperature data set. The size of LU84 temperature data set is 64913 14-bit samples. Next, we assume that the number of instructions executed in compressing a byte by the SimIt-Arm simulator corresponds to the number of instructions executed by the Tmote Sky when compressing a byte of data. In (Lane & Campbell, 2006), the authors state that the energy expended for the execution of instructions depends on the memory address mode of the instruction operands and on the type of instructions. We therefore conclude that, from the type of instruction used by AERLCS, on average the energy consumed in executing each instruction is 15 $n$J (Lane & Camplell, 2006; F. Marcelloni & Vecchio, 2009) . Also, for these calculations, considering the Cc2420 radio that is on-board the Tmote Sky motes from MoteIv, the time spent and the current consumed for transmitting one byte is 32$\mu$s and 17.4$m$A respectively. Table 5 list the essential parameter used in our evaluation. Given $N_{ins}$, the total number of instructions executed in compressing the whole data set is

$$NT_{ins} = \frac{N_{ins} \times NS \times ADC}{8} \tag{10}$$

Where $N_{ins}$, $NS$ and $ADC$ are as defined in Table 5. The computational energy in Joules consumed in compressing the whole data set is written as

$$E_{compute} = E_{ins} \times NT_{ins} \tag{11}$$

Where $E_{ins}$ *is* as defined in Table 5.

For a given block size ($B_{size}$), the fixed packet payload length ($s$) in bytes is determined from the expression

$$s = \left\lceil \frac{B_{size} \times cr}{8} \right\rceil \tag{12}$$

Note that, the Compression rate ($cr$) is a function of $B_{size}$ (see Figure 2) and is determined by the compression algorithm.

Next, we determine the number of packets needed to transmit the uncompressed data. This is defined by

$$NP_{raw} = \left\lceil \frac{NS \times ADC}{s \times 8} \right\rceil \tag{13}$$

Similarly, we determine the number of packets needed to transmit the compressed data. This is

defined by

$$NP_{compress} = \left\lceil \frac{CS}{s \times 8} \right\rceil \quad (14)$$

Where $CS$ is the compressed data size in bits.

The energy consumed in transmitting one byte of data ($E_{byte}$) is defined by

$$E_{byte} = V_{cc} \times I_{byte} \times t_{byte} \quad (15)$$

Thus, the energy consumed in transmitting one packet is given as

$$E_p = E_{byte} \times L_p \quad (16)$$

Where $L_p = (h + s)$ is the length in bytes of the transmitted packet. While $h$ is the packet header in bytes, $s$ is the packet payload in bytes.

Next, we determine the energy consumed by the radio in transmitting the uncompressed data. This is given by the expression

$$E_{Traw} = E_p \times NP_{raw} \quad (17)$$

Similarly, the energy consumed by the radio in transmitting the compressed data is given by

$$E_{Tcompress} = E_p \times NP_{compress} \quad (18)$$

Thus, the energy saving ($E_{save}$) is computed by subtracting (11) and (18) from (17). That is

$$E_{save} = E_{Traw} - E_{compute} - E_{Tcompress} \quad (19)$$

The percentage energy saving ($\eta$) is calculated using the expression

$$\eta = \frac{E_{save}}{E_{Traw}} \times 100 \quad (20)$$

**Table 5** Essential Energy Saving Evaluation Parameter

| Parameter | value |
|---|---|
| Size of LU84 temp data set ($NS$) | 64913 samples |
| Analog -to-Digital converter resolution ($ADC$) | 14 |
| Supply voltage ($v_{cc}$) | 3 V |
| The time it takes to transmit one byte of data ($t_{byte}$) | 32 µs |
| The current drawn in active mode while transmitting one byte of data ($I_{byte}$) | 17.4 mA |
| Average energy consumed for the execution of one instruction ($E_{ins}$) | 15 nJ |
| Number of instructions executed while compressing one byte of data ($N_{ins}$) | 57.8584 |
| Block size ($B_{size}$) | 2 - 480 samples |
| Packet header length ($h$) | 9 bytes |

Block size, compression rate and packet payload length are important parameters that affect the percentage energy saving. In our numerical simulations in MATLAB, using our proposed scheme and varying the block size from 2 to 480 samples, we determine the compression rates. Next, using (10) to (20), we compute the corresponding packet payload lengths and the percentage energy savings with respect to the energy consumed to transmit uncompressed data. Figure 4 depicts the plot of the percentage energy saving against the packet payload length. The followings were observed during the numerical simulations: (i) The energy consumption per packet increases with the increase in the packet payload length. This is due to the increase in the packet length. (ii) The energy consumption of the compressed data packets decreases with the increase in the packet payload length. This is partly due to the lower overhead cost of larger packets and partly due to lower compression rate and large packet size that result in fewer number of packets. (iii) The energy consumption of the uncompress data packets decreases with the increase in the packet payload length. This is due to the lower

overhead cost of larger packets and few numbers of packets due to large packet size.

For packet payload length of 4 bytes, the percentage energy saving achieved is -17.70% as seen in Figure 4. At that instance, the block size is 2 and the compression rate obtained is 14.2396. Data expansion result. That is, the compressed data size is greater than the uncompress data size. However, as the packet payload length increases beyond 4 bytes, the percentage energy savings increases until at payload length of 12 bytes when the maximum percentage energy saving of 29.29% occurs. Beyond this point ($s$ = 12 bytes), the percentage energy saving decreases gradually and at payload length of 275 bytes, the percentage energy saving is 17.19%.

For the LU84 temperature data set, the Plot depicted in Figure 4 gives the maximum percentage energy saving achievable for a given fixed packet payload length. Thus, if we use the TinyOS default data payload length of 29 bytes, the percentage energy saving from the plot is 24.98%. If we set the data payload length to 56 bytes, the percentage energy saving from the plot is 21.39%.

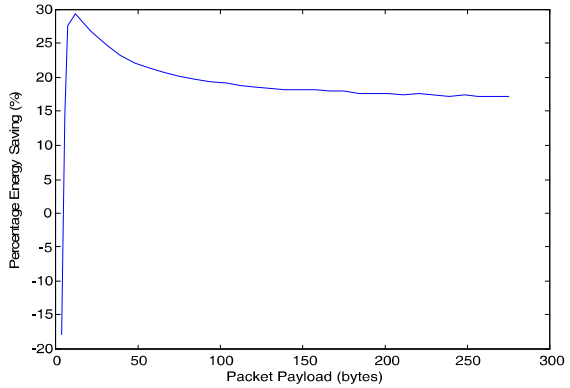



**Figure 4** Percentage Energy Saving versus Packet Payload for the LU 84 temperature data set

Therefore, 20% to 29% of the energy consumed to transmit uncompressed data can be saved through compression as a result of the use of AERLCS for near real-time and real-time applications. The energy savings by AERLCS is slightly less than that obtained by the LEC algorithm. However, AERLCS is more suitable to use in WSNs than LEC because while AERLCS is very robust to packet losses LEC is not.

## Conclusion

There are many literatures that discuss data compression schemes in WSNs. However, many of the data compression schemes proposed do not take packet losses into consideration. Hence, those compression schemes cannot perform well due to the unreliable nature of the radio links in WSNs. In this paper, we have proposed an efficient and robust lossless compression scheme for WSNs. AERLCS which is block-based encodes each block of data independently. As such, side information does not need to be carried across packet boundaries. Hence, AERLCS is highly robust to packet losses. Simulation results show that AERLCS compression performance is comparable with those of LEC. It however outperforms S-LZW. Thus, AERLCS has good compression performance which led to reduced power consumption and network lifetime extension. AERLCS is suitable for both real-time and delay-tolerant compression applications.

## References


Alsalaet, J. K., & Ali, A. A. (2015). Data compression in wireless sensors network using MDCT and embedded harmonic coding. *ISA Transactions*, *56*, 261-267. Elsevier. doi:10.1016/j. isatra. 2014. 11.023

Capo-Chichi, E. P., Guyennet, H., & Friedt, J.-M. (2009). K-RLE: A New Data Compression Algorithm for Wireless Sensor Network. *2009 Third International Conference on Sensor Technologies and Applications*, 502-507. Ieee. doi: 10.1109/SENSORCOMM.2009.84

Chou, J., Petrovic, D., & Ramchandran, K. (2003). A Distributed and Adaptive Signal Processing Approach to Reducing Energy Consumption in Sensor Networks. *Proceedings of IEEE INFOCOM, 2003* (Vol. 00, pp. 1054 – 1062).

Ciancio, A, & Ortega, A. (2005). A distributed wavelet compression algorithm for wireless multihop sensor networks using lifting. *IEEE Int Conf Acoustics Speech Signal Processing ICASSP* (Vol. IV, pp. 825-828).

Ciancio, Alexandre, Pattem, S., Ortega, A., &

Krishnamachari, B. (2006). Energy-Efficient Data Representation and Routing for Wireless Sensor Networks Based on a Distributed Wavelet Compression Algorithm. *Proceedings of the Fifth international Conference on Information Processing in Sensor Networks*, 309-316.

Donoho, D. L. (2006). Compressed Sensing. *IEEE Transactions on Information Theory*, *52*(4), 1289-1306.

Kolo, J. G., Ang, L.-M., Seng, K. P., & Prabaharan, S. R. S. (2013). Performance Comparison of Data Compression Algorithms for Environmental Monitoring Wireless Sensor Networks. *Int. J. Computer Applications in Technology*, *46*(1), 65-75. doi:10.1504/IJCAT.2013.051389

Kolo, J. G., Shanmugam, S. A., Lim, D. W. G., Ang, L.-M., & Seng, K. P. (2012). An Adaptive Lossless Data Compression Scheme for Wireless Sensor Networks. *Journal of Sensors*, *2012*, 20 pages. doi:10.1155/2012/539638

Lane, N. D., & Campbell, A. T. (2006). The Influence of Microprocessor Instructions on the Energy Consumption of Wireless Sensor Networks. *Proc. Third Workshop on Embedded Networked Sensors, Cambridge, MA, May 30–31, IEEE Press, Piscataway, NJ, USA* (Vol. 34, pp. 41-45). Citeseer. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.2273&rep=rep1&type=pdf

Liang, Y., & Peng, W. (2010). Minimizing Energy Consumptions in Wireless Sensor Networks via Two-Modal Transmission. *Computer Communication Review*, *40*(1), 13-18.

Liao, W.-H., & Yang, H.-C. (2012). A power-saving data storage scheme for wireless sensor networks. *Journal of Network and Computer Applications*, *35*(2), 818-825. Elsevier. doi:10.1016/j.jnca.2011.11.015

Marcelloni, F., & Vecchio, M. (2009). An Efficient Lossless Compression Algorithm for Tiny Nodes of Monitoring Wireless Sensor Networks.

*The Computer Journal*, *52*(8), 969-987. doi:10.1093/comjnl/bxp035

Marcelloni, Francesco, & Vecchio, M. (2010). Enabling energy-efficient and lossy-aware data compression in wireless sensor networks by multi-objective evolutionary optimization. *Information Sciences*, *180*(10), 1924-1941. Elsevier Inc. doi:10.1016/j.ins.2010.01.027

Pennebaker, W. B., & Mitchell, J. L. (1992). *JPEG Still Image Data Compression Standard*. Norwell, MA, USA: Kluwer Academic Publishers.

Pradhan, S. S., Kusuma, J., & Ramchandran, K. (2002). Distributed compression in a dense microsensor network. *IEEE Signal Processing Magazine*, *19*(2), 51-60. doi:10.1109/79.985684

Predd, J. B., Kulkarni, S. R., & Poor, H. V. (2006). Distributed Learning in Wireless Sensor Networks. *IEEE Signal Processing Magazine*, *23*(4), 56-69.

Sadler, C. M., & Martonosi, M. (2006). Data compression algorithms for energy-constrained devices in delay tolerant networks. *Proceedings of the 4th international conference on Embedded*

*networked sensor systems - SenSys '06*, 265. New York, New York, USA: ACM Press. doi:10.1145/1182807.1182834

Schoellhammer, T., Greenstein, B., Osterweil, E., Wimbrow, M., & Estrin, D. (2004). Lightweight temporal compression of microclimate datasets [wireless sensor networks]. *29th Annual IEEE International Conference on Local Computer Networks*, 516-524. IEEE (Comput. Soc.). doi:10.1109/LCN.2004.72

Sensirion homepage. (2011). Retrieved March 8, 2011, from www.sensirion.com

SensorScope deployments homepage. (2011). Retrieved March 8, 2011, from http://sensorscope.epfl.ch/index.php/Main_Page

SimIt-ARM Simulator. (2011). Retrieved March 8, 2011, from http://simit-arm.sourceforge.net/

Tharini, C., & Ranjan, P. V. (2009). Design of Modified Adaptive Huffman Data Compression Algorithm for Wireless Sensor Network. *Journal of Computer Science*, *5*(6), 466-470.

TinyNode homepage. (2011). Retrieved March 8, 2011, from http://www.tinynode.com

Villas, L. a., Boukerche, A., Guidoni, D. L., de Oliveira, H. a. B. F., de Araujo, R. B., & Loureiro, A. a. F. (2012). An energy-aware spatio-temporal correlation mechanism to perform efficient data collection in wireless sensor networks. *Computer Communications*, (April). Elsevier B.V. doi:10.1016/j.comcom.2012.04.007

Welch, T. A. (1984). A Technique for High-Performance Data Compression. *Computer*, *17*(6), 8-19.

Wu, M., Tan, L., & Xiong, N. (2016). Data prediction , compression , and recovery in clustered wireless sensor networks for environmental monitoring applications. *Information Sciences*, *329*, 800-818. Elsevier Inc. doi:10.1016/j.ins.2015.10.004

Xiao, J.-jun, Cui, S., Luo, Z.-quan, & Goldsmith, A. J. (2006). Power Scheduling of Universal Decentralized Estimation in Sensor Networks. *IEEE Transactions on Signal Processing*, *54*(2), 413-422.