Journal of Computer Science 5 (6): 466-470, 2009
ISSN 1549-3636

Design of Modified Adaptive Huffman Data Compression Algorithm for Wireless Sensor Network

C. Tharini and P. Vanaja Ranjan
Department of Electrical and Electronics Engineering,
College of Engineering, Guindy, Anna University, Chennai, India

Abstract: Problem statement: Efficient utilization of energy has been a core area of research in wireless sensor networks. Sensor nodes deployed in a network are battery operated. As batteries cannot be recharged frequently in the field setting, energy optimization becomes paramount in prolonging the battery-life and, consequently, the network lifetime. The communication module utilizes a major part of the energy expenditure of a sensor node. Hence data compression methods to reduce the number of bits to be transmitted by the communication module will significantly reduce the energy requirement and increase the lifetime of the sensor node. The present objective of the study contracted with the designing of efficient data compression algorithm, specifically suited to wireless sensor network. **Approach:** In this investigation, the natural correlation in a typical wireless sensor network data was exploited and a modified Huffman algorithm suited to wireless sensor network was designed. **Results:** The performance of the modified adaptive Huffman algorithm was analyzed and compared with the static and adaptive Huffman algorithm. The results indicated better compression ratio. **Conclusion:** Hence the proposed algorithm outperformed both static and adaptive Huffman algorithms, in terms of compression ratio and was well suited to embedding in sensor nodes for compressed data communication.

Key words: Wireless sensor network, data compression, Huffman algorithm

INTRODUCTION

Wireless Sensor Network (WSN) comprises of several autonomous sensor nodes communicating with each other to perform a common task. A wireless sensor node consists of a processor, sensor, communication module powered by a battery. Power efficiency is considered to be a major issue in WSN, because efficient use of energy extends the network lifetime. Energy is consumed by the sensor node during sensing, processing and transmission. But almost 80% of the energy is spent in the communication module for data transmission in sensor network[1]. Sensor networks have a wide range of application in temperature monitoring, surveillance, bio medical, precision agriculture. Failure of sensor node causes a partition of the WSN resulting in critical information loss. Hence there is great interest shown by the many researchers in extending the lifetime of sensor nodes by reducing the energy required for transmission. Several algorithms have been proposed for energy efficient wireless sensor network in literature.

The spatio-temporal correlations among sensor observations are a significant and unique characteristic of the WSN which can be exploited to drastically increase the overall network performance. The existence of the above mentioned correlation in sensor data is exploited for the development of energy efficient communication protocols well suited to WSN. Recently there is a major interest in the Distributed Source Compression (DSC) algorithm which utilizes the spatial correlation in a sensor network for data compression. WSN application requires dense sensor deployment[1] and as a result of this, multiple sensors record information about a single event. Therefore it is unnecessary for every sensor node to send redundant information to the sink node due to the existence of high spatial correlation. Instead a smaller number of sensor measurements might be adequate to communicate the information to the sink with certain reliability. In[2] a distributed way of continuously exploiting existing correlations in sensor data based on adaptive signal processing and distributed source coding principles is discussed. In[6,8] hardware architecture for data compression using adaptive Huffman algorithm for data compression is proposed. In[3,4,7] also the spatial correlation in sensor data is

Corresponding Author: C. Tharini, Department of Electrical and Electronics Engineering, College of Engineering, Guindy, Anna University, Chennai, India

exploited for the data compression. Though research has progressed in the area of data compression, not many have worked in lossless algorithms for wireless sensor networks. Compression algorithms are mainly for data storage and therefore simple and application specific algorithms are required for resource constrained sensor nodes. In[5] a simple compression algorithm based on static Huffman coding particularly suited for memory and computational resource constrained wireless sensor node is proposed. This algorithm exploits the temporal correlation in sensor data and the algorithm computes a compressed version using a small dictionary. In[5], to design the dictionary, the statistics of the data are required, unlike in real time, statistics of the data will change depending on the event. Therefore the objective of the study is to design a simple algorithm to compress sensor data which does not require prior knowledge of the statistics of sensor data and the data compression is performed adaptively based on the temporal correlation in sensor data.

MATERIALS AND METHODS

To establish a modified algorithm for data compression in wireless sensor network, the following simple and adaptive algorithms were simulated and compared.

Simple Huffman algorithm: In the simple Huffman algorithm proposed in[5], each sensor node measure m_i is converted by an ADC to binary representation r_i using R bits, where R is the resolution of the ADC. For each new measure m_i, the compression algorithm computes the difference $d_i = r_i - r_{i-1}$, which is input to an entropy encoder. The encoder performs compression losslessly by encoding differences d_i more compactly based on their statistical characteristics. Each d_i is represented as a bit sequence bs_i composed of two parts s_i and a_i, where s_i gives the number of bits required to represent d_i and a_i is the representation of d_i. Code s_i is a variable length code generated by using Huffman coding. The basic idea of Huffman coding is that symbols that occur frequently have a smaller representation than those that occur rarely. The a_i part of the bit sequence bs_i is a variable length integer code generated as follows:

- If $d_i>0$, a_i corresponds to the n_i lower-order bits of the direct representation of d_i
- If $d_i<0$, ai corresponds to the n_i lower-order bits of the two's complement representation of (d_{i-1})
- If $d_i = 0$, s_i is coded as 00 and a_i is not represented

The procedure used to generate a_i guarantees that all the possible values have different codes. Once bs_i is generated, it is appended to the bit stream which forms the compressed version of the sequence of measures mi. Here $<<s_i,a_i>>$ denotes the concatenation of s_i and a_i. Since transmission of a bit needs energy comparable to the execution of thousand instructions, just saving only a bit by compressing original data corresponds to reduce power consumption[5].

We observe that the proposed algorithm is simple but has an important drawback. In order to assign probability, the statistics of sensor data must be known already. Hence this algorithm may not be suitable for real time sensor data.

Adaptive Huffman algorithm: Huffman coding requires prior knowledge of the probabilities of the source sequence. If this knowledge is not available, Huffman coding becomes a two pass procedure: the statistics are collected in the first pass and the source is encoded in the second pass. In the Adaptive Huffman coding procedure, neither transmitter nor receiver knows anything about the statistics of the source sequence at the start of transmission. The tree at both the transmitter and the receiver consists of a single node that corresponds to all symbols Not Yet Transmitted (NYT) and has a weight of 0. As transmission progresses, nodes corresponding to symbols transmitted will be added to the tree and the tree is reconfigured using an update procedure. Considering a simple 4 bit ADC representation for each data, then before the beginning of transmission, a fixed 4 or 5 bit code depending whether the symbol is positive or negative is agreed upon between the transmitter and receiver.

The actual code consists of two parts: The prefix corresponding to the code obtained by traversing the tree and the suffix corresponding to 4 or 5 bit binary representation corresponding to positive or negative data respectively. In the process of coding, the probability for the incoming source sequence is assigned as the elements get in to the tree formed. This gives rise to a problem where the elements which come in the initial stages of tree formation having lesser probability hold smaller codes. Thereby, the compression ratio obtained is lesser. The pseudo code for adaptive Huffman algorithm is shown in Fig. 1.

Using Adaptive Huffman algorithm, we derived probabilities which dyanamically changed with the incoming data, through Binary tree construction. Thus the Adaptive Huffman algorithm provides effective compression by just transmitting the node position in the tree without transmitting the entire code. Unlike static huffman algorithm the statistics of the sensor data need not be known for encoding the data.

```
module adaptivehuffman_encode(diff, btree)
        createroot( )
        search (diff, btree)
        if(node present)
                increment node->wt
                code = traverse(node)
        else
                prefix= traverse(nyt)
                if(diff >= 0)
                    suffix = diff|4
                if(diff < 0)
                    suffix = diff|5
                code = <<prefix, suffix>>
                nyt->lchild = createnyt()
                nyt->rchild = createnode(diff)
                nyt = nyt->lchild
        endif
        update(btree)
        balance(btree)
endmodule
```

Fig. 1: Pseudo code of adaptive Huffman encoding algorithm

But the disadvantage in this algorithm is its complexity in constructing the binary tree which makes it unsuitable for sensor nodes.

Modified adaptive algorithm: In static and dynamic algorithms, the ultimate objective of compression was achieved with fixed and dynamic probabilities respectively. The main disadvantage of Static Huffman algorithm is that, we do not have the prior knowledge of the incoming source sequence. Also the statistics of sensor data may be varying. In Adaptive Huffman algorithm, though the probabilities are assigned dynamically, because of the increased number of data available in the source sequence, the number of levels and hence the number of bits transmitted increases and is found to be effective only for very frequently and first occurring data. Furthermore, the binary tree construction is based on the order of arrival of incoming data. Hence, both static and adaptive Huffman algorithms are observed to have some drawbacks.

The proposed Modified Adaptive Huffman algorithm overcomes the disadvantages of both static and dynamic Huffman algorithm by combining the advantages of the two algorithms and ultimately increasing the compression ratio.

The algorithm uses a tree with leaves that represent sets of symbols with the same frequency, rather than individual symbols. The code for each symbol is therefore composed of a prefix (specifying the set, or the leaf of the tree) and a suffix (specifying the symbol within the set of same-frequency symbols). Here the Binary Tree is constructed with the incoming elements and the codes framed by traversing the tree as in Adaptive Huffman algorithm. Elements are grouped as nodes and codes are specifically assigned as in Static Huffman algorithm. At every update the weights at every level is checked and updated such that the higher weights will be occupying the initial stages of the tree.

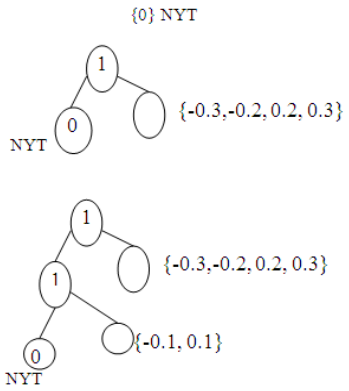

Fig. 2: Binary tree

This enhances the performance in two ways:

- It reduces the number of levels in the tree
- It brings the maximum possible elements to the top level of the tree

Binary tree construction: Binary tree construction used in the algorithm is shown in Fig. 2, where each node consists of set of elements. Considering the temporal correlation in sensor data only the difference of the sensor data $d_i = r_i - r_{i-1}$ is encoded. Consider for example the set {-0.2, 0.1, 0.3} is transmitted. Initially the encoder will have only the NYT node. Since -0.2 occurs for the first time, it is transmitted as 10010 and a new node which contains this data is inserted. When the next data 0.1 arrives, the tree is traversed in search of the node1 which contains the data 0.1 in the binary tree. Since the node is not available, the code corresponding to NYT node i.e., 0 is transmitted, followed by 1 corresponding to the data 0.1. This is followed by the insertion of the node1 in the tree. So the code transmitted is 01. For the next data is 0.3, the tree is traversed in search of the node 2 containing the data 0.3. Since node 2 is already available in the tree, the prefix corresponding to node traversal 1 is transmitted. The binary representation of the array index containing the data is transmitted as suffix i.e., 11 corresponding to 3 which is the array index containing the data 0.3 is transmitted. So the code is transmitted is 111.

Initially the algorithm starts with subsets each having different probability. But this does not require to transmit the tree prior to decompression. We can decide that the tree is always the same when encoding or decoding.

```
module modifiedhuffman_encode(diff, btree)
        //Creation of root in transmitter.
        createroot ( )
        //Traversal of tree in search of node.
        search (diff, btree)
        if (node present)
                prefix = traverse (node)
                suffix = arrayindex (diff)
                increment node->wt
        else
                prefix = traverse (nyt)
                if(diff >= 0)
                        suffix = diff4
                if(diff < 0)
        suffix = diff5
        nyt->lchild = createnyt( )
        //Insertion of node in the tree.
        nyt->rchild = createnode (diff)
        nyt = nyt ->lchild
        code = <<prefix, suffix>>
        update (btree)
        balance(btree)
endmodule
```

Fig. 3: Pseudo-code for modified adaptive Huffman encoding algorithm

In the proposed algorithm, once we reach a leaf while decoding, we know how many bits are there still to be read and that allows us to make a single operation to get a bit string which is readily converted to an integer. The pseudocode of modifiedhuffman encoding algorithm is shown in Fig. 3.

Compression ratio: Compression ratio is calculated as defined in[5]. The formula used for compression ratio analysis is given as follows:

Compression ratio = 100(1- compressed size/original size)

Compressed size is the number of bits obtained after compression and original size will be the total number of bits required without using compression algorithm.

RESULTS

The temperature data from three locations on the Squannacook river, Nashua river Watershed was obtained from[9] and simulations were carried out. Two sets of data (each 200) were considered in our simulation. One set corresponds to highly correlated data and the other set corresponds to medially correlated data. The autocorrelation graphs are represented in Fig. 4 and 5 corresponds to the two sets of data, computed using PAST software.

The performance of the modified adaptive Huffman, Static and Adaptive Huffman compression algorithm were analyzed in terms of compression in number of bits required for transmission is shown in Fig. 6 and the compression ratio of each algorithm is shown in Fig. 7.

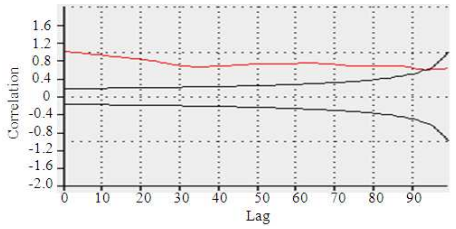

Fig. 4: Autocorrelation plot for highly correlated data

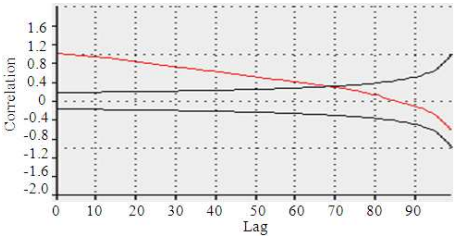

Fig. 5: Autocorrelation plot for medially correlated data

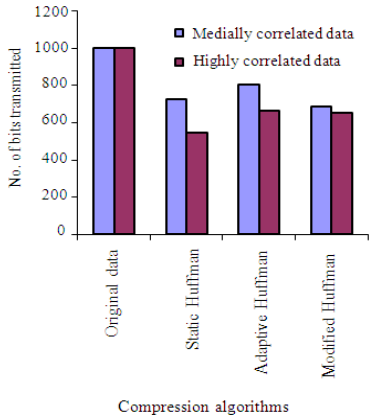

Fig. 6: Compression analysis in terms of No. of bits transmitted

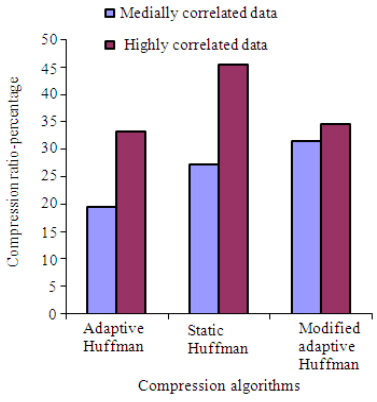

Fig. 7: Compression ratio

DISCUSSION

From Fig. 7, it is observed that, modified adaptive Huffman algorithm provided better compression ratio for medially correlated data than either static or adaptive Huffman algorithms. Adaptive Huffman algorithm provided only 20% compression for medially correlated data and its performance was good only for highly correlated and repeated data. Also adaptive Huffman algorithm required more levels in Binary tree construction which increased the complexity in encoding and decoding. As the sensor nodes are resource constrained this algorithm was not suitable for wireless sensor node.

The performance of the Static Huffman algorithm was found to be better for highly correlated data. But the algorithm was not adaptive for changing statistics of the sensor data. Hence this algorithm was found wanting when the sensor data statistics were constantly changing or unknown, as in a wireless sensor network. The results of the proposed modified adaptive Huffman algorithm, on the contrary, clearly prove that it is adaptive for changing correlations in sensor data. Also, this algorithm provides 10-15% improvement in compression of medially correlated data when compared to the other two algorithms. The binary tree construction of the proposed algorithm requires less number of levels, simplifying the process of encoding and decoding. Consequently, less memory is utilized making it suitable for embedding in a wireless sensor node.

CONCLUSION

A simple algorithm namely modified adaptive Huffman algorithm which does not require prior knowledge of the statistics of sensor data is proposed for data compression in the sensor network. The data compression is performed adaptively based on the temporal correlation in the sensor data. This modified adaptive Huffman encoding algorithm effectively combines the advantages of static and adaptive Huffman algorithms to provide effective compression by reducing the number of levels in the binary tree. The implementation of this algorithm shows better compression ratio than adaptive Huffman algorithm. Since the number of levels is restricted, this algorithm requires less computation than adaptive huffman which is an important requirement for wireless sensor nodes. This algorithm outperforms the static and adaptive Huffman algorithm in providing better performance in terms of number of bits transmitted.

REFERENCES

1. Akylidiz, I.F., W. Su, Y. Sankarasubramaniam and E. Cayirici, 2002. Wireless sensor networks: A survey. Comput. Networks, 38: 393-422. DOI: 10.1016/S1389-1286(01)00302-4
2. Chou, J. and K. Ramachandran, 2003. A distributed and adaptive signal processing approach to reducing energy consumption inn sensor networks. Proceeding of the INFOCOM 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, Mar. 30-Apr. 3, IEEE Xplore Press, USA., pp: 1054-1062. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1208942
3. Tang, C., C.S. Raghavendra and K.V. Prasanna, 2003. An energy efficient adaptive distributed source coding scheme in wireless sensor networks. Proceeding of the IEEE International Conference on Communications, May 11-15, IEEE Xplore Press, USA., pp: 732-737. DOI: 10.1109/ICC.2003.1204270
4. Ye, Q., Y. Liu and L. Zhang, 2006. An extended DISCUS scheme for distributed source coding in wireless sensor networks. Proceeding of the International Conference on Wireless Communications, Networking and Mobile Computing, Sept. 22-24, IEEE Xplore Press, Wuhan, pp: 1-4. DOI: 10.1109/WiCOM.2006.286
5. Francesco Marcelloni, 2008. A simple algorithm for data compression in wireless sensor networks. IEEE. Commun. Lett., 12: 411-413. DOI: 10.1109/LCOMM.2008.080300
6. Lin, M.B., J.F. Lee and G.E. Jan, 2006. A lossless data compression and decompression algorithm and its hardware architecture. IEEE Trans. Very Large Scale Integrat. Syst., 14: 925-936. DOI: 10.1109/TVLSI.2006.884045
7. Pradhan, S.S., Julius Kusuma and K. Ramachandran, 2002. Distributed compression for dense microsensor networks. IEEE. Signal Proc. Mag., 19: 15-60. DOI: 10.1109/79.985684
8. Cesare, A., Romolo Camplani and C. Galperti, 2007. Lossless compression techniques in wireless sensor networks: Monitoring microacoustic emissions. Proceeding of the IEEE International Workshop on Robotic and Sensor Environments, Oct. 12-13, IEEE Xplore Press, Ottawa, Canada, pp: 1-5. DOI: 10.1109/ROSE.2007.4373963
9. http://www.mass.gov/dep/water/resources/