IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 3, Issue 4 (Nov. – Dec. 2013), PP 01-07
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Power Efficient Adaptive Compression Technique for Wireless Sensor Networks

S.Selvarani[1], S.Selvaraju[2], P.Yasodha Devi[3], S.Kalaivani[4], M.Vasanth[5], M.Kannan[6]

[1](Department of ECE, Muthayammal College of Engineering, India)
[2](Department of ECE, Muthayammal College of Engineering, India)
[3](Department of ECE, Muthayammal College of Engineering, India)
[4](Department of ECE, Muthayammal College of Engineering, India)
[5](Department of ECE, Mahendra College of Engineering, India)
[6](Department of EEE, Muthayammal College of Engineering, India)

Abstract: *Wireless sensor network consists of many tiny disposable low power devices called nodes, which are spatially distributed in order to perform various application global tasks. These nodes form a network by communicating with each other either directly or through other nodes. Sensor networks are fundamentally constrained by the difficulty and energy expense of delivery information from sensor to sink. Power saving is a critical issue in wireless sensor networks since sensor nodes are powered by batteries which cannot generally change or recharged. In this paper we have focused on radio communication which is often considered as the main cause of energy consumption. One of the solutions to this problem would be generally achieved by reducing the size of the transmitted/received data through data compression. A new adaptive compression technique will be considered for lossless data compression which reduces the amount of data that must be passed through the network and to the sink and thus have energy benefits that are multiplicative with the number of hops the data travel through the network. Possible extensions of this paper, use RUN LENGTH CODING to compress decompress the data in WSN.*

Keywords: *WSN, Data Compression, Huffman Coding, Run Length Coding.*

I. INTRODUCTION

In general, a wireless sensor network (WSN) consists of spatially distributed autonomous sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance and is now used in many industrial and civilian application areas, including industrial process monitoring and control, machine Health monitoring, environment and habitat monitoring, healthcare applications, home automation, and traffic control [1].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. Sensor networks have a wide range of application in temperature monitoring, surveillance, bio medical, precision 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. Wireless sensor networks with long lifetime requirements have severe power constraints.

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. Huffman coding particularly suited for memory and computational resource constrained wireless sensor node is proposed [2].

The concept of wireless sensor networks is based on a simple equation: Sensing + CPU + Radio = Thousands of potential applications. [3, 4].Power saving is a critical issue in wireless sensor networks (WSNs) since sensor nodes are powered by batteries which cannot be generally changed or recharged. As radio communication is often the main cause of energy consumption, extension of sensor node lifetime is generally achieved by reducing transmissions/receptions of data, for instance through data compression. Huffman algorithm for data compression is proposed.

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.

In this paper, we introduced a new method called as Run Length coding. Using this method we reduced the encoding time but Huffman algorithm the encoding time is large when compare to run length coding. These methods ultimately increase the compression ratio and throughput.

II. HUFFMAN CODING

Huffman coding basically divided in to two categories: -

1. Static Huffman coding
2. Adaptive/ dynamic Huffman coding

Static Huffman coding suffers from the fact that the uncompressed need have some knowledge of the probabilities of the symbol in the compressed files. This can need more bits to encode the file. If this information is unavailable compressing the file requires two passes. FIRST PASS finds the frequency of each symbol and constructs the Huffman tree. SECOND PASS is used to compress the file. We already use the concept of static Huffman coding using ternary tree and we conclude that representation of Static Huffman Tree using internal nodes, Path length, height of the tree, in memory representation, in fast searching and in error detection & error correction. Now here we try to use the concept of adaptive Huffman coding using ternary tree. Faller, Gallager, first conceived adaptive Huffman coding independently. Knuth contributed improvements to the original algorithm, and the resulting algorithm is referred to as algorithm FGK. All of these methods are defined- word schemes that determine the mapping from source messages to code words on the basis of a running estimate of the source message probabilities. The code is adaptive, changing so as to remain optimal for the current estimates. In this way, the adaptive Huffman codes responds to locality, in essence, the encoder is learning the characteristics of the source. The decoder must learn along with the encoder by continually updating the Huffman tree so as to stay in synchronization with the encoder. Here we are given the concept of error detection and error correction. And the main point is that, this thing is only beneficial in TERNARY TREE neither in binary tree nor in other possible trees [6].

III. 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.

Using Adaptive Huffman algorithm, we derived probabilities which dynamically 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 [8].

```
void compress(int data)
{
   HuffStruct *encoder; char buffer[1514];
       encoder= Huff_Initialize_Adaptive_Encoder(257);
   while(data) {
      write_byte(encoder, buffer, data);
   }
}
      void decompress(unsigned int data)
      {
      HuffStruct *encoder; char writeBuffer[1514];
```

```
encoder = Huff_Initialize_Adaptive_Encoder(257); while(data) {
        read_byte(encoder, writeBuffer, data);
    }
    }
```

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

IV. 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[9] 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 [10].

V. BINARY TREE CONSTRUCTION

Binary tree construction used in the algorithm is shown in Fig 1, 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[11].

Initially the algorithm starts with subsets each having different probability. But this does not require transmitting the tree prior to decompression. We can decide that the tree is always the same when encoding or decoding [12].

A binary tree corresponding to the Huffman code is called Huffman tree. The task of constructing the Huffman code is equivalent to the problem of constructing a corresponding tree. We present a general scheme for constructing the Huffman tree: Make a list of coded symbols (in this case we consider each character as a singleton binary tree whose weight is equal to the weight of the character).From the list, select 2 nodes with the least weight. Will form a new node and join it as a child, two nodes chosen from the list. The weight of the generated node is set equal to the sum of the weights of its child nodes. Add the generated node to the list. If the list is more than one node, then repeat 2-5 [13].

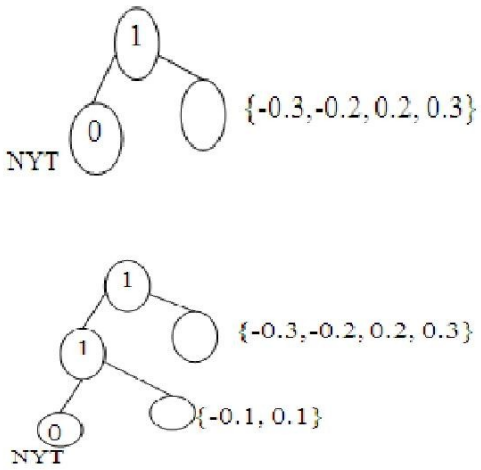

Fig 1. Binary tree

Compression ratio: Compression ratio is defined as compressed size to the original size [5]. The formula used for compression ratio analysis is given as follows:
Compression ratio = 100(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.

VI. RUN LENGTH CODING

Run length coding replaces sequences of the same data values within a file by a count number and a single value.
Example:

ABBBBBBBBBCDEEEEF

The 17 byte of data string has to be compressed as 10 bytes of A *8B C D *4E F

VII. SIMULATION RESULT

By using GloMoSim simulator the throughput, compression ratio, bandwidth utilization and energy consumption occurring in the network is analyzed. Graph is plotted with different network parameter shown in table.

Table of Network Parameter

PARAMETRES	VALUES
PARTITION-NUM-X	1
PARTITION-NUM-Y	1
TERRAIN-RANGE-X	250
TERRAIN-RANGE-Y	250
NUMBER-OF-NODES	30
NODE-PLACEMENT	UNIFORM
MOBILITY	NONE
ROUTING PROTOCOL	AODV
PROPAGATION-LIMIT	-111.0
TEMPARATURE	290.0

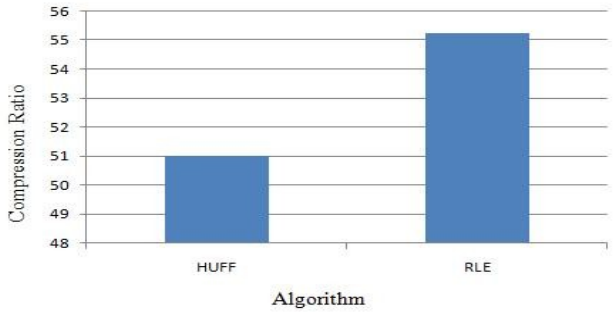

Fig 2. Compression ratio Vs Algorithm.

In Fig 2, Encoded data can take up less storage space and less bandwidth for trans/reception. Digital data are compressed by finding repeatable patterns of binary 0s and 1s. The more patterns can be found, the more the data can be compressed in wireless sensor network. 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. Due to data compression we achieve better energy and increase network life time. Depending on the compression ratio increase the better bandwidth utilization and reduce the energy consumption. In Huffman, we encode the data by using the binary tree its very complex. In order to reduce the complexity we use run length coding method to encode the data. Through the run length encoding the data as it replaces sequences of the same data values within a file by a count number and a single value. The RLE compress the 55.25% of the data while transmitting. But in Huffman the compression level is 51%.

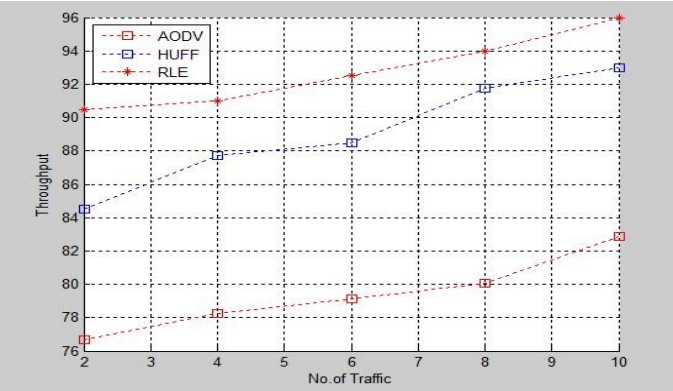

Fig.3.Throughput.

In Fig.3, Throughput is the average rate of successful message delivery over a communication channel. It is inversely proportional to the traffic in the channel. Because, if the traffic is increase the load will be increase in the channel. There will be congestion in the network while transmitting the data's. So the throughput level will be decreased. The graph is simulated using AODV protocol. Without compression the performance of throughput analysis in AODV is low. In Huffman technique, the encoding process is large when compare to run length encoding technique. Because binary formation is used in the Huffman. But in RLE, the run length count method is used for encoding. It will reduce the length of the data. So the encoding procedure is easy in the RLE technique.

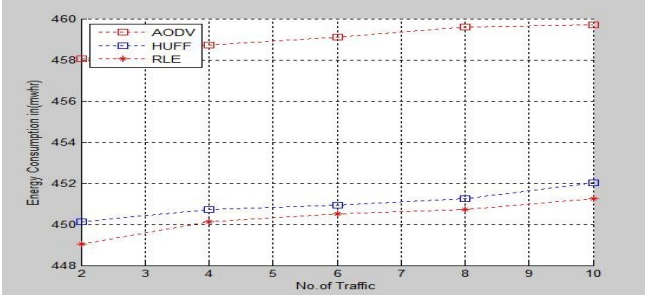

Fig. 4.Energy Consumption.

In Fig.4, when the number of traffic is increase energy consumption of the network also increased. Compare to AODV we get less energy consumption in Huffman. Our motivation is compress the data of source node and increase the network life time. The energy is calculating from, how much amount of energy is consumed in a process. usually the power consumption of the receiving circuitry is often greater than transmitting circuitry. When the network traffic increase, the average energy consumed by the sensor nodes will also increase. This is because of network congestion loading to packet collision and retransmission. When a compression technique is used to encode the source data the number of data bytes through the network will be reduced and reduces the energy consumption. This is clear from the graph that without any energy compression scheme, the energy consumption is more. However, while using compression algorithm like Huffman.

If and Run length coding the average energy consumption is less. If is also observed that compared to Huffman scheme, RLE has better compression & hence energy conservation.

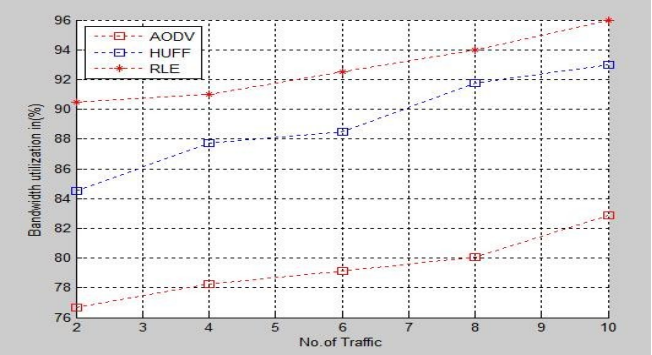

Fig. 5.Bandwidth utilization.

In Fig 5, Bandwidth utilization is defined as measures the amount of data transferred over a network. The graph is simulated using AODV protocol. Without compression the performance of bandwidth utilization in AODV is low. The encoding process is simple in run length coding when compare to Huffman technique. The RLE compress the 55.25% of the data while transmitting. But in Huffman the compression level is 51%. So the bandwidth utilization of RLE is better when compared to Huffman technique.

VIII. 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 by a tree formation 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 this algorithm uses run length encoding for data compression in wireless sensor nodes. This algorithm outperforms the static and adaptive Huffman algorithm in providing better performance in terms of number of bits transmitted. The RLE compress the 55.25% of the data while transmitting. But in Huffman the compression is 51%.

References

[1] Jason Lester Hill, "System Architecture for Wireless Sensor Networks" University of California, Berkeley Spring 2003.
[2] Chris Townsend, Steven Arms Micro Strain,"Wireless Sensor Networks: Principles and Applications" Inc.
[3] Jennifer Yick, Biswanath Mukherjee, Dipak Ghosal "Wireless sensor network survey" Department of Computer Science, University of California, Davis, CA 95616, United States
[4] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci "A Survey on Sensor Networks*" IEEE Communications Magazine August 2002.*
[5] Cesare Alippi, Romolo Camplani, Cristian Galperti Dipartimento di Elettronica e Informazione, Politecnico di Milano, "Lossless Compression Techniques in Wireless Sensor Networks: Monitoring Micro acoustic Emissions" *IEEE* Robotic and Sensors Envir International Workshop ononments Ottawa, Canada, 12-13 October 2007.
[6] Cauligi S. Raghavendra and Viktor K. Prasanna, Caimu Tang "An Energy Efficient Adaptive Distributed Source Coding Scheme in Wireless Sensor Networks" *2003 IEEE.*
[7] Qianyu Ye, Yu Liu, Lin Zhang "An Extended DISCUS Scheme for Distributed Source Coding in Wireless Sensor Networks*"* Beijing University of Posts and Telecommunications Beijing, China.
[8] Francesco Marcelloni and and Massimo Vecchio "A Simple Algorithm for Data Compression in Wireless Sensor Networks*" IEEE COMMUNICATIONS LETTERS,* VOL. 12, NO. 6, JUNE 2008.
[9] Ming-Bo Lin, Jang-Feng Lee, and Gene Eu Jan "A Lossless Data Compression and Decompression Algorithm and Its Hardware Architecture" *IEEE TRANSACTIONS on very large scale integration (vlsi) systems,* vol. 14, no. 9, September 2006.

[10] S. Sandeep Pradhan, Julius Kusuma, and Kannan Ramchandran *"*Distributed Compression in a Dense Micro sensor Network" *IEEE SIGNAL PROCESSING MAGAZINE 1053-5888/02/$17.00©2002IEEE*.

[11] Cesare Alippi, Romolo Camplani, Cristian Galperti Dipartimento di Elettronica e Informazione, Politecnico di Milano, P.za L.da Vinci 32, 20133 Milano, "Lossless Compression Techniques in Wireless Sensor Networks: Monitoring Microacoustic Emissions*"* *IEEE International Workshop on Robotic and Sensors Environments Ottawa*, Canada, 12-13 October 2007.

[12] Yusof. Mohd Kamir, Mat Deris. Mohd Sufian, and Abidin. Ahmad Faisal Amri *"*Study of Efficiency and Capability LZW++Technique in Data Compression" World Academy of Science, Engineering and Technology 59 2009.

[13] Y. Zhao and J. Garcia-Frias, University of Delaware Newark, "Joint Estimation and Data Compression of Correlated Non-Binary Sources Using Punctured Turbo Codes" Conference on Information Sciences and Systems, Princeton University, March 20{22, 2002}.