International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 7 (2015) pp. 17327-17337

http://www.ripublication.com

Compression of Data In Body Sensor Networks

Viona Pamela Quadros
MTech Computer Science and Engineering
Dept. of Computer Science
SRM University, Kattankulathur 603203
Chennai, Tamil Nadu, India
vionaquadros@gmail.com
Dr. M Pushpalatha
Professor
Dept. of Computer Science
SRM University, Kattankulathur 603203
Chennai, Tamil Nadu, India
pushpalatha.m@ktr.srmuniv.ac.in

Abstract

In recent years, the use of body sensors in medical field has allowed for the remote monitoring and health care of patients. E-health helps in efficient collection, analysis, diagnosis and storage of medical data from distant locations with the use of communication networks. Body sensors used for collecting medical data are resource constrained in energy, storage and processor speed. Most of these sensors are battery powered and large amount of data is collected by them continuously. The data transmission from these sensors to cloud databases results in high energy dissipation. Hence, reduction of data size through compression provides significant energy conservation. These compression techniques impose restrictions of fidelity (lossless techniques), programmability, and adaptability and implement ability. Keeping these factors in mind, in this paper, a hybrid compression technique which is a combination of Deflate, Burrows Wheeler Transform and a variation of Run Length Encoding with threshold N is proposed. The comparative results of the proposed technique with other compression methods like Run Length Encoding, Huffman, Deflate and Aggregated Deflate RLE is shown to provide better compression ratio and energy efficiency. However, there is a slight tradeoff with the compression time, as it increases with the increase in file size.

Index Terms: ADR, BSN, BWT, Deflate, RLE.

Introduction

Body sensor networks (BSNs) are used for remote and continuous monitoring of patients. The sensor units used by patients continuously extract health parameters from patients and send it to cloud databases for storage and diagnosis. In battery powered sensor networks, majority of the energy consumption occurs due to the wireless transmission of sensed data. Using a typical wireless communication system, the transmission energy for one bit is equivalent to 1000 cycles of an embedded processor. Reducing this transmission energy, by compressing the sensed data, is essential to meeting the battery life and form factor requirements of many BSN applications[1]. Hence, the large amount of data generated by the sensor units is subjected to compression to provide significant energy savings.

BSN platform and applications have several constraints which must be considered when designing the compression algorithm. The compression algorithm must be lossless, since the data collected is pertaining to health parameters. The algorithm must conform to the space constraints of body sensor units, enable ease of code development and deployment and require low processing and memory resources [1][2][3]. Some of the compression techniques that conform to these factors are Sensor Lempel-Ziv-Welch (S-LZW) [4], Run Length Encoding (RLE) [1][5], Huffman coding [6][7] and Deflate.

The proposed method is a hybrid combination of Deflate, Burrows Wheeler Transform and Run Length Encoding with Threshold N. The Deflate compression is the first step, in the proposed technique, where the data is divided into a series of blocks. These blocks are individually compressed using LZ77 technique and Huffman coding, which forms the two step process of the Deflate compression. The output of the Deflate method is transformed using Burrows Wheeler transform. This permutation of data is in a form that allows the Run Length Encoding to perform the best possible compression on that data. This forms the final step of the proposed compression method.

This paper is structured as follows. The section 2 discusses related work which consists of compression techniques that conform to the requirements of BSN applications. This is followed by the proposed scheme (section 3) consisting of algorithm and explanations. Section 4 gives the metrics for analyzing results and results. The section 5 contains the conclusion of the paper.

Related Work

The BSN application generated data have requirements such as lossless recovery, large data size, fast transmission and storage. Keeping in mind all these requirements, the acquired sensor data is compressed using appropriate compression algorithm. The reduction in data size, using these techniques reduces the transmission bits which in turn reduces energy consumption. In this section, the various compression algorithms that conform to the BSN application constraints are discussed.

Run Length Encoding and Modification

Run Length Encoding (RLE) is an easy to understand and simple algorithm. In this technique, several runs of the same character are replaced with the number of repetitions of the character followed by the character. This method is most commonly used in black and white images which are represented with 0s and 1s. Also, in text files with sufficient redundancy, this method can provide a good compression ratio[1].

A variation Of RLE is the use of a threshold N (RLE-N)[8]. This method compresses data only if the runs of the characters cross a certain threshold. Else, it will print the characters as in the original output stream. This method of compression is considered in the proposed compression technique because, in cases where sufficient redundancy is not present, the RLE slightly worsens the compression ratio, by about 0.1%. Hence the effect on compression is not necessarily positive. It is suggested that RLE should be avoided if it reduces the file size by less than 30% [9].

The idea behind RLE-N is in the use of a predefined threshold N. If the number of repetitions of the characters is greater than or equal to N, the current symbol and repetition count is output as a pair in compressed format. Else, the original stream is sent to the compressed format.

Lempel-Ziv-Welch

Lempel-Ziv-Welch (LZW) [10] is a universal lossless data compression algorithm that is simple to implement and has the possibility for very high throughput in hardware implementations. UNIX file compression utility and GIF format widely use this method for compression. LZW is a dictionary based technique with is initialized at the start to contain all strings of length one. The character sequence is searched for longest substring match in the dictionary. Successively larger entries are added in the dictionary, to hold back references for new matches. The decoding algorithm scans the encoded string and replaces it with the matched entry found in the dictionary.

LZ77 and LZ78 are lossless compression algorithms which were proposed by Abraham Lempel and Jacob Ziv. LZ77 maintains a sliding window during compression while LZ78 uses an explicit dictionary for encoding. These algorithms have formed the basis for several compression schemes such as GIF and Deflate.

Huffman Encoding

The Huffman Encoding [6][7] is a method of finding an optimal prefix code for a given input stream. It is an entropy coding technique that constructs a variable length code table for encoding a source symbol. The lengths of the code in the Huffman tree are inversely proportional to the frequency of occurrence of the symbol to be encoded.

There are several variations of Huffman coding such as n-ary Huffman coding, Adaptive Huffman coding, Canonical Huffman code and so on. In the present, Huffman coding technique is used as part of several compression techniques like Deflate, JPEG and MP3.

Deflate Compression

The Deflate compression technique is a combination of the LZ77 [4][10] and the Huffman Encoding technique. A Deflate stream consists of a series of blocks with each

block preceded by a 3-bit header. The blocks of data will be encoded depending on the bit values present in the header [11].

The working of Deflate compression has two steps. In the first step, duplicate string elimination takes place using the LZ77 technique. In this method, the duplicated occurrence of a string is replaced with a pointer to that string. This step is followed by bit reduction using Huffman coding. Two Huffman trees are used for compression. One compresses the literals and the second one match distances. The length of each sequence of the literals is inversely proportional to the probability of that symbol needing to be encoded. The literals pertaining to the pointers are found using a hash table.

Aggregated Deflate Run Length Encoding

The Aggregated Deflate Run Length Encoding (ADR)[1] is a compression technique which combines the popular Deflate technique and Run Length Encoding technique. This algorithm has been provided for application in BSN generated data. The result of ADR is optimal for body sensor data and is seen to give a better compression ratio than the popular RLE, Huffman Encoding and Deflate algorithm.

The ADR compression technique compresses the sensor data using the Deflate algorithm. The output is then sent to the Run Length Encoding technique to compress further redundancy in data.

Burrows Wheeler Transform

The Burrows-Wheeler Transform (BWT) is a technique that provides a reversible transformation for text in a form that is easier to compress. It is commonly referred to as “block sorting”, because it rearranges a character string into runs of similar characters [4][12]. By itself, it does not perform any kind of compression, but the data output from this transformation can be fed to the Run Length Encoding algorithm to get favorable results.

The transformation of data is achieved by taking the last column of a sorted matrix of shifted copies of our input stream. The decoder can reconstruct the original data stream by using this last column. The use of BWT is that, the data generated has redundant characters packed consecutively and provides good compression when used with RLE. The amount of data needed to reverse the BWT transformation is equal to the size of the input data and one index position in string.

Proposed Scheme

The proposed compression technique is a hybrid combination of Deflate algorithm, Burrows Wheeler Transform and a variation of Run Length Encoding known as RLE-N. The order of compression is: the original input stream is compressed using Deflate; symbolically represented in the form: Deflate (input). The output from Deflate is transformed using BWT i.e. BWT(Deflate(input)). The Deflate compressed and transformed data is then encoded using RLE-N algorithm i.e. RLE-N (BWT (Deflate (input))).

The Deflate algorithm is the first step of the proposed hybrid compression. This algorithm is a combination of LZ77 and Huffman coding. The data is converted into series of blocks for compression. The header of each block contains 3 bits which give information about the existence of consecutive blocks and type of encoding method used on that block. Deflate works in the following steps.

1. The matching and replacing of duplicate series of bytes within each block, with pointers.
2. Replacing the previously compressed symbols with newly compressed symbols based on frequency of occurrence.

In step 1, LZ77 technique is used to take advantage of repetitive occurrence of symbols in the input stream. This technique is also known as sliding window compression because it has to keep track of some amount of previous data (2kB, 4kB or 32kB) to insert back reference for duplicate series of bytes. In step 2, a Huffman coding technique is performed on the input data subjected to LZ77 compression. This method will create an un-prefixed tree of non-overlapping intervals. It generates sequences of new symbols whose length is inversely proportional to the probability of the symbol needing to be encoded.

The Burrows Wheeler transform is the second step in the proposed hybrid technique. The BWT forward transform involves performing cyclic permutations of the input string, which clusters together character of similar context. However, rather than storing the cyclic permutations of the string of size n, in a matrix of space complexity $O(n^2)$, an array R can be used to store references to the rotated input string. Thus, the space complexity of the algorithm is reduced to O(n). The time complexity of the compression is O(n log n) for creation of array R plus time for sorting using quicksort [12].

The steps of BWT are as follows.

1. The input string is subjected to cyclic permutations and the sorted in lexicographic order.
2. The references of the last character in the sorted strings are stored in an array R.
3. The array R along with index number of row corresponding to original input is taken as output.

Using only the array R and index of original input string, the original input can be recovered. The time complexity for decompression is $O(n) + O(\sum)$, where $\sum$ is the number of characters present in the input.

The final step of the proposed technique is RLE-N. The transformed data output from BWT is now in a form that contains all the repetitive data grouped together. This data can now be compressed to its maximum ability using RLE-N. The advantage of using this variation of Run Length Encoding is that, it takes into account the probability that all runs of specific characters do not have a large amount of repetitions. Hence, instead of replacing every run length with the number of runs followed by character, only the character runs which cross a particular threshold N are replaced. Else, the input stream is written as they occur in the compressed format. It takes into account the redundancy present in the text files and performs compression only if the compression of text files is guaranteed.

The algorithm of RLE-N is given below.

```
Threshold = N
While (not end of string)
    {
    If (current char = previous char)
            runLength++
    else If ( runLength >= Threshold )
            add runLength and previous char
    else
            add previous char runLength times
    }
```

Thus at the end of this 3-step compression, the input data is reduced to its maximum and is now transmitted to the cloud databases. At the receiving end, the decompression is carried out by reversing the steps taken during compression. i.e. Deflate(BWT(RLE-N(input)). The data retrieved is the original, exact and error-free data collected by the body sensor units.

Analysis of Results

This paper gives a hybrid compression technique that considers the resource constraints present in BSN and is applicable for body sensors generated data. The metrics for analyzing results include the compressed file size, compression ratio, compression time and saving percentage in file size. The proposed compression technique is evaluated by applying the algorithm on different file size. The results are used to construct graphs for the different metrics.

The comparison of the compressed file size to the original file size gives the basic idea of the amount of compression that has been performed by the compression algorithms. The graph in Fig. 1 compares the proposed technique (DEF-BWT-RLE) with other compression techniques such as RLE-N, Huffman (HUF), Deflate (DEF) and Aggregated Deflate RLE (ADR). From Fig. 1, it can be seen that the compression techniques Deflate, ADR and DEF-BWT-RLE provide very good compression compared to Huffman and RLE-N. Fig. 2 shows that the proposed technique provides better compression than Deflate and ADR with increase in file size.

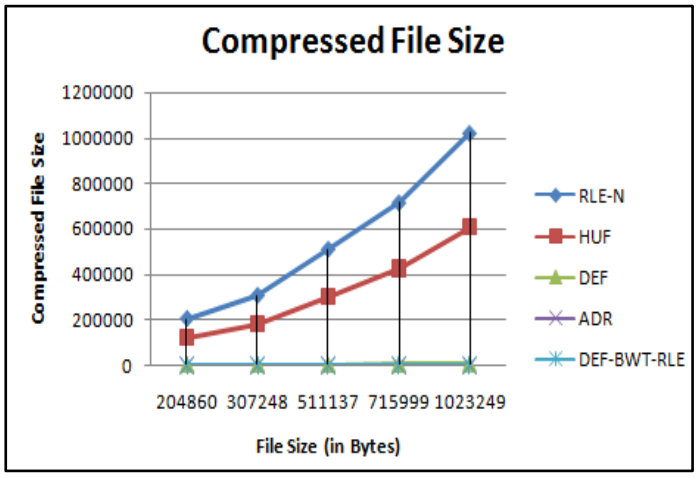

Figure 1: A graph showing the original file size of the data compared to the compressed file size (in bytes).

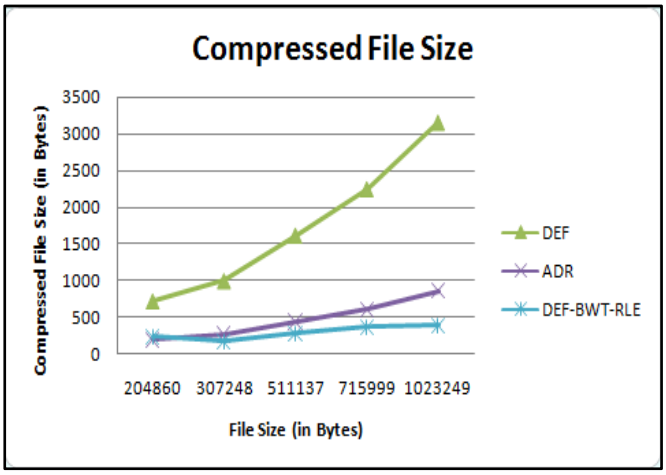

Figure 2: A graph showing the original file size of the data compared to the compressed file size for Deflate, ADR and DEF-BWT-RLE.

Based on the data got from the compressed file size (Fig.1 and Fig. 2), the compression ratio and saving percentage in file size is shown in the graphs, Fig. 3 and Fig. 4 respectively. The data compression ratio (CR) is also known as compression power and is used to quantify the reduction in data size (CS) compared to the original file size (OF). The calculation for the compression ratio is done using the formula (1) given below.

CR = — (1)

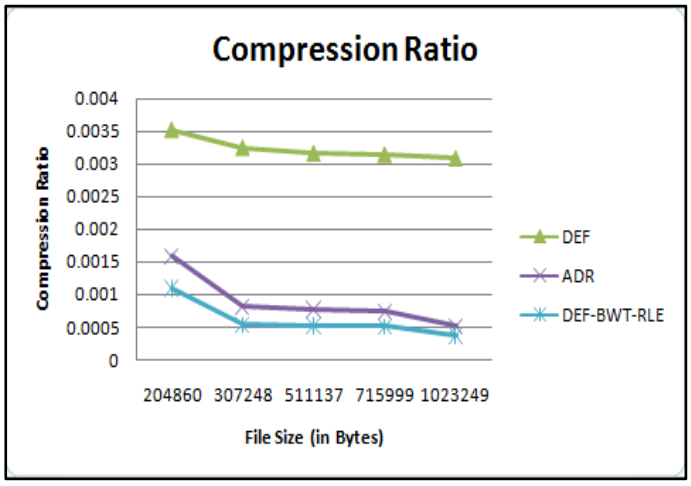

Figure 3: A graph showing the comparison of compression ratio of different compression techniques to their file sizes (in bytes).

The percentage of reduction in file size after the data is passed through a compression algorithm is known as its saving percentage (SP). The calculation for saving percentage in file size is done using the formula (2) shown as follows.

SP = 100 ——— (2)

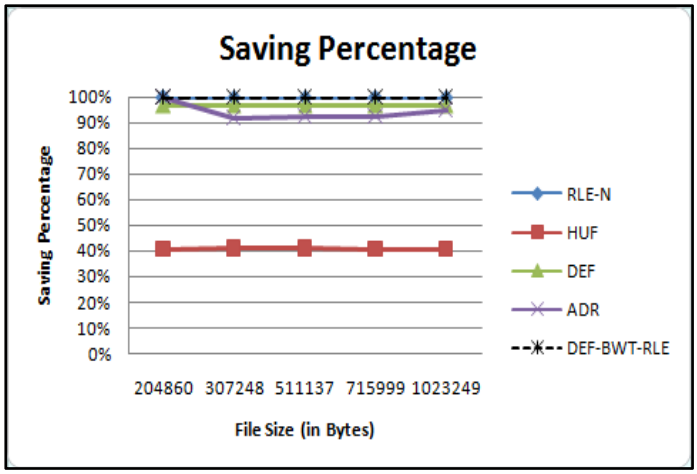

Figure 4: A graph showing the comparison of saving percentage of different compression techniques to their file sizes (in bytes).

From the graphs (Fig. 3 and Fig. 4), we can see that the proposed compression technique performs better than the other compression techniques like Run Length Encoding, Huffman coding technique, Deflate and Aggregated Deflate RLE in terms of compression ratio and saving percentage in file size. This reduction in file size

reduces the amount of transmission bits and hence provides savings in transmission energy.

The compression / decompression time is the difference between the start of the compression / decompression (T2) until the time that the output is received (T2). The compression time (T) is calculated using formula (3) given below.

$$T = T2 - T1 \quad (3)$$

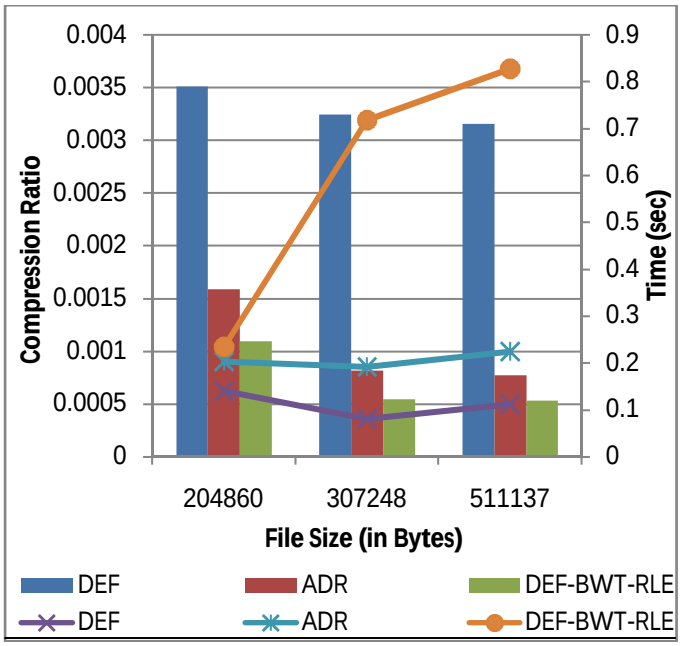

Figure 5: A graph showing the compression ratio and compression time for different compression algorithms.

Fig. 5 gives the graph of compression ratio and compression time of different compression techniques taken for different file sizes. This graph shows that the size improvements gained by the proposed technique comes with computational cost. The large number of string compares needed to perform the compression, increases the compression time. This compression time increases gradually with increase in file size. However, it can be seen that even with the long computation time, the amount of transmission bits reduced is significant. This will in turn reduce the transmission energy, which is the main goal of the proposed technique.

There is a tradeoff in the compression and decompression time because of the use of the slow running Burrows Wheeler Transform. The use of RLE-N technique ensures that there is no unnecessary increase in file size if there is an absence of redundancy.

Conclusions

Body sensors generate data that must be transmitted for analysis, diagnosis and storage. However, the sensor units are resource constrained in battery life, processor

speed and storage. The proposed compression technique takes these parameters into consideration. It performs well in terms of compression ratio and saving percentage. As the file size increases the compression ratio also shows better results. However, the increase in the compression factor by the proposed technique comes with the increase in computation time. The long computation time is overshadowed by the significant reduction of file size in the compressed format. This reduces the transmission energy thereby increasing the battery lifetime of the body sensor units which accomplishes the main goal of the proposed compression technique.

The proposed method of compression is optimal for files of large data sizes since the compression is better. The data to be transmitted should mainly be transmitted for storage and future research or analysis since the time for compression and decompression is large.

References

[1] Basant Tiwari, Dr. Abhay Kumar, 2012, "*Aggregated Deflate-RLE Compression Technique for Body Sensor Network,*" CSI Sixth International Conference on Software Engineering, ISBN: 978-1-4673-2174-7, pp. 293-298.

[2] Tifenn Rault, Abdelmadjid Bouabdallah, Yacine Challal, 2014, "*Energy efficiency in wireless sensor networks: A top-down Survey*", Elsevier, Computer Networks 67 (2014), pp. 104–122.

[3] Yao Liang and Yimei Li, 2014, "*An Efficient and Robust Data Compression Algorithm in Wireless Sensor Networks*", IEEE Communications Letters, vol. 18, no. 3.

[4] C. M. Sadler, M. Martonosi, 2006, "*Data compression algorithms for energy-constrained devices in delay tolerant networks,*" International Conference on Embedded Networked Sensor Systems, ISBN: 1-59593-343-3, pp. 265–278.

[5] Pu, I.M., 2006, "*Fundamental Data Compression"*, Elsevier.

[6] Huffman, D. A., 1952, "*A Method for the Construction of Minimum Redundancy Codes*", Proceedings of the Institute of Radio Engineers, Volume 40, Number 9, pp. 1098-1101.

[7] Henry Ponti Medeiros, Marcos Costa Maciel, Richard Demo Souza, and Marcelo Eduardo Pellenz, 2014, "*Lightweight Data Compression in Wireless Sensor Networks Using Huffman Coding*", International Journal of Distributed Sensor Networks, Volume 2014 (2014), Article ID 672921, pp. 1-11.

[8] Li Bin, NI Guiqiang, Luo Jianxin, Zhang Xue, 2011, "*BWT-based data preprocessing for LZW*", International Conference on Multimedia and Signal Processing, ISBN: 978-1-61284-314-8, pp. 37-40.

[9] B. Balkenhol, S. Kurtz, and Y. M. Shtarkov, 1999, "*Modifications of the Burrows and Wheeler data compression algorithm*", Proceedings of the IEEE Data Compression Conference, pp. 188-197.

[10] Ziv J., Lempel A., 1977, "*A Universal Algorithm for Sequential Data Compression*", IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343.

[11] David Salomon, 2007, "*Data Compression: The Complete Reference (4 ed.)*", Springer, pp.230-241, ISBN: 978-1-84628-602-5.

[12] M. Burrows, and D. J. Wheeler, 1994, "*A block-sorting lossless data compression algorithm*," Technical Report 124, Digital Equipment Corporation, Palo Alto, California.