On the Performance of Resource-aware Compression Techniques for Vital Signs Data in Wireless Body Sensor Networks

Joseph Azar
FEMTO-ST Institute/CNRS
Univ. Bourgogne Franche-Comté
Belfort, France
Email: joseph.azar@univ-fcomte.fr

Abdallah Makhoul
FEMTO-ST Institute/CNRS
Univ. Bourgogne Franche-Comté
Belfort, France
Email: abdallah.makhoul@univ-fcomte.fr

Rony Darazi
TICKET Lab
Antonine University
Hadat-Baabda, Lebanon
Email: rony.darazi@ua.edu.lb

Jacques Demerjian
LARIFA-EDST
Faculty of Sciences, Lebanese University
Fanar, Lebanon
Email: jacques.demerjian@ul.edu.lb

Raphaël Couturier
FEMTO-ST Institute/CNRS
Univ. Bourgogne Franche-Comté
Belfort, France
Email: raphael.couturier@univ-fcomte.fr

***Abstract*—Wireless Body Sensor Networks open up tremendous important applications such as consistent monitoring of a patient's vital signs. One of the main challenges that a Wireless Body Sensor Network faces is the transmission of the collected vital signs measurements. Data transmission is considered to be the greatest consumer of energy in a sensor node. Multiple data compression techniques have been proposed in the literature to reduce the size of the collected data. Thus, the transmission energy consumption. In this paper, we compare the performance of three resource-aware data compression techniques that are proposed in the literature and showed good results: Lightweight Temporal Compression, Differential Pulse Code Modulation and Discrete Wavelet Transform lifting scheme. Then, we propose to adapt the lossy Lightweight Temporal Compression algorithm and combine it with the lossless Differential Pulse Code Modulation algorithm in order to achieve a higher level of compression and reduce the data reconstruction error rate. To evaluate our approach, multiple series of simulation has been done on vital signs data. The results showed that our proposed compression scheme achieved a reduction by up to 95%, and reduced the transmission energy consumption by up to 5 times.**

Keywords—Wireless Body Sensor Network, data compression, energy efficiency, data reduction.

I. Introduction

Recently, there is a great demand for Wireless Body Sensor Networks (WBSN) in many fields, such as medical care, sports, entertainment, and military [1] [2]. One of the important applications of a WBSN is distant patients monitoring. It consists of attaching sensor nodes on a patient's body in order to measure its physiological parameters. The attached sensor nodes, consisting of a battery, microcontroller (MCU), transceiver, memory, and sensor unit, transmit periodically the collected measurements to a coordinator, which is commonly a smartphone or portable device, which in turn forwards the received data to the base station [3].

One of the main challenges that wireless sensor networks and specially WBSNs face in patients monitoring applications is the energy consumption of sensor nodes due to data transmission [4] [5] [6]. In this paper, we target the transmission of vital signs data such as heart rate, respiration rate, blood pressure, blood oxygen saturation, and blood temperature. These vital signs are usually calculated based on raw sensed signals, like the ElectroCardioGram (ECG) signal and the PhotoPlethysmoGram (PPG) signal. In order to increase the lifetime of a WBSN, data compression algorithms can be implemented on sensor nodes to compress the collected vital signs measurements prior to transmission. Thus, reducing the transceiver unit energy consumption that is considered to be the greatest consumer of energy [7].

The compression algorithm implemented on sensor nodes must have a high compression ratio in order to reduce the number of transmitted bits and increase the percentage of energy saving. In like manner, it must have a low computational time/memory complexity in a way that the energy saved from transmitting the compressed data must be greater than the energy consumed by performing additional computation and processing. Many resource-aware compression techniques have been used and developed for data reduction in the context of Wireless Sensor Networks (WSNs) and WBSNs. A simple lossy temporal compression algorithm called Lightweight Temporal Compression (LTC) has been developed in [8] for the compression of microclimate data. The authors showed that the LTC is suitable for low power devices and it performs comparably to the Lempel-Ziv-Welch (LZW) and wavelet compression, comsumes little CPU, and requires very little storage.

In [9], the authors proposed the simple delta encoding algorithm, known as Differential Pulse Code Modulation (DPCM) for accelerometer data compression in WBSN, and compared it to the Huffman encoding. The results showed that the delta encoding outperformed the Huffman encoding in terms of data reduction, computational complexity, and energy savings. A method referred to as LiftingWise has been proposed in [10]. The LiftingWise method is a modified version of the original Discrete Wavelet Transform (DWT) Lifting Scheme (LS) algorithm in which it can be applied on a set of data with arbitrary length while the original LS is applied on a signal S_n of length 2^n. This method has been used to process the data disseminated from objects deployed in a monitoring environment. It was compared with two other simple compression techniques appropriate for usage in WSNs: The Offset compression and Marcelloni compression [11]. The results had proved the effectiveness of this method in reducing the number of bits of the collected data while taking into consideration the limited resources of sensor nodes.

The objective of this paper is twofold. First, we compare the performance of three simple resource-aware compression algorithms: LTC, DPCM (delta encoding), and DWT LS on numeric vital signs data extracted from raw sensed signals. Second, we propose to adapt the LTC algorithm and combine it with the DPCM in order to achieve further compression and reduce the data reconstruction error rate.

The rest of the paper is organized as follows: Section II discusses the characteristics of vital signs data in WBSN. Section III presents some background information about the LTC, DWT LS, and DPCM algorithms. Section IV explains our proposed adaptive version of the LTC. Section V details the experimental results. Section VI concludes the paper.

II. Characteristics of vital signs data

Vital signs data are collected using WBSN and exhibit particular statistical characteristics. Table I shows the mean $\bar{s}$ and the standard deviation $\sigma_{\bar{s}}$ of the five vital signs samples, in addition to the mean $\bar{d}$ and the standard deviation $\sigma_{\bar{d}}$ of the differences between consecutive samples. As seen in Table I, the differences between consecutive samples are characterized by a low standard deviation. Therefore, these data can be considered as smooth data, which makes the compression techniques presented in the following sections good candidates for vital signs data compression in WBSN.

However, there is always the possibility of noise in the data due to sensor displacement. Per example, we can find measurements having a value of zero or artifacts in the form of unreasonable values such as low heart rates (<30 bpm) or high respiration rates (>40 bpm). These artifacts affect the performance of a compression algorithm and increase the error rate when reconstructing the data (lossy compression). We can notice in Table I that respiration rate and systolic blood pressure datasets contains noise since they have greater values of $\bar{d}$ and $\sigma_{\bar{d}}$ as compared to the other datasets.

TABLE I: Statistical characteristics of vital signs data

	$\bar{s}$	$\sigma_{\bar{s}}$	$\bar{d}$	$\sigma_{\bar{d}}$
HR	64.5	1.7	0.1	0.3
RESP	18.4	11.1	0.8	2.5
SpO2	90.6	4.0	0.0	0.1
ABPsys	111.9	21.2	0.3	2.3
BLOODT	36.0	0.0	0.0	0.0

III. Resource-aware Data Compression Techniques

In this section, we present the following resource-aware data compression techniques: LTC, integer LS based Haar wavelet transform, and DPCM.

A. Lightweight Temporal Compression

The LTC algorithm was developed for the context of habitat monitoring in [8]. It is a simple lossy algorithm, just like the Run Length Encoding (RLE) in the sense that it tries to represent a long sequence of similar data with a single symbol by searching for linear trends. The LTC algorithm is initialized with an error bound e at the beginning. The greater is the error bound, the more is the saving from compression.

Consider a dataset x, and an empty set s. As shown in Figure 1(a), the LTC algorithm is initialized by getting the first data point (t_1, v_1) from x, and storing it into z. Then, the second point (t_2, v_2) is used to initialize the upper and lower limits: $UL(t_2, v_2 + e)$ and $LL(t_2, v_2 - e)$. After that, the $highLine$ connecting UL and z, and the $lowLine$ connecting LL and z are calculated. For each data point (t_j, v_j), where $j > 2$, we transform the point into a vertical segment using the margin e. If the data point belongs to the set of all possible lines, we update UL and LL and recalculate the $lowLine$ and the $highLine$ as seen in Figure 1(b). When a point (t_j, v_j) does not belong to the set of all possible lines, as seen in Figure 1(c), we output z to s, and set z to be the point $(t_{j-1}, (UL + LL)/2)$. Algorithm 1 illustrates this process.

One of the advantages of the LTC algorithm is that it achieves high data compression savings on smooth data. Furthermore, its memory requirements are constant, the amount of work needed to process a new data point is small, and needs $\Theta(n)$ time to examine n points.

B. Integer Lifting Scheme based Haar wavelet transform

The LS is a technique developed by Sweldens in [12] that performs the DWT. The integer version of the LS can be used as a lossless compression method that converts a set of integers to another set of integers and results in an efficient reversible implementation [13].

Compressing data using LS consists of three steps: The first step is split phase that split a sequence of data x_j of length 2^j, where j is a positive integer, into odd and even sets of length 2^{j-1} each. The second step is predict step, in which odd set is predicted from even set and transformed into the differences set denoted by d_{j-1}. The prediction operation results in smaller values that can be represented in fewer bits. Note that the integer LS version of the Haar transform is used in this paper. The Haar transform prediction operation predicts that the odd element will be equal to the even element. The odd element is then replaced by the difference between the predicted value (the even element) and the actual value of the odd element as defined in Equation 1. The third step is update step, in which the even set is transformed into the averages set denoted by s_{j-1}, and each even element in the even set is replaced with the average of the even/odd pair as in Equation 2.

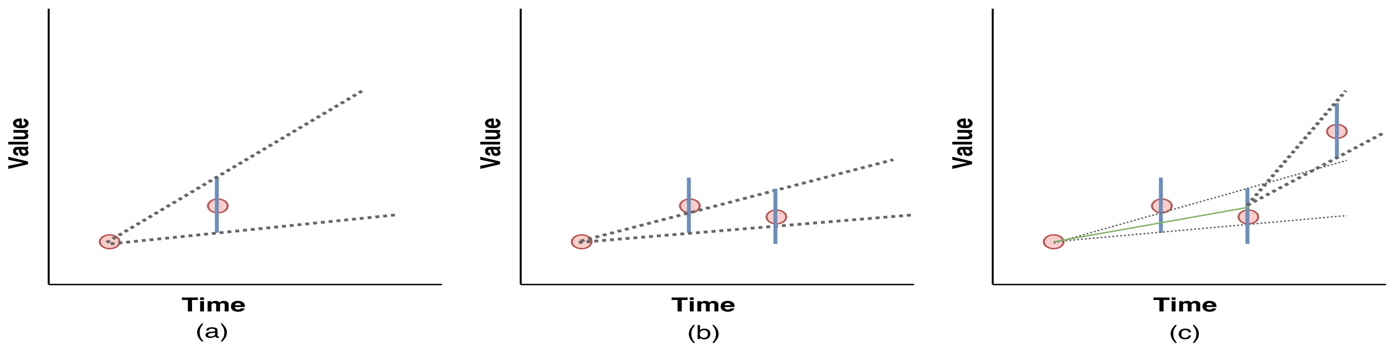

Fig. 1: The algorithm clarified. (a) LTC is initialized with a fixed point, and a point that is transformed into vertical line segment that decides the set of all possible lines. (b) The set of all lines is reduced with each new point. (c) When a point does not belong to the set of all possible lines, the past dataset is topped off and the procedure begins once again

Algorithm 1 LTC algorithm

```
Require: $x, n, e$ //Vector $x$ of length $n$, error bound $e$
Ensure: $s$ // Set of data points
 1: $z \leftarrow (t_1, v_1)$
 2: $UL \leftarrow v_2 + e$
 3: $LL \leftarrow v_2 - e$
 4: $highLine \leftarrow calculateHighline(z, UL)$
 5: $lowLine \leftarrow calculateLowline(z, LL)$
 6: for $j = 3$ to $n$ do
 7:   $ul \leftarrow v_j + e$
 8:   $ll \leftarrow v_j - e$
 9:   if $highline$ below $ll$ or $lowline$ above $ul$ then
10:     $s.add(z)$
11:     $z \leftarrow (t_{j-1}, (UL + LL)/2)$
12:     $UL \leftarrow ul$
13:     $LL \leftarrow ll$
14:     $highLine \leftarrow calculateHighline(z, UL)$
15:     $lowLine \leftarrow calculateLowline(z, LL)$
16:   else
17:     if $UL > ul$ then
18:       $UL \leftarrow ul$
19:     end if
20:     if $LL < ll$ then
21:       $LL \leftarrow ll$
22:     end if
23:     $highLine \leftarrow calculateHighline(z, UL)$
24:     $lowLine \leftarrow calculateLowline(z, LL)$
25:   end if
26: end for
27: $s.add(z)$
28: $s.add((t_n, v_n))$
29: return $s$
```

$$d_{j-1} = x[2n+1]_j - x[2n]_j. \tag{1}$$

$$s_{j-1} = x[2n]_j + \lfloor d_{j-1}/2 \rfloor. \tag{2}$$

The aforementioned operations can be repeated j times on a dataset of length 2^j. The final result is dataset s_j of one average (the mean value of all samples) and j sets of differences [13].

The Haar wavelet is well known for its simplicity and speed of computation. It has a time complexity of $\Theta(n \log n)$ and allows a fully in-place calculation of the forward transform.

C. *Differential Pulse Code Modulation*

The DPCM is a lossless compression method that is based on the fact that neighboring samples are correlated. Correlated values are usually similar, so their differences are small, resulting in compression. DPCM can be applied by keeping the first sample as a reference and changing every next sample with the difference between this sample and the previous one:

$$d(k) = \begin{cases} x(k), & \text{if } k = 0 \\ x(k) - x(k-1), & \text{if } k > 0 \end{cases} \tag{3}$$

The DPCM is very simple. It has a time complexity of $\Theta(n)$ and requires the less number of calculations as compared to LTC and LS algorithms.

IV. Proposed Adaptive Lightweight Temporal Compression

The LTC algorithm presented in the previous section takes a dataset x with an error bound e as input and returns a set s of data points as output. One of its weaknesses is that it suffers when the data are noisy. In the worst case, when no line can be fit between more than two consecutive points, the output of the algorithm may require more bits than that needed to represent the uncompressed dataset. Furthermore, it poorly reconstructs noisy data. In order to minimize the reconstruction error rate, we adapted the LTC in a way that when a point (t_j, v_j) does not belong to the set of all possible lines, as seen in Figure 1(c), we add the points (t_{j-1}, v_{j-1}) and (t_j, v_j) to s instead of only adding the point midway between UL and LL (Algorithm 2, lines 12-15). As a result, the compression ratio of the algorithm will decrease since additional points are added to the output set s. To achieve the trade-off between the compression ratio and the reconstruction error, we combined the adapted lossy LTC algorithm with the lossless DPCM algorithm. Note that in implementation, each data point (i, v) in s consists of the index i of the measurement in the dataset and its value v in order to reconstruct the data at the base station. Consider a vector

x containing vital signs measurements. In the first place, we apply the adapted LTC algorithm on x, which results in a set s of data points. Then, for each data point in s, we add its index into a set I, and its value into a set V. After that, we compress the set of indexes I and the set of values V using the DPCM. This process is defined in Algorithm 2.

This adaptation ensures the reduction of reconstruction error rate and the achievement of additional compression level even if the data are noisy.

Algorithm 2 Adaptive LTC

```
Require: $x, n, e$ //Vector $x$ of length $n$, error bound $e$
Ensure: $s$ // Set of data points
1: $s.add((t_1, v_1))$
2: $z \leftarrow (t_1, v_1)$
3: $UL \leftarrow v_2 + e$
4: $LL \leftarrow v_2 - e$
5: $highLine \leftarrow calculateHighline(z, UL)$
6: $lowLine \leftarrow calculateLowline(z, LL)$
7: for $j = 3$ to $n$ do
8:     $ul \leftarrow v_j + e$
9:     $ll \leftarrow v_j - e$
10:    if $highline$ below $ll$ or $lowline$ above $ul$ then
11:        $z \leftarrow (t_{j-1}, (UL + LL)/2)$
12:        if $(t_{j-1}, v_{j-1})$ not in $s$ then
13:            $s.add((t_{j-1}, v_{j-1}))$
14:        end if
15:        $s.add((t_j, v_j))$
16:        $UL \leftarrow ul$
17:        $LL \leftarrow ll$
18:        $highLine \leftarrow calculateHighline(z, UL)$
19:        $lowLine \leftarrow calculateLowline(z, LL)$
20:    else
21:        if $UL > ul$ then
22:            $UL \leftarrow ul$
23:        end if
24:        if $LL < ll$ then
25:            $LL \leftarrow ll$
26:        end if
27:        $highLine \leftarrow calculateHighline(z, UL)$
28:        $lowLine \leftarrow calculateLowline(z, LL)$
29:    end if
30: end for
31: if $(t_n, v_n)$ not in $s$ then
32:     $s.add((t_n, v_n))$
33: end if
34: $I \leftarrow get\_Indexes(s)$
35: $V \leftarrow get\_Values(s)$
36: $DPCM(I)$
37: $DPCM(V)$
38: return $[I, V]$
```

V. Experimental Results and Analysis

The performance of the aforementioned compression techniques has been evaluated using a custom Java-based simulator. In the followings, we discuss the results obtained from applying the LS, DPCM, LTC, and our adaptive version of LTC (LTC*) on datasets collected from the online Multi-parameter Intelligent Monitoring in Intensive Care (MIMIC) I database "Numerics" of PhysioNet [14]. Note that the error bound for the LTC and LTC* algorithms was set to 1 in our simulation. We tested the compression techniques on the following vital signs: heart rate (HR), respiration rate (RESP), blood oxygen saturation (SpO2), systolic blood pressure (ABPsys), and blood temperature (BLOODT). Furthermore, we suppose in our simulation that each vital sign is observed by one sensor node.

Multiple simulations have been run on each vital sign measurements over 43 periods. We consider that the collected measurements are accumulated in the memory of the sensor node, and transmitted after each period $p = 60sec$.

Three metrics are discussed in the followings: data and communication compression, loss of information, and transmission energy consumption.

A. Data and communication compression

The performance of the compression techniques is computed using the Compression Ratio (CR) as denoted in Equation 4:

$$CR(\%) = 100 \times \left(1 - \frac{comp\_size}{uncomp\_size}\right), \tag{4}$$

where $comp\_size$ and $uncomp\_size$ are the sizes in bits of the compressed and uncompressed bitstreams respectively.

Table II presents the compression ratios obtained by the four compression algorithms. The results show that the performance of the LTC algorithm strongly depends on the statistical characteristics of the data. Therefore, it achieved a CR by around 91% in the case of smooth data (HR, SpO2, and BLOODT), and by around 63% when the data are less smooth (RESP). On the other hand, the DPCM algorithm outperformed the LS algorithm on all vital signs measurements and achieved compression ratios that vary between 35% and 50%. As can be seen, the highest compression ratios were achieved by the proposed LTC* algorithm. Note that the performance of both LTC* and LTC algorithms was close, as the performance of the LTC* was a little better.

TABLE II: Compression ratios (%) obtained by the four compression algorithms on the five vital signs measurements

	LTC	**LS**	**DPCM**	**LTC***
HR	92.3	45.0	50.3	95.1
RESP	63.0	38.6	43.6	66.5
SpO2	90.3	30.3	34.9	94.1
ABPsys	87.5	43.0	47.0	92.4
BLOODT	91.9	44.0	47.7	94.4

The reduction in the size of the data (data compression) has a direct reflection on the communication cost in WBSN. Hence, reducing the packet length ultimately reduces the radio on-time of the transceivers (communication compression) [15]. Table III illustrates the number of packet transmissions achieved by each algorithm after 43 periods. Considering that each packet can contain at most 25 bytes of payload, we can see that the LTC and LTC* algorithms achieved the greater packet transmissions reduction. They performed the same number of packet transmissions in the case of HR, SpO2, and BLOODT datasets, as the LTC* further reduced this number in the case of RESP, and ABPsys datasets (noisy datasets).

TABLE III: Number of packet transmissions achieved by each algorithm after 43 periods

	Uncompressed	LTC	LS	DPCM	LTC*
HR	129	43	86	86	43
RESP	129	64	87	86	61
SpO2	129	43	86	86	43
ABPsys	129	47	87	87	44
BLOODT	129	43	86	86	43

B. Lossy compression vs loss of information

Since the LTC compression algorithm is lossy, we need to assess how much the reconstructed data at the base-station differ from the uncompressed (original) data. To do so, we calculate the Root Mean Squared Error (RMSE) as denoted in Equation 5:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(A_i - B_i\right)^2}, \tag{5}$$

Where A_i is the original measurement, B_i is the reconstructed measurement, and n is the number of measurements. Furthermore, we calculate the Scores Root Mean Squared Error (S-RMSE) defined by Equation 6:

$$S - RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(S(A_i) - S(B_i)\right)^2}, \tag{6}$$

Where S is a function that assigns a score to each vital sign measurement. The assigned score is based on the scoring template for the National Early Warning Score (NEWS) used in U.K [16]. A regular healthy range is defined for each vital sign, and the allocated score reflects how much the measured sample differs from this range. The more is a vital sign measurement outside the defined normal range, the higher is the assigned score, which indicates its severity level. What is more important to us is to not miss the information carried by a measurement. In other words, the score assigned to each measurement based on the NEWS system reflects the critical level of the vital sign, which must remain intact when reconstructing the data.

TABLE IV: RMSE and S-RMSE between the original dataset and the reconstructed dataset

	LTC		LTC*	
	RMSE	**S-RMSE**	**RMSE**	**S-RMSE**
HR	0.713	0.0	0.701	0.0
RESP	1.778	0.428	0.643	0.227
SpO2	0.348	0.02	0.34	0.018
ABPsys	1.206	0.06	0.688	0.025
BLOODT	0.0	0.0	0.0	0.0

Table IV shows the RMSE and S-RMSE between the original data and the reconstructed data after applying the LTC and LTC* algorithms. We can see that the LTC algorithm yields high RMSE when there is noise in data (RESP, ABPsys). On the other hand, the proposed LTC* yields a RMSE that is close to 0.5, and a S-RMSE that is close to 0.1 on all vital signs data. Here shows the importance of our proposed LTC* algorithm, which achieved an insignificant loss of information. Figure 2 shows the reconstruction of 1 period RESP data using LTC and LTC*. It is clear that the LTC* reconstructed signal fits the original signal more than the LTC reconstructed signal.

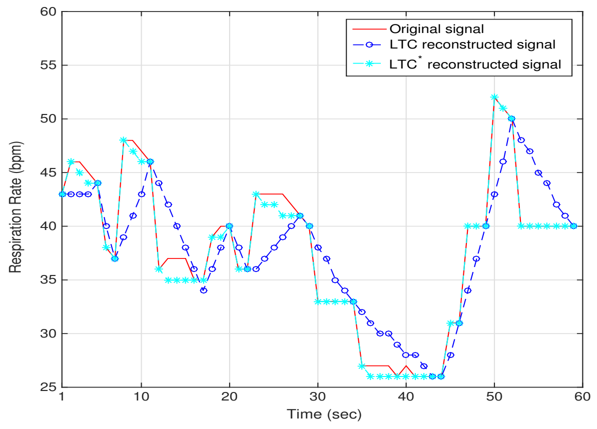

Fig. 2: Reconstruction of 1 period RESP data using LTC and LTC*

C. Transmission Energy Consumption

In this section, we estimate the transmission energy consumption of the sensor nodes after each period when applying the four compression algorithms. To do so, we use the radio model proposed in [17]. The equation used to calculate transmission costs is defined by Equation 7.

$$E_{TX}(k, d) = E_{elec} \times k + \beta_{amp} \times k \times d^2. \tag{7}$$

Where k is the sizes in bits of the bitstreams to be transmitted, and d is the distance in meters between the sensor node and the coordinator. Note that in this model, the radio dissipates E_{elec} = 50nJ/bit to run the transmitter circuitry and β_{amp} = 100pJ/bit/m2 for the transmitter amplifier. In our simulation, we consider that the distance between a sensor and the coordinator is $d = 1m$.

Figures 3 and 4 illustrate the variation of the transmission energy consumption of RESP sensor node where the data are noisy and HR sensor node where the data are very smooth. We can notice in Figure 3 that the performance of the LTC algorithm is not stable over time. So that after some periods, the transmission energy consumed by the LTC algorithm is greater than the one consumed by the DPCM and the DWT LS. On the other hand, the proposed LTC* algorithm achieved the lowest transmission energy consumption among all methods. In the case of smooth data as shown in Figure 4, the LTC and LTC* algorithms achieved a better reduction in the energy consumption compared to the DWT LS and DPCM algorithms.

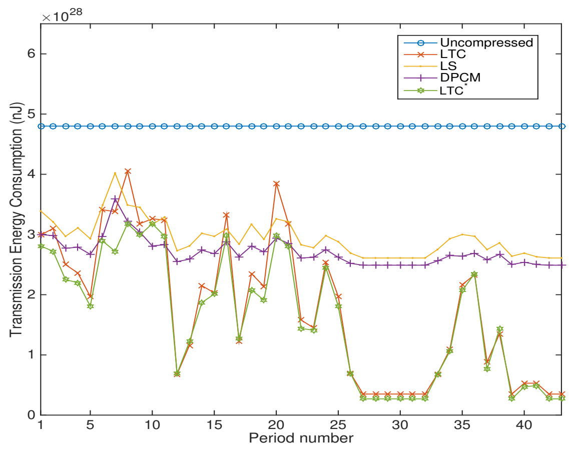

Fig. 3: Transmission energy consumption of RESP sensor node over 40 periods

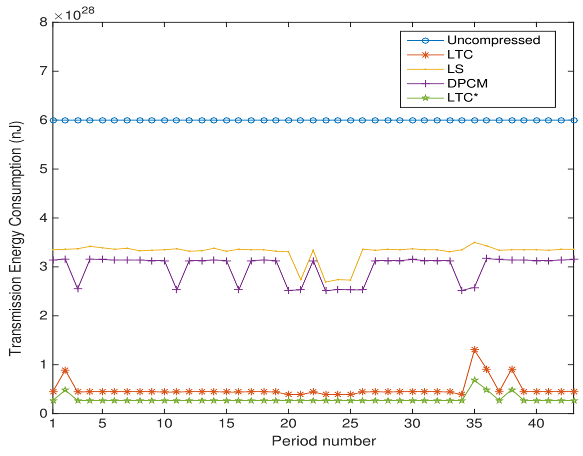

Fig. 4: Transmission energy consumption of HR sensor node over 40 periods

VI. Conclusion

Data compression can be considered as a direct way to reduce the energy consumption due to wireless transmission. In this paper, we compared the performance of three resource-aware compression techniques: LTC, DWT lifting scheme, and DPCM on vital signs measurements calculated from raw sensed signals. The results showed that the lossy LTC achieved the higher compression ratio on smooth data with insignificant loss of information. However, the data reconstruction error increased when the data became noisy, which leads to missing some important features. On the other hand, the lossless DWT lifting scheme and DPCM algorithms yielded a constant performance and achieved a data reduction by between 35% and 50%, with a little preference for the DPCM over the DWT lifting scheme. In order to achieve a higher compression with a minimal loss of information, we proposed to adapt the LTC algorithm and combine it with the lossless DPCM algorithm. The adapted LTC achieved the higher compression ratio among the other algorithms on smooth and noisy data and reduced the data reconstruction error rate. Thus, increasing the lifetime of the WBSN.

Acknowledgment

This work is partially funded with support from the Labex ACTION program (contract ANR-11-LABX-01-01), the National Council for Scientific Research in Lebanon CNRS-L, the Hubert Curien CEDRE programme, and the Lebanese University Research Program (Number: 4/6132).

References

[1] X. Lai, Q. Liu, X. Wei, W. Wang, G. Zhou, and G. Han, “A survey of body sensor networks,” *Sensors*, vol. 13, pp. 5406 – 5447, 2013.

[2] N. Boudargham, J. B. Abdo, J. Demerjian, and C. Guyeux, “Exhaustive study on medical sensors,” in *SENSORCOMM 2017, The Eleventh International Conference on Sensor Technologies and Applications*. IARIA, 2017.

[3] C. Habib, A. Makhoul, R. Darazi, and C. Salim, “Self-adaptive data collection and fusion for health monitoring based on body sensor networks,” *IEEE Trans. Industrial Informatics*, vol. 12, no. 6, pp. 2342–2352, 2016.

[4] A. Makhoul, H. Harb, and D. Laiymani, “Residual energy-based adaptive data collection approach for periodic sensor networks,” *Ad Hoc Networks*, vol. 35, pp. 149–160, 2015.

[5] A. Makhoul, D. Laiymani, H. Harb, and J. M. Bahi, “An adaptive scheme for data collection and aggregation in periodic sensor networks,” *IJSNet*, vol. 18, no. 1/2, pp. 62–74, 2015.

[6] D. Laiymani and A. Makhoul, “Adaptive data collection approach for periodic sensor networks,” *2013 9th International Wireless Communications and Mobile Computing Conference, IWCMC 2013, Sardinia, Italy, July 1-5, 2013*, pp. 1448–1453, 2013.

[7] G. Anastasi, M. Conti, M. D. Francesco, and A. Passarella, “Energy conservation in wireless sensor networks: A survey,” *Ad Hoc Networks*, vol. 7, pp. 537 – 568, 2009.

[8] T. Schoellhammer, B. Greenstein, E. Osterweil, M. Wimbrow, and D. Estrin, “Lightweight temporal compression of microclimate datasets [wireless sensor networks],” in *29th Annual IEEE International Conference on Local Computer Networks*. IEEE, 2004.

[9] S. Arrabi and J. Lach, “Adaptive lossless compression in wireless body sensor networks,” in *Proceedings of the Fourth International Conference on Body Area Networks*, ser. BodyNets ’09, 2009.

[10] E. Aboelela, “Liftingwise: A lifting-based efficient data processing technique in wireless sensor networks,” *Sensors*, vol. 14, pp. 14 567–14 585, 2014.

[11] F. Marcelloni and M. Vecchio, “An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks,” *The Computer Journal*, vol. 52, pp. 969–987, 2009.

[12] W. Sweldens, “The lifting scheme: A custom-design construction of biorthogonal wavelets,” *Applied and Computational Harmonic Analysis*, vol. 3, pp. 186 – 200, 1996.

[13] A.Jensen and A. Cour-Harbo, *Ripples in Mathematics: The Discrete Wavelet Transform*. Springer, 2001.

[14] A. L. G. *et al.*, “Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals,” *Circulation*, 2000.

[15] M. A. Razzaque, C. Bleakley, and S. Dobson, “Compression in wireless sensor networks: A survey and comparative evaluation,” *ACM Trans. Sen. Netw.*, vol. 10, no. 1, pp. 5:1–5:44, Dec. 2013.

[16] “National early warning score (news), royal college of physicians, london, u.k., may 2015.” https://www.rcplondon.ac.uk/projects/outputs/national-early-warning-score-news, accessed: 2017-12-07.

[17] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*. IEEE, 2000.