International Journal of Computer Applications (0975 – 8887)
Volume 24– No.6, June 2011

Median Predictor based Data Compression Algorithm for Wireless Sensor Network

Ashish K. Maurya
Department of Electronics and Computer Engineering
Indian Institute of Technology Roorkee
Roorkee, India

Dinesh Singh
Department of Electronics and Computer Engineering
Indian Institute of Technology Roorkee
Roorkee, India

ABSTRACT

Large scale wireless sensor networks (WSNs) have emerged as the latest trend in revolutionizing the paradigm of collecting and processing data in diverse environments. Its advancement is fueled by development of tiny low cost sensor nodes which are capable of sensing, processing and transmitting data. Due to the small size of sensor nodes there are various resource constraints. It is the severe energy constraints and the limited computing resources that present the major challenge in converting the vision of WSNs to reality. In this paper, we propose a simple and efficient data compression algorithm which is lossless and particularly suited to the reduced memory and computational resources of a wireless sensor networks node. The proposed data compression algorithm gives good compression ratio for highly correlated data. Simulations for the proposed data compression algorithm are performed on TOSSIM.

Keywords

Wireless Sensor Network, Data Compression.

1. INTRODUCTION

A wireless sensor network is a large collection of tiny sensor nodes that contain a sensing unit, a wireless radio transceiver, a microprocessor and a power unit. Sensor nodes are capable of performing some processing, gathering information and communicating with other connected nodes in the network. In a typical application, a WSN is scattered in a region where it is meant to collect data through its sensor nodes. Instead of the conventional methods, WSN deploys a large number of small nodes which gather data to be interpreted in a distributed manner. For ease of deployment, sensor devices should be inexpensive, small, and have a long lifetime, which makes it important to design protocols, software and solutions to make the most efficient use of limited resources of energy, computation and storage in a sensor node [1], [2]. The area of communications and protocol design for sensor networks has been widely researched in the past few years, and many solutions have been proposed and compared.

1.1 Wireless Sensor Network Model

The major components of a typical sensor network are: sensor nodes, the sensor field, the sink and the task manager. Sensors nodes or motes are in charge of collecting data and routing this information back to a sink. A sensor field can be considered as the area in which the nodes are placed. A sink is a sensor node with the specific task of receiving, processing and storing data from the other sensor nodes. The task manager or base station is centralized point of control within the wireless sensor network, which extracts information from the network and disseminates control information back into the network. It also serves as a gateway to other networks, a powerful data processing/storage centre and an access point for a human interface [1].

1.2 Data Compression

Data compression [4] is the process of encoding information using fewer bits than an unencoded representation would use, through use of specific encoding schemes. Compressed data can only be understood if the decoding method is known by the receiver. There are two types Compression techniques: lossless and lossy. In lossless data compression technique; the compressed-then-decompressed data is an exact replication of the original data. WinZip program is an example of lossless compression technique. In lossy data compression technique, the decompressed data may be different from the original data. JPEG is an example of lossy compression technique. Compression is usually broken into two steps: modeling and coding. Modeling describes the generation of the input source which is to be compressed and coding maps symbols from the input alphabet into compact binary sequences. Some popular coding schemes are Huffman coding, arithmetic coding, LZW coding.

In this paper, we propose a simple and efficient data compression algorithm for wireless sensor networks which will save energy by compressing the size of the data. The algorithm is lossless and particularly suited to the reduced memory and computational resources of a wireless sensor node.

2. RELATED WORK

Wireless sensor networks have several resource constraints: limited power supply, bandwidth for communication, processing speed, and memory space. One possible way to achieve maximum utilization of those resources is applying data compression on sensor data. Existing compression algorithms are not applicable for sensor networks because of their limited resources. Therefore, some compression algorithms have been specifically designed for wireless sensor networks [2], [4].

Hans and Schafer [3] present an overview of lossless data compression in the context of audio data. Barr and Asanovi´c show in [4] that the energy required for transmitting a bit can be equivalent to the energy consumption of a thousand microcontroller operations. However, their results were acquired from a Compaq Personal Server handheld, which features 32 megabytes of RAM and 16 kilobytes of cache. This significantly exceeds the resource constraints on many of today’s mote platforms.

Pradhan *et al.* [5] suggested a framework for distributed compression, in which he has used joint source and channel coding that brings about the minimization in the amount of inter-node communication for compression using both a quantized source and correlated side information within each individual node. With the introduction of many application domains like various kind of sensor networks that are severely energy and power constrained, the topic of energy efficiency is becoming more important. Researchers have considered both of the aspects, data compression as well as energy efficiency.

Zhuang and Li [6] have implemented the compression algorithms for seismic data. In this paper, the amount of energy reduction due to the reduction in data after compression has been estimated while considering only the energy costs of communication

Sadler and Martonsi [7] have described a variation of lossless LZW algorithm pertaining to the common sensor platforms with few kilobytes of memory. This version can carry out the compression of the data block with a length of 528 bytes at a time. The S-LZW algorithm causes the saving in energy by the factor of more than 1.5x locally, and over 2.5x as far as the overall network is concerned, for the tests carried out with the data reprieved from real sensor networks. However, the evaluation of the system was made out with delay-tolerant network setting while data was buffered before being transmitted.

Tsiftes et al. [8] have compared the mechanisms for compressing code updates to remotely reconfigure nodes. They have proposed SBZIP algorithm which blends multiple reprocessing with coding steps. However, the characteristic of the decoder part that can still be run on memory constrained sensor platforms is maintained even now. The results show that the saving in energy that could be notched up over the network is up to 67% when GZIP is used.

The EasiPC packet compression mechanism is proposed in [9] by JU and Cui. The requisite for this method is that each packet field is analyzed and classified in advance and then for each category, we have different compression methods associated. Randomly changing fields are always transferred uncompressed while sequence number fields are encoded by difference coding or length variable coding. Either the difference coding or length variable coding is used for sensor reading. In addition to the corresponding transmission delay, the compression gain up to 50% can be achieved by this mechanism due to its distinct feature of removing redundant information from the packet.

Reinhardt, M Hollick and R. Steinmetz [10] have proposed a stream oriented compression scheme based compression technique. In this framework, for efficient data transfer between the nodes the data compression has been shifted into a dedicated layer for sensor network.

A different approach has been explored by Marcelloni et al. [11]: In order to keep the algorithm as simple as possible and to avoid complex computations on embedded nodes, their solution relies on a two-phase coding process based on a lookup table of the size of the analog-digital converter and compresses the raw bits of a sensor reading. Here, a codeword is a hybrid of unary and binary codes supplied by an adequate dictionary similar to the one used for DC coefficient coding in JPEG compression. Since the size of the dictionary is fixed and encoding is done via mapping, the algorithm is well suited for on-the-fly compression. However, the obtainable compression ratio is highly dependent on a good mapping strategy.

Data compression shrinks raw data down to smaller volumes, which is desirable for data communication since the compressed data can require significantly less time and energy to transmit compared to the raw data. Previous research for data compression in communication mainly focuses on how to decrease delay or save required transmission bandwidth. No any algorithm discusses the energy savings with memory storage savings when sensor readings are compressed at the originating node.

We propose a simple compression algorithm particularly suited to the reduced memory and computational resources of a WSN node. The compression algorithm is lossless. Since the sensor nodes are deployed under various circumstances, the correlation of data is unpredictable. Our anticipated goal is to reach good compression ratio in the high correlated data and low correlated data.

3. THE COMPRESSION ALGORITHM

Model of Proposed algorithm

The Median Predictor based Data Compression (MPDC) algorithm consists of three phases:

1. Selector: This phase selects three previous values with current value. Initially, assuming three readings are zero.

2. Median Predictor: It predicts the median value along with lowest value and highest value among three previous values selected and calculates the deviation of current value from the median value.

3. Huffman Coder: It calculates the Huffman code of the deviation obtains from median predictor which results in compress data. The basic idea of Huffman coding is to map an alphabet to a representation for that alphabet, composed of sequences of bits of variable sizes, so that symbols that occur frequently have a smaller representation than those that occur rarely. In this case, the symbols are $R + 1$ (where, R is the resolution of the ADC) and the probabilities decrease with the increase of the values. We have used table of Huffman variable length codes (table 1) used in Marcelloni et al. [11] algorithm. Figure 1 describes the block functioning of proposed MPDC algorithm.

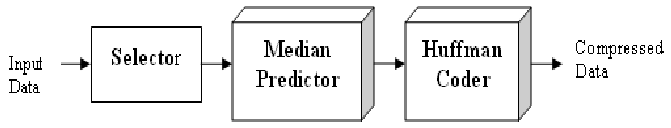

Figure 1 Model of proposed MPDC algorithm

Since the application of sensor nodes is mostly used to report the status of interested area to users, it is more useful to measure x_i acquired by sensor node. Measure x_i is converted to a binary representation r_i on R bits by an ADC, where R is the resolution of the ADC [11]. For each new acquisition x_i, the compression algorithm predicts the median value among three previous values and computes the deviation of current value from median.

Table 1: The Huffman variable length codes [11]

n_i	s_i	d_i
0	00	0
1	010	-1,+1
2	011	-3,-2,+2,+3
3	100	-7,…,-4,+4,…,+7
4	101	-15,…,-8,+8,…,+15
5	110	-31,…,-16,+16,…,+31
6	1110	-63,…,-32,+32,…,+63
7	11110	-127,…,-64,+64,…,+127
8	111110	-255,…,-128,+128,…,+255
9	1111110	-511,…,-256,+256,…,+511
10	11111110	-1023,…,-512,+512,…,+1023
11	111111110	-2047,…,-1024,+1024,…,+2047
12	1111111110	-4095,…,-2048,+2048,…,+4095
13	11111111110	-8191,…,-4096,+4096,…,+8191
14	111111111110	-16383,…,-8192,+8192,…,+16383

```
encode (x, prevArray[], Table)
// x is current value and prevArray[] contains previous three values
   a = prevArray[0]
   b = prevArray[0]
   c = prevArray[2]
   l = minimum(a,b,c)
   m = median(a,b,c)
   h = maximum(a,b,c)
   IF (l <= x <= h) THEN
      SET set_bit TO '0'
      SET pbs TO set_bit
      //pbs is previous bit set
      IF (x >= m)
         SET set_bit TO '0'
         SET pbs TO << pbs, set_bit >>
         // Compute Huffman Code
         HC(diff(x,m))
         SET hmc TO HC(diff(x,m))
         //hmc is Huffman Code
         SET final_code TO << pbs, hmc >>
      ELSE
         SET set_bit TO '1'
         SET pbs TO << pbs, set_bit >>
         HC(diff(x,m))
         SET hmc TO HC(diff(x,m))
         SET final_code TO << pbs, hmc >>
      ENDIF
   ELSE
      SET set_bit TO '0'
      SET pbs TO set_bit
      IF (x > h)
         SET set_bit TO '0'
         SET pbs TO << pbs, set_bit >>
         HC(diff(x,h))
         SET hmc TO HC(diff(x,h))
         SET final_code TO << pbs, hmc >>
      ELSE
         SET set_bit TO '1'
         SET pbs TO << pbs, set_bit >>
         HC(diff(x,h))
         SET hmc TO HC(diff(x,h))
         SET final_code TO << pbs, hmc >>
      ENDIF
   ENDIF
RETURN final_code
```

Figure 2 Pseudo-code of the MPDC encode algorithm

Once final code is generated, it is appended to the bit stream which forms the compressed version of the sequence of measures x_i. Figure 2 summarizes the algorithm used to encode current value x_i. Here, << pbs, hmc >> denotes the concatenation of pbs and hmc. The compression algorithm described in figure 2 can be implemented in a few lines of code and requires only maintaining the first two columns of Table 1 in memory.

4. SIMULATION RESULTS

Compression ratio [2] is the performance metric to compute the performance of data compression algorithm and is defined as:

comp_ratio = 100 * (1 – comp_size / orig_size)

where comp_size and orig_size are, respectively, the size of the compressed and the uncompressed bit stream.

Simulations were performed on TOSSIM [14], a discrete event simulator for TinyOS [13] sensor networks.

In this experiment, assuming R = 14 as 14-bit ADC converter is used and consider samples acquired by a node every 2 minutes during 48 hours (in total, 1440 samples). Considering that uncompressed samples are normally represented by 16-bit unsigned integers. So the original size of uncompressed data = 23040 bits (= 1440*16).

Table 2 summarizes the results obtained by applying the compression algorithm. We note that, though the compression algorithm is very simple, it is able to obtain considerable compression ratios.

Table 2: Results Obtained by applying the proposed compression algorithm

	Sample1	Sample2	Sample3
orig_size	23040 bits	23040 bits	23040 bits
comp_size	7466 bits	7545 bits	7498 bits
comp_ratio	67.60 %	67.25 %	67.46 %

.

5. CONCLUSIONS

Energy efficiency and reducing memory requirement is a critical consideration for sensor network applications, such as those used in surveillance and monitoring. Processing data consumes much less power than transmitting data in wireless medium, so it is effective to apply data compression before transmitting data for reducing total power consumption by a sensor node. In this paper, we have simulated a lossless data compression algorithm particularly suited to the reduced storage and computational resources of a wireless sensor network node. We have simulated the algorithm on TOSSIM and evaluated the performance of algorithm on performance metrics compression ratio by compressing the data at originator node. We obtained the compression ratio 67.60 %, 67.25 % and 67.46 % for three samples respectively. The future work may be extended on actual mote like mica2 mote. The performance of algorithm may be evaluated on different performance metrics such as saving memory percentage, execution time etc.

6. REFERENCES

[1] Akyildiz, I.F.; Weilian Su; Sankarasubramaniam, Y.; Cayirci, E.;, "A survey on sensor networks," *Communications Magazine, IEEE* , vol.40, no.8, pp. 102- 114, Aug 2002.

[2] Kimura, N.; Latifi, S.; , "A survey on data compression in wireless sensor networks," International Conference on Information Technology: Coding and Computing (ITCC 2005), vol.2, pp. 8- 13, April 2005

[3] M. Hans and R. W. Schafer, "Lossless compression of digital audio," IEEE Signal Processing Magazine, vol.18, no.4, pp. 21–32, July 2001.

[4] Kenneth. C. Barr and Krste. Asanovi´c, "Energy-aware Lossless Data Compression," ACM Transactions on Computer Systems, Vol. 24, No. 3, pp. 250–291, August 2006.

[5] Pradhan S., Kusuma J., and Ramchandran K., "Distributed Compression in a Dense Microsensor Network," IEEE Signal Processing Magazine, vol. 19, no. 2, pp. 51-60, 2002.

[6] Y. Zhang and J. Li, "Efficient seismic response data storage and transmission using ARX model-based sensor data compression algorithm," Earthquake Engineering and Structural Dynamics, vol. 35, pp. 781–788, 2006.

[7] C. M. Sadler and M. Martonosi, "Data Compression Algorithms for Energy-Constrained Devices in Delay Tolerant Networks," in Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys), 2006.

[8] N. Tsiftes, A. Dunkels, and T. Voigt, "Efficient Sensor Network Reprogramming through Compression of Executable Modules," in Proceedings of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2008.

[9] H. Ju and L. Cui, "EasiPC: A Packet Compression Mechanism for Embedded WSN," in Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2005

[10] A. Reinhardt, M. Hollick, and R. Steinmetz, "Stream-oriented Lossless Packet Compression in Wireless Sensor Networks," in Proceedings of the Sixth Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2009.

[11] Francesco Marcelloni and Massimo Vecchio, "A Simple Algorithm for Data Compression in Wireless Sensor Networks," IEEE Communications Letters, Vol. 12, No. 6, June 2008.

[12] David Gay, Philip Levis, David Culler, Eric Brewer, "nesC 1.2 Language Reference Manual," August 2005.

[13] Phillip Levis, "TinyOS Programming," June 28, 2006.

[14] Philip Levis and Nelson Lee, "TOSSIM: A Simulator for TinyOS Networks," September 17, 2003.