Data Compression for Energy Efficient IoT Solutions

Biljana Stojkoska

Telfor 2017

Data Compression for Energy Efficient IoT Solutions

Biljana Risteska Stojkoska, *Member, IEEE*, and Zoran Nikolovski

***Abstract* — Energy is the most important resource in state-of-the-art Internet of Things solutions. There are a lot of concepts and techniques dedicated to save energy, mainly focused to reduce transmission, since the energy used for preprocessing (encoding) is incomparable smaller than energy used for broadcasting. If applications do not require real-time measurements, data compression is one solution to energy saving problem. The goal of this paper was to develop new coding scheme for delta compression, that can be used for efficient data compression of temporally correlated data, such as temperature measurements coming from different smart devices. We proved that our coding scheme can achieve up to 85% energy saving. Compared to other coding techniques, our scheme has greater compression ratio and lower memory requirements.**

***Keywords* — data reduction, delta compression, internet of things.**

I. Introduction

The combination of information technologies and advanced communication and sensing systems, creates a variety of new potential applications under the umbrella of Internet of Things (IoT). IoT represents a worldwide network of uniquely addressable interconnected smart objects or smart devices [1]. In IoT, smart objects have its own processor, memory and module for communication (usually wireless). The power supply for the devices can be provided from the traditional electric grid, but also, many devices come with battery or distributed renewable power supply, such as solar panel.

Energy saving for wireless devices has been very challenging problem for decades. In IoT context, energy saving is not only important from power supply perspective, but also from network perspective. Namely, new applications are constantly fed with raw data coming from many sensors. Therefore, reducing network traffic is very important in order to avoid saturation and to achieve many devices to work cooperatively within the same data hub [2].

Data compression is suitable approach in applications were data are not needed in real time. Other limitation is the fact that implementing complex algorithm for energy saving is constrained by the limited memory, capacity and processor power of the smart devices. Therefore, in this paper, we aim to develop a lightweight algorithm for data compression that is based on delta coding scheme.

Similar approaches can be found in the literature. For example, authors in [3] developed lossless compression technique for temperature measurements from Telos motes with 14-bit analog-to-digital converter (ADC) and achieved compression ratios of 66.99% and 67.33%. Authors in [4] investigate a dynamic lossy compression method on smart meters data and have a compression gain of up to 65%. In [5] a lightweight adaptive lossless data compression scheme based on two code options performs compression performance up to 74.02% on humidity and temperature measurements from a 14-bit ADC. We investigated our algorithm on temperature measurements obtained in indoor environment from MicaZ [6] devices with 12-bit ADC, and proved that our coding scheme can achieve up to 85% energy saving. Since the evaluation for different coding scheme is evaluated on different dataset, it is very difficult to compare compression techniques. Therefore, we developed a LZW [7] based scheme to show that our approach performs better. We also want to highlight that our scheme is not the most optimal regarding compression ratio, but is very lightweight, computationally cheap and suitable for IoT devices with limited resources.

The rest of this paper is organized as follows. The next section provides statistical processing on temperature sensor measurements. Section III provides the design of our coding scheme. Section IV examines the experimental results. Finally, we conclude this paper in section V.

II. Statistical processing of temperature measurements

With statistical processing of a big amount of data, we can get more accurate stochastic model of the process. For this purpose, we collected "raw" temperature measurements from MICAz Crossbow [6] smart devices, also known as nodes or motes, identified with a unique identification number "nodeid". MICAz use very efficient, open-source TinyOs operating system, developed by UC Berkeley [8], optimized to work on devices with limited resources. TinyOs is written in NesC (Network embedded systems C) programming languages. For our purpose, we use the data gathered using MOTE-VIEW Application by Crossbow Technology Inc. [6]. We use temperature measurements obtained from the MICAz nodes for a period of one week,

This project was financially supported by the Faculty of Computer Science and Engineering.

Biljana Risteska Stojkoska is with the Faculty of Computer Science and Engineering, University Ss. Cyril and Methodius, Rugjer Boshkovikj 16, P.O. Box 393, 1000 Skopje, Macedonia (e-mail: biljana.stojkoska@finki.ukim.mk). Zoran Nikolovski is with the Faculty of Electrical Engineering and Information Technologies, University Ss. Cyril and Methodius, Rugjer Boshkovik bb, P.O. Box 574, 1000 Skopje, Macedonia.

978-1-5386-3073-0/17/$31.00 ©2017 IEEE

sampled on 3 minutes. Each temperature value is represented as a 10 bits code produced directly from the output of the smart device ADC. The ADC output is saved in SD internal memory, and then it is send to the base station. To store this data, 2B are allocated. By using only 10 bits long ADC output, we have redundancy of 6 bits for each temperature data.

Temperature process is a slow changing process and there is a temporal correlation, i.e. temperature value of the next sampling moment depends on the temperature value of previous sampling moment. If the ADC output is 10 bits, up to 1024 states (symbols) can be generated. Each symbol from the ADC output is statistically depended, so the next state depends of the previous states and not only from the probability of the symbol. If calculating Shannon entropy of this source is possible, then with proper coding of these symbols, it would be possible to decrease average length of the code word and to achieve efficient data transfer [9].

Let assume that the frequency of the ADC output samples is f_s. Then the sample period is $T=1/f_s$, so the time difference between the two sample moments is T. Let $u(nT)$, $u(nT) \in S$ stands for the ADC output's state in discrete moments of time, where $n=0\ldots\infty$. The state of the ADC output in random moment kT is $u(kT)$. The first difference of the ADC output is defined with the equation (1) and (2).

$$\Delta(kT) = u[kT] - u[(k\text{-}1)T] \quad (1)$$

$$\Delta(k) = u[k] - u[(k\text{-}1)] \quad (2)$$

Using statistic processing on this data we achieve to find the distribution of the probability of Δ, for each sensor node. The probability of occurrence of Δ will be shown trough histograms, as a Gaussian approximation for each histogram.

As can be seen in Fig. 1 and Fig.2, the distribution of the probability of Δ follows the Gaussian law for distribution of probability. If we make an approximation that the random variable Δ has a Gaussian distribution and if we analyze the average value and the variance of this distribution, we will see that the most probable values that Δ can get are 0, -1 and 1. Using this we can do statistic encoding on the values that Δ can get. According to the encoding theory, those symbols that have higher probability of occurrence should be encoded with less bits, and the symbols that have lower probability of occurrence should be encoded using more bits.

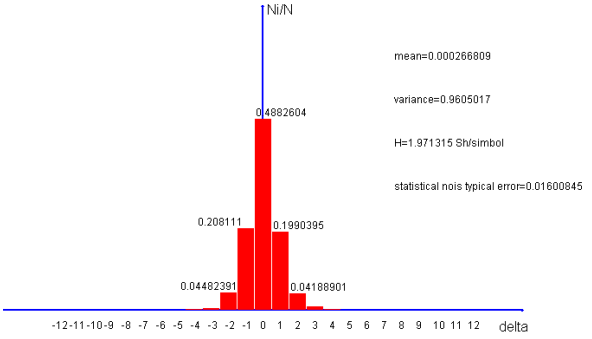

Fig. 1. Histogram of distribution of the probability of Δ for the node 37.

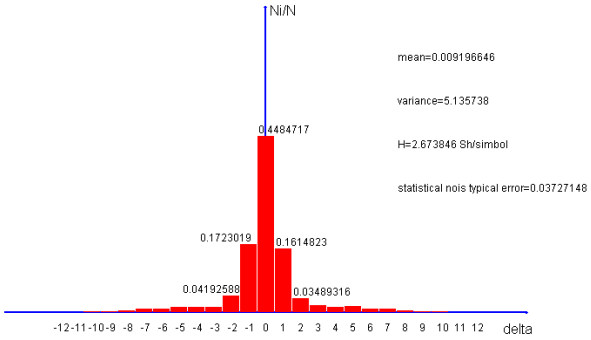

Fig. 2. Histogram of distribution of the probability of Δ for the node 25.

III. DELTA ENCODING SCHEME

There are many optimal code schemes that can be used to encode Δ [10][11]. One example of that is the Huffman code, which is characterized with variable length of the code words and it is a prefix code (no code word is a prefix to other code word). Huffman code can be used for IoT data compression, but in reality, it imposes a few problems. Since the distribution of Δ is gained using statistic methods, this distribution does not give us a real view of the process. According to this, optimality of the code will be questionable. In real time, the variance and the mean value of this process are variable, so will the probability of occurrence of the symbols. Another big problem would appear from the number of symbols that Δ theoretically can accept. Since we cannot certainly depend on the statistic analyze, we cannot be certain that Δ difference will not accept some higher value. The Huffman code is based on building a code tree for the symbols (or a lookup table), that takes a lot of memory and a lot of processing resources for locating the symbols. Due to these limitations, if there are many symbols in the lookup table, Huffman scheme would be hard to implement and also inefficient, especially for smart IoT devices with limited processing capabilities. Therefore, in this paper we propose an encoding method that is avoiding this implementing problem and the problem of eventually big temperature difference at the two sample moments. This scheme can solve the problem of randomly high value for Δ, but in this case, we would not use compression and we will send a redounded information. But it is approved because of the very small probability of a big value for Δ. This code scheme is relatively easy to implement and the calculations during the encoding are on a minimum rate. The decoding algorithm is also very simple.

In our coding scheme, encoding data are not sent synchronically, but when the memory space given for inscribing the encoding words is fulfilled. This is because Δ difference is coded with variable length codes. The difference between the present and the previous value of the temperature (Δ difference) is encoded with this coding scheme. The choice of the minimum difference, which this code scheme can decode, depends of the threshold of sensibility of the application, since some applications require larger precision of the measured values. The threshold of sensibility is defined as the smaller value of the

difference between two measurements for which the application is interested in. In other words, the changes of the value of the physical magnitude, which are smaller from this threshold of the application, don't have any influence in the functionality of the application. The threshold of sensibility of the application measured with units of physical magnitude (for temperature °C, °F, K) we will mark with δ, and the threshold of sensibility of the application in raw format we will mark with θ. For example, if its measure of temperature with measuring rectifier that have linear characteristic and measurable interval from 0 – 100 °C, and the exit of this rectifier is digitalized with 10 bits ADC, the threshold of sensibility of the rectifier is about 0.1 °C per least significant bit (LSB), i.e. 0.1 °C/LSB. If the application has a threshold of sensibility of $\delta = 0.5$ °C, then $\theta = 5$. If we want to reach the maximum punctuality during encoding, then θ should equalize with the threshold of sensibility of the measuring rectifier ($\theta = 1$(LSB)).

A. *Data organization of Δ frame*

The encoding data are sent organized in structure called frame. The frame has one field that contains the primary value of the temperature and other field (memory buffer) in which the codes for the value of Δ in every sampling moment are inscribed in sequence (Fig. 3). The memory buffer in our case is with 32B length. We named this buffer as FIFO register with 256 bites length. The inscribing of the code of Δ is fulfilled from right, so the register is shifted to the left for as many bit positions as the length of the code word (Fig. 3).

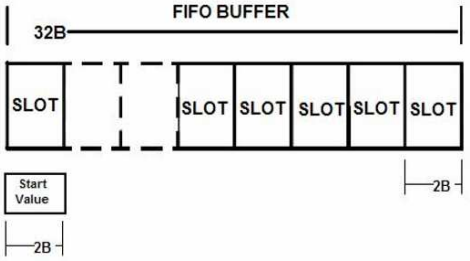

Fig. 3. Data organization of Δ frame

B. *Δ code scheme*

Code words, which are used for encoding Δ, are reached with combination of four basic 2bit codes. Three of this bit code words are used for encoding of the most probable values of Δ, 0, $-\theta$ and θ. The four 2bit code word is used as a prefix and suffix for reaching code words for the rest values of Δ. Table 1 shows the binary values of these four codes and their marks.

Code words for Δ, when the value of (abs(Δ)DIVθ) ≥ 2), are reached when as a prefix of the code word is put the code FLAG, and then, depending of the sign of Δ are put as much basic codes as amounts ((abs (Δ) DIV θ) – 1) (if Δ < 0 the code DOWNF is used, if Δ > 0 the code UPF is used). At the end as a suffix, the code FLAG is used. Table 2 shows the codes of few of the values Δ when (abs(Δ)DIVθ) ≥ 2).

From Table 2 we can see that the length of code words grows when we increase the value of abs(Δ). For values of abs(Δ) > 7θ we do not have compression in the data anymore. But with the previous implemented statistical analyses we saw that the probability abs(Δ) > 7θ (in this statistic data processing $\theta = 1$) is very small. With the usage of this delta code scheme there is no possibility to occur error in decoding, because it can decode any value of Δ, but the cost will be paid in transfer of the redundant data i.e. the length of the code word will be bigger than the length of the authentic symbol.

TABLE 1: BASIC CODE IN DELTA CODE SCHEME.

Binary code	***Mark***	Δ
00	EQU	0
01	FLAG	/
10	UP	
11	DOWN	-
1	UPF	/
0	DOWNF	/

TABLE 2: ENCODING OF Δ WHEN abs(Δ)DIVθ) ≥ 2.

Binary code	***Mark***	Δ
10110	FLAG UPF FLAG	2θ
10010	FLAG DOWNF FLAG	-2θ
101110	FLAG UPF UPF FLAG	3θ
100010	FLAG DOWNF DOWNF FLAG	-3θ
1011110	FLAG UPF UPF UPF FLAG	4θ
1000010	FLAG DOWNF DOWNF DOWNF FLAG	-4θ

IV. EXPERIMENTAL RESULTS

In order to explore the efficiency of compression based on Delta coding, the same database was used as in the case of statistic processing of temperature measurements. For our smart devices based on TinyOs (TOS) platform, data are sent organized in packets. The packet, beside the application data, contains and so called overhead data (Header and CRC) (Fig. 4). The magnitude of the Δ frame is chosen to be the same with the maximum permitted magnitude of the application data which can be sent in one packet, which contributes to decrease the sent overhead. Fig. 4 shows the data structure of the TOS packet.

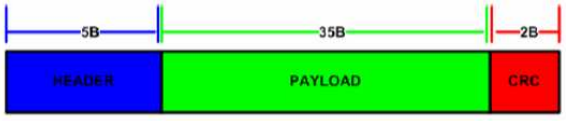

Fig. 4. Data structure of TOS packet

The goal of this analysis was to obtain the number of TOS packets sent by the smart device with and without coding, respectively. Additionally, the number of TOS packets can be used to calculate the energy saving achieved by applying delta coding scheme.

The compression ratio per frame (CRPF) for Δ algorithm is defined as relation of the number of encoded values in one frame (EVPF) and the number of encoded values which can be put in the memory space with length of one frame (NVPF).

$$CRPF = \frac{EVPF}{NVPF} \quad (3)$$

The compression ratio (CR) for N sent frames can be obtained using (4).

$$\mathrm{CR(N)} = \frac{1}{\mathrm{N}} \sum_{i=1}^{N} CRPF_i \quad (4)$$

We examine two nodes (nodeid=25 and nodeid=37). Number of temperature data values was 3861 for nodeid=25 and 4016 for nodeid=37. The results from our analysis are given in Fig. 5 and Fig. 6, for two different values for θ.

If E_p is the energy that the node uses for sending one packet, N_o number of packets send without coding, N_e is the number of packets send with coding, than data communication energy savings is calculated with (5).

$$E_s[\%] = (1 - \frac{N_e E_p}{N_o E_p})100 = (1 - \frac{N_e}{N_o})100 \quad (5)$$

Data communication energy savings (in %) when Δ encoding is used is given in Fig. 7.

Different algorithms can be used for data compression (LZ77, LZ78, LZW, Gzip, etc.). The aim of these algorithms is to decrease the magnitude of files with encoding. At these algorithms the efficiency does not depend just from the compression ratio that that algorithm reaches, also depends from the complexity of that algorithm. For example, if we use complex algorithm, we will increase the power the node spends for complex calculations. Therefore, we performed a comparison between our Δ coding scheme and well-known LZW based coding from the literature, i.e. SLZW [7], a modification of LZW optimized for senor measurements. We considered 12 bits long code words, and 750B and 1500B memory space. The results are presented in Fig. 8. The memory required for both LZW algorithms was more than 3KB, while the memory required for our coding scheme was only 50B.

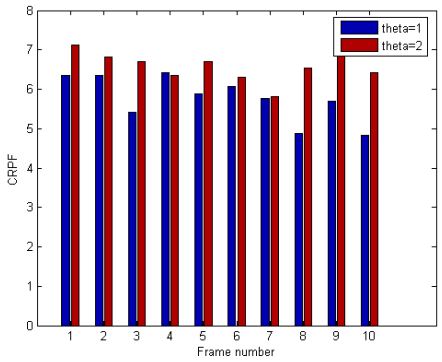

Fig. 5. Compression ratios per frame for the node with nodeid=25.

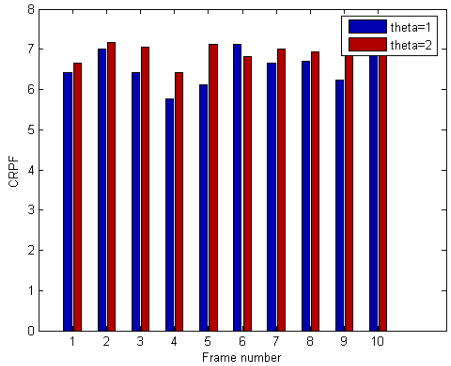

Fig. 6. Compression ratios per frame for the node with nodeid=37.

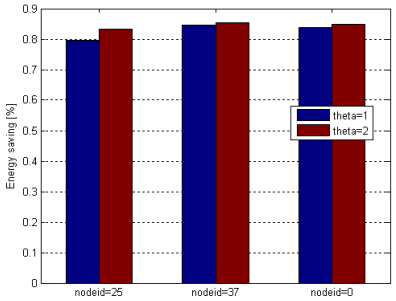

Fig. 7. Data communication energy savings (in %) when Δ encoding is used.

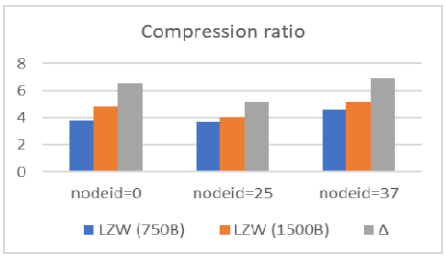

Fig. 8. Comparison of LZW and our coding scheme.

V. Conclusion

In this paper we investigated Delta coding as a technique for energy saving in IoT environment. We showed that this technique can achieve up to 85% energy saving if applied on data that are temporally corelated. Therefore, for applications where the sensor measurements are not needed in real-time, using Delta coding is highly recommendable.

References

[1] Stojkoska, Biljana L. Risteska, and Kire V. Trivodaliev. "A review of Internet of Things for smart home: Challenges and solutions." *Journal of Cleaner Production* 140 (2017): 1454-1464.

[2] Stojkoska, Biljana Risteska, Dimitar Solev, and Danco Davcev. "Variable step size LMS Algorithm for Data Prediction in wireless sensor networks." *Sensors & Transducers* 14, no. 2 (2012): 111.

[3] Marcelloni, Francesco, and Massimo Vecchio. "A simple algorithm for data compression in wireless sensor networks." IEEE communications letters 12, no. 6 (2008).

[4] Ukil, Arijit, Soma Bandyopadhyay, and Arpan Pal. "IoT data compression: Sensor-agnostic approach." In *Data Compression Conference (DCC), 2015*, pp. 303-312. IEEE, 2015.

[5] Kolo, Jonathan Gana, S. Anandan Shanmugam, David Wee Gin Lim, Li-Minn Ang, and Kah Phooi Seng. "An adaptive lossless data compression scheme for wireless sensor networks." *Journal of Sensors* 2012 (2012).

[6] Datasheet, MICAz. "Crossbow technology inc." San Jose, California 50 (2006).

[7] Sadler, Christopher M., and Margaret Martonosi. "Data compression algorithms for energy-constrained devices in delay tolerant networks." In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pp. 265-278. ACM, 2006.

[8] Levis, Philip, and David Gay. *TinyOS programming*. Cambridge University Press, 2009.

[9] Shannon, Claude E. "A mathematical theory of communication." *ACM SIGMOBILE Mobile Computing and Communications Review* 5, no. 1 (2001): 3-55.

[10] Srisooksai, Tossaporn, Kamol Keamarungsi, Poonlap Lamsrichan, and Kiyomichi Araki. "Practical data compression in wireless sensor networks: A survey." *Journal of Network and Computer Applications* 35, no. 1 (2012): 37-59.

[11] Infanteena, S. Denis, and EA Mary Anita. "Survey on compressive data collection techniques for wireless sensor networks." In *Information Communication and Embedded Systems (ICICES), 2017 International Conference on*, pp. 1-4. IEEE, 2017.