AN EFFECTIVE DATA COMPRESSION TECHNIQUE EXPLOITING SPATIAL-TEMPORAL CORRELATIONS IN WSN

[1]Dr.P.Suresh Pandiarajan, [2]V.Krishna meera
[1]Associate professor, [2]Assistant professor
[1]Department of ECE,
[1]P.S.R.Engineering college,Sivkasi,India

***Abstract: Data* compression and dimensionality is used to reduce the data combination and aggregation application to prevent data congestion in the wireless sensor network. A new distributed algorithm of data compression based on hierarchical cluster model for sensor networks is proposed. This distributed algorithm consists of two different data compression algorithms. At first "Delta encoding" is used to compress the data by merge the consecutive data difference in the network and "Run length encoding" in which similar data are combined together . Then we will achieves an effective data compression without losing significant data accuracy.**

***Index Terms*–compression, cluster, run length encoding.**

Introduction

Any sensor network such as environment monitoring systems with sensor nodes need to collect data periodically and transmit them to the data sink through multi-hops. The emerging technology of compressive sensing (CS) opens new frontiers for data collection in sensor networks and target localization in sensor networks. CS method can substantially reduce the amount of data transmissions and balance the traffic load throughout the entire network. The proposed work based on spatial –temporal compression in which data can be compressed. Wireless sensor networks (WSNs) are comprised of stand-alone, self-powered sensors that monitor some aspect of their surroundings and report information wirelessly. WSNs have great potential in several diverse fields, but applications we will be focusing on in this thesis involve long-term monitoring and surveillance. In these types of applications, the goal is to have the network cover the largest possible area while maintaining a desired level of spatial and temporal resolution. More spatial resolution is achieved by packing the coverage area with a higher density of nodes. More temporal resolution is achieved by having each node sample its environment at a higher rate.

At first it will choose the cluster head by highly efficient data gathering method and then it will collect the cluster members by radius 0f 25 cm, it will send the data to cluster head in which it can compressed by using delta encoding and run length encoding . In Delta encoding it can be group the consecutive data difference and in Run length encoding similar data can be grouped together through this we can attain the effective data compression in which we can gain efficient energy in the network.

2. PROPOSED WORK

In our proposed system exploits both spatial and temporal similarity in observed data for efficient data compression. It is highly effective for hierarchical networks. since compression takes place at the energy rich CH, energy balancing is also achieved. In our paper has three modules .There are data aggregation, delta encoding, run length encoding. First to collect the data in the hierarchical network and forming cluster to elect the cluster head and the data compression is takes place. Our objective is to perform spatial-temporal compression on the monitored data to maximize energy and bandwidth efficiency. Our major requirements are as follows:

1. The accuracy of the recovered data must fall within the desired error bound
2. The number of data transmissions and the other sources of energy waste must be minimized
3. The algorithms executed on the sensor nodes must have low-complexity implementations and must have minimal memory requirements

4. The exact correlation structure of the measured process at any given time is unknown to the entire network

The data compression is used to reduce the energy consumption and saves the sensors life time. The compression takes place at the cluster head by using hierarchical networks.

2.1Node creation

Sensor node is the primary working component that performs various activities like cluster creation, data collection and transfer data among switching centers.

The sensor nodes are created by using hierarchical networks and forming cluster. The cluster is defined by group of nodes in a sensor network and the clustering is formed by choose the nearest node in the radius 2.5cm in the sensor networks.

2.2 Cluster Head Election

Given the geographic location of the central point of a cluster-area, the sensor node that is the closest to the central point will become the CH. Since the sensor nodes do not know who is the closest to the central point of a cluster-area, and we do not know if there is a sensor node falling into the close range of the central point, we let all nodes within the range of Hr from the center be the CH candidates of the cluster, where r is the transmission range of sensors. The value of his determined such that there is at least one node within H hops from the central point of a cluster. To elect the CH, each candidate broadcasts a CH election message that contains its identifier, its location and the identifier of its cluster. The CH election message is propagated no more than 2 H hops. After a timeout, the candidate that has the smallest distance to the center of the cluster among the other candidates becomes the CH of the cluster. Even for the extreme case that no sensor node falls within H hops from the central point occurs, there will be no CH for this cluster-area and the nodes in this cluster-area will accept the invitation from neighboring CHs and become members of other clusters. Thus, no node will be left out of the network.

2.3 Distributed Implementation

This section presents a distributed implementation of the clustering method.

Every sensor node knows its geographic location. This location information can be obtained via attached GPS or some other sensor localization techniques.

- The sink knows the area of the whole sensor field, but does not need to know the location information of all sensor nodes. This is a reasonable assumption, since in the most applications of the sensor networks, the sink usually knows the area that has sensors deployed for surveillance or environmental monitoring. In our distributed algorithm, the sink divides the field into C cluster-areas, calculates the geographic central point of each cluster-area, and broadcasts the information to all sensor nodes to elect CHs. The sensor node that is the closest to the center of a cluster-area is selected to be the CH. The CHs then broadcast advertisement messages to sensor nodes to invite sensor nodes to join their respective clusters.

2.4 Sensor Node Clustering

After a CH is elected, the CH broadcasts an advertisement message to other sensor nodes in the sensorfield, to invite the sensor nodes to join its cluster. An advertisement message carries the information: theidentifier and location of the CH, the number ofhop that the message has traveled. The hop count is initialized to be 0.When a sensor node receives an advertisement message, if the hop count of message is smaller than it recorded from the same CH, it updates the information in its record including the node of previous hop and the number of hop to the CH, and further broadcasts the message to its neighbor nodes; otherwise, the message is discarded. The maximal hop count for the advertisement message is set to hops and so that all nodes can receive the advertisement messages from at least one CH.

2.5 Calculating Central Points of Cluster-areas:

Given a sensor field and the number of cluster C to be divided to, the sink needs to find out the central points of C cluster-areas. We first divide the whole sensor field into small grids. Then, we place a virtual node at the center of each grid to represent the grid. C nodes in the grids will be chosen as the approximate central points of the cluster-areas. We use an auxiliary graph GA=VA,EA to help finding the central points, where VA is the set of nodes in the grids, and each node vi in VA has an edge to each of the nodes in its neighboring grids. An example of calculating the central points of cluster-areas: an irregular sensor field is roughly divided into small grids, and a virtual node is placed at the center of each grid. 5 nodes are chosen as the approximate central points. Except those on the border of the sensor field, has 8 neighboring grids. The distance of all edges in EA is set to 1. We compute a subset of nodes VC, VC⊂VA and |VC| =C, such that the total distance from all nodes in VA to their nearest nodes in VC is minimized. The nodes in VC are the approximate central points of the C cluster-areas in the sensor field. We use the same iterative algorithm presented in Section 4.2 to compute the set of nodes VC from VA in graph GA. After computing VC, the sink can calculate the geographic locations of the nodes in VC, which are the approximate locations of C central points of the cluster-areas. The sink then broadcasts the locations information of central points to all sensor nodes for CHs election. The size of the grids that the sink divides the sensor field to depends on the accuracy of locating the central points. The smaller the size is, the more accurate the locations information will be, but it incurs more computation cost in this case. In our simulation, we simply set the grid size as a×a.

2.6Hierarchical network

Hierarchical network models are iterative algorithms for creating networks which are able to reproduce the unique properties of the scale-free topology and the high clustering of the nodes at the same time

The hierarchical network model is part of the scale-free model family sharing their main property of having proportionally more hubs among the nodes than by random generation; however, it significantly differs from the other similar models (Barabási–Albert, Watts–Strogatz) in the distribution of the nodes' clustering coefficients: as other models would predict a constant clustering coefficient as the function of the degree of the node, in hierarchical models nodes with more links are expected to have a lower clustering coefficient. Moreover, while the Barabási-Albert model predicts a decreasing average clustering coefficient as the number of nodes increases, in the case of the hierarchical models there is no relationship between the size of the network and its average clustering coefficient.

The development of hierarchical network models was mainly motivated by the failure of the other scale-free models in incorporating the scale-free topology and high clustering into one single model. Since several real-life networks (metabolic networks, the protein interaction network, the World Wide Web or some social networks) exhibit such properties, different hierarchical topologies were introduced in order to account for these various characteristics. The difference between the hybrid and hierarchical network is hub it can be used in hybrid network therefore it send the data to the hub once it last or corrupt the data we need to resend but i hierarchical network the data can be directly send to the base station. Hybrid networks use a combination of any two or more topologies, in such a way that the resulting network does not exhibit one of the standard topologies (e.g., bus, star, ring, etc.). For example a tree network connected to a tree network is still a tree network topology. A hybrid topology is always produced when two different basic network topologies are connected.

3Block Diagram

The flow of the various processes is given by the following sequence diagram of fig.1

Block Diagram

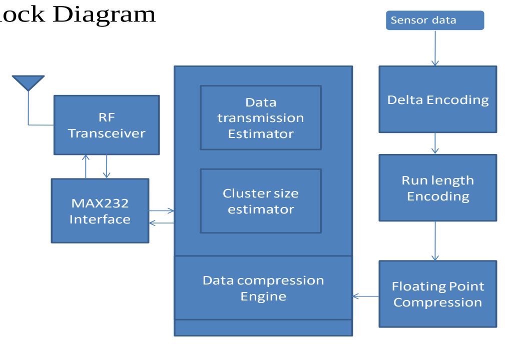


Fig 1 Block diagram

4 Data Aggregation

A wireless sensor network typically consists of a sink node sometime referred to as a base station and a number of small wireless sensor nodes. The sensor nodes monitor a geographical area and collect sensory information and Sensory information is communicated to the base station through the wireless hop by hop transmission. Data aggregation reduces the amount of network traffic which helps to reduce energy consumption on sensor nodes. The data aggregation is used to collect the data from sensor node and transmit to base station.

Data aggregation uses the parameters of nodes joining the cluster so that the data attributes are selected and stored in an aggregated format for further evaluation and usage. Aggregation refers to the technique that models the data and information in a dimensional construct that is easy to store and retrieve. The data collection technique is being employed to store and collect data items and parameter on a database server. All the related data items are stored inaccessible data form. The problem encountered in the recent past was of the battery power consumption, more efficient data aggregation and collection techniques with right decision making capabilities and mechanism for mentioned problem using data collection for cluster head and data aggregation techniques using data cube aggregation. Aggregation technique like data cube collection approach has been used for storage of node parameter values and cluster location.

5 Comparison Of Algorithms

Let us discuss about the three types of algorithms in which Run Length Encoding better than S-LZW(Sensor-Lempel–Ziv–Welch) and MAS (Minimalist, Adaptive and Streaming compression algorithm)Encoding for compress the data.

5.1 S-LZW

S-LZW (Sensor-LZW) is an adaptation of the popular lossless data compression algorithm LZW . S-LZW follows the same procedure used by the LZW algorithm, but with little restrictions regarding the size of the used data structures. The added restrictions ensure that the requirements of the algorithm are still within the bounds of the available resources in sensor nodes. Before heading into the details of the modification, it is first important to understand why LZW is not suitable for WSN. LZW is a dictionary-based compression algorithm; it works by converting strings of symbols into integer codes. LZW does not use a static dictionary; instead, it builds the dictionary on the fly in a special way to allow both the encoder and the decoder to be able to generate the same dictionary from the input data. First, both the encoder and the decoder initialize the dictionary with 256 entries containing the symbols in the ASCII code. Then the dictionary continues to grow while parsing the input.

The dictionary is the main obstacle preventing LZW from being applicable to WSN. Throughout the compression mechanism, the dictionary keeps growing and can reach sizes much higher than the available RAM on sensor nodes, and this can clearly disrupt the stability of the system. Another problem that LZW faces is that it requires a predefined data volume, i.e. in order for it to start the compression procedure a significant amount of data must already be available. That is why S-LZW can only be used in delay tolerant networks. Several modifications were done on LZW to make it portable to WSNs. Most of these modifications focus on reducing the amount of RAM required for LZW to operate. Here is a list of modifications that gave the birth to S-LZW: S-LZW uses a 512-entrydictionary. As we mentioned before, this dictionary will be initialized with 256 ASCII code symbols. With this size, the dictionary may get full while compressing or decompressing certain datasets. There are two protocols to follow when the dictionary fills, either fix the dictionary to its state whence it get full, or reset the dictionary to the 256 entries. The authors of S-LZW proved that using the fixed protocol produces better results when compressing data of small block sizes (528 bytes). S-LZW divides input data into block sizes of 528 bytes, and then it compresses these blocks individually. It is important to note that S-LZW requires 528 bytes of data to be available in order to compress it, if this amount of data is not available, it has to wait until data accumulate and reach the required size because compressing data of smaller size will be inefficient .

WSN data sampling rate is relatively low and it may take some time to collect 528 bytes of data to be able to start the compression. This is why this algorithm can only be used in delay tolerant networks. The last modification enhances S-LZW by allowing it to benefit from the similarity of data generated by sensor nodes. This is done by adding a mini-cache, which is a hash-indexed dictionary of size N, where N is a power of two, which stores recently used and created dictionary entries. The authors show that it is best to use mini-caches of sizes 32 or 64 dictionary entries.

Duplication free run length coding (DFRLC) solves the problem of run length coding (RLC) .RLC is best when there are lot of redundant bits are there. Number of similar bits are called run. The longer the run the more will be the compression. Here the problem comes when there is redundant bit is very less at that time the size of file or the image will be increase instead of decrease the size of image or file. Let us take an example the sequence of pixels 99999995555555 is encoded by RLC as (9 6 5 7), where 6 and 7 are the runs for the intensity values 9 and 5, respectively . But when a sequence of dissimilar pixels 1 2 3 4 5 will be encoded as (1 1 2 1 3 1 4 1 5 1) by RLC. The size of the image or file will be increased. So this problem will be solved by the DFLRC. In DFLRC an entropy rule-based generative coding method is proposed to construct code words that are capable of encoding both intensity level and different flag values into a single codeword. Hence, there is no need for adding extra byte, dedicating intensity value nor reserving a bit for flag. The proposed algorithm has no duplication problem, and the number of pixels that can be encoded by a single run is infinite.

The two lossless compression algorithm that are used are Improved LZW (ILZW) and Duplication free run length coding (DFRLC).The image will be first compressed with ILZW and the output code generated will be used as the input for the DFRLC. For Example, an image is introduced. Now firstly if the image is a grey scale image, it is sliced into eight binary images and if the image is a colored it will be converted into red, green and blue signals each of which is gray scale image will be then sliced into eight binary images by using bit-plane slicing. The image will be separated into red, green and blue by using color separation or through semantic separation. By this the multiple colors will be reduced in 2 colors black (0) and white (1). Hence an output of code which is in the form of binary value 0 and 1. Then this will be added to the dictionary and each output code in the dictionary associates a frequency counter to phase in binary codes progressively using Adaptive Huffman algorithm to decrease the number of bits . The output produced by the process will be used as the input for DFRLC. The code generated from the ILZW will have runs in its code, this code can be further compressed using DFRLC. In this the output will be taken as input to the DFRLC and codes words will been assigned to the different intensity levels. The codeword assigned to the intensity level will be according to the Probability of Occurrence . Greater the Probability of Occurrence shorter will be the code word. Then the pixels are encoded by using End Bit Independent (EBI) codeword. In this the last bit y is independent no value is assigned to it. It is encoded as a flag to determine the status of next neighbor codeword . There are three types of status S1, S2 and S3. If there is run of three or more pixels then it is assigned y as S3 to indicate that the next codeword encodes the count of pixels in the run. In this End Bit Dependent (EBD) codeword is used to count the runs of the pixels. S2 is used or assigned to y when encoder reads the run of two pixels and for these two pixels single code word has been generated. Here S2 indicating that the encoded pixel intensity value is repeated twice and the next codeword encodes the intensity level of the next pixel(s) .S1 is used where single pixel is read by the encoder and single code word will be generated. Here S1 indicate the next codeword is the intensity level of the next pixel or pixels . By these codeword the number of bits will be reduced and the compressed output generated might be better. Decompression Now, in decompression the final output will be firstly decompressed with DFRLC and a codeword be generated as the output which will be used as input for decompression process for ILZW and after the decompression we will get back the original image without loss the flow of the process.

5.2 MAS COMPRESSIONALGORITHM

MAS stand for Minimalist, Adaptive and Streaming compression algorithm. Minimalist means that it uses the minimum possible amount of resources. Adaptive means it generates variable-size output according to the number of digits in the input, and Streaming means that it does not require buffering of the data before starting the compression process.

MAS is a specialized lossless compression algorithm that only compresses single-precision floating-point data. MAS' implementation does not require any correlation or similarity in the input data, which makes it general and applicable in various domains.

MAS can encode any floating-point number satisfying the following two conditions:

- The number of significant digits should be at most 7, if a floating-point number exceeds 7 Significant digits it would be truncated.
- The floating-point number when put in scientific notation must have a power of 32 or less. Although these conditions mean that some numbers represent able by IEEE standard 754 will not be represent able in MAS. However, these numbers almost do not exist in the data generated by WSN. Numbers that have decimal powers of more than ±32 are almost not found in any application in WSN.

One of the greatest merits of MAS is that it does not require any floating-point operation (addition, subtraction, multiplication, division) to compress floating-point numbers. This is very important because most microcontrollers and processors in sensor platforms are not equipped with an FPU (Floating point unit). The FPU is responsible for carrying out operations on floating point numbers. In the absence of an FPU, these operations are emulated in software but at the cost of time and cycles, which could lead to higher energy consumption. Internally, MAS treats floating-point numbers as strings of characters to carry out the needed operations with a low number of cycles.

5..2.1 MAS Encoding Technique

The first step in encoding a floating-point number is to write it in scientific notation. Scientific notation allows the representation of very small or big numbers with ease. So any number is first written under the following format

(d = digit, e = exponent)$\pm d.dddddd \times 10\pm^{e}$

- Number of significant digits (n) (number of d's): represented on 3 bits because its maximum value is 7.
- The exponent (e): represented on 5 bits because its maximum value is 32.
- Number sign (ns): represented on 1 bit. (0 for positive, 1 for negative).
- Exponent sign (es): represented on 1 bit. (0 for positive, 1 for negative).
- The integer formed by the d's without the decimal point (ddddddd): variable bit size depending on the number of d's(maximum 24 bits). Details are in Table 1.

Table 1.Number of bits needed to represent an integer in binary formed by a certain number of digits.

Number of digits	Number of needed bits
1	4
2	7
3	10
4	14
5	17
6	20
7	24

It is clear that MAS exploits the significant number of digits to represent floating point numbers, for example: -0.0001 is represented on 14 bits while 92301.1 are represented on 30 bits.

Integers form a large part of real numbers, but the above representation may be unfair for integers because there is no need for the exponent and its sign when representing integers. Therefore, a special encoding has been chosen for integers, but there should be a discriminator for the decoder to know which type of number it is going to decode. We exploit the fact that the number of significant digits cannot be zero, and use this as a discriminator between the 2 encodings

- 3 bits that are all zeroes acting as a discriminator.
- Number of significant digits (n) in the integer: 3 bits.
- Number sign (ns): 1 bit.
- The integer: variable bit size following Table 1 (maximum 24 bits).

6 Software Used: MATLAB

- Recent versions of the MATLAB compiler can compile to C ,C++ and binary code, allowing the use of different optimization options for high-speed executable.
- As a consequence, performance of the MATLAB linear-algebraic solvers has been drastically improved in the latest versions of the software, which include also additional accelerators for operations with vector/matrix data types.
- The open architecture allows for every rapid extension of the range of functionality of MATLAB by developing and sharing new toolboxes.
- Using MATLAB tools and toolboxes, it is possible to develop a prototype of an application for a relatively very short time.
- It is easy to exchange toolboxes and parts of code within a team and between teams working in the same area.

6 .1MANIPULATING MATRICES

Entering Matrices

The best way for we to get started with MATLAB is to learn how to handle matrices. Start MATLAB and follow along with each example, We can enter matrices into MATLAB in several different ways:

a) Enter an explicit list of elements.

b) Load matrices from external data files .c & Generate matrices using built-in functions

6.2 EVALUATION

The evaluation metrics of any compression algorithm for WSN are based on the resource limitations of sensor nodes. Thus, we chose the following metrics to evaluate the presented algorithms: compression ratio, processing cost, memory requirements, and energy savings.In the following sub-sections, we will start by describing the chosen platform and the chosen simulators, and then we will move to present the simulation results.

7 RESULTS AND DISCUSSION:SIMULATION RESULTS

7.1CREATION OF NODES

1000 numbers of nodes were created in which the red indicates the transmitted data and the blue indicates nodes that does not receive the datarefer fig 2

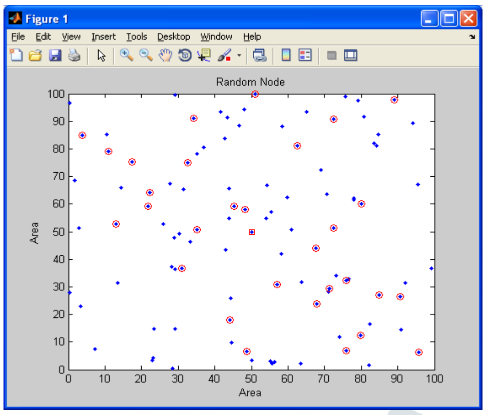


Fig 2 Creation of nodes

7.2CLUSTER HEAD FORMATION

The cluster head can be elected by highly energy efficient node in which the members are elected by 25 cm radius in circular path. As in the fig 3 the red represents the cluster head and blue represents the cluster members

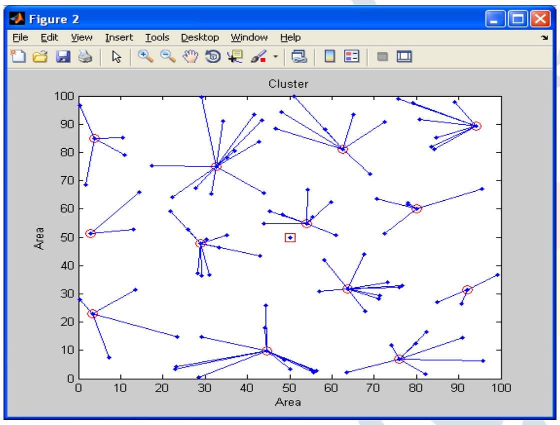


Fig 3 cluster head formation

7.3DELTA COMPRESSION

The output of the delta compressionprocess is given by
Fig 3

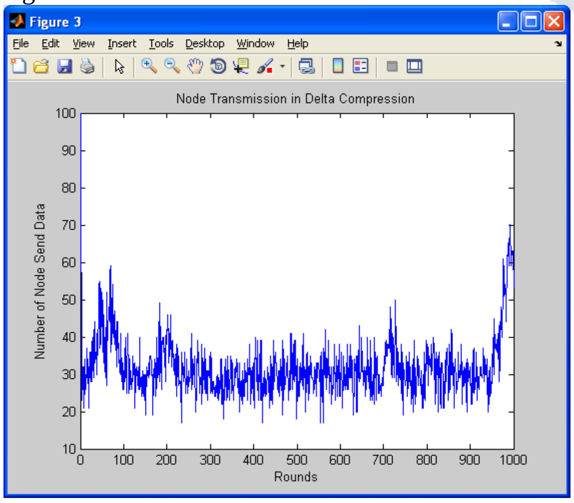


Fig 3 Output of delta compression

7.4DIFFERENCE BETWEEN ACTUAL DATA AND COMPRESSED DATA

- Fig 4 indicates the difference between actual and compressed data .The Blue represents actual data. The RED represents compressed data in delta encoding

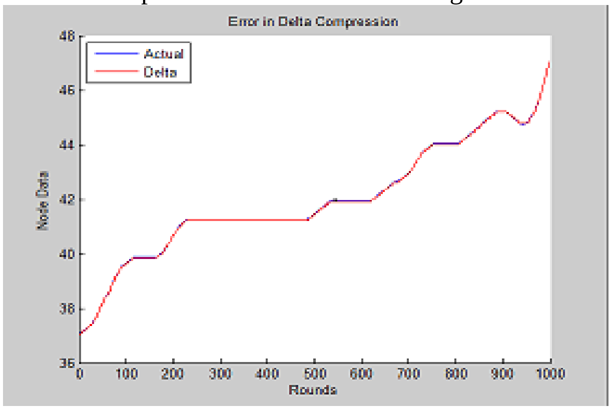


Fig 4 Data output

7.5RUN LENGTH ENCODING

- This algorithm consists of replacing large sequences of repeating data with only one item of this data followed by a counter showing how many times this item is repeated. Fig 5 indicates the actual and RLE compressed data.
- RLE is probably the easiest compression algorithm It replaces sequences of the same data values within a file by a count number and a single value.
- For example ABBBBBBBBBCDEEEEFit can be compressed by A *8B C D *4E F

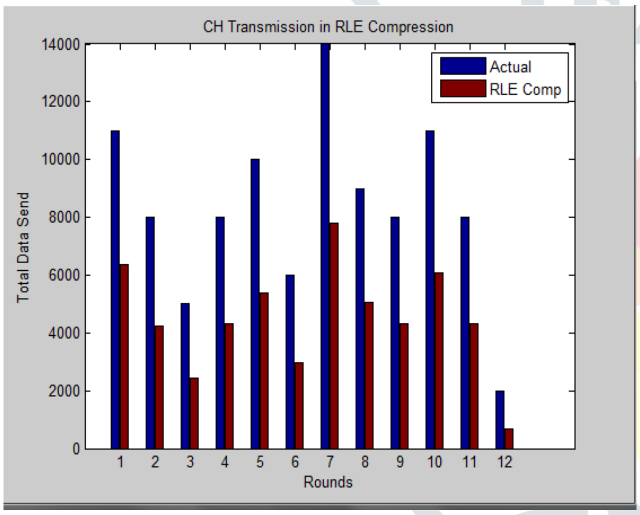


Fig 5 Output of RLE.

7.6 FINAL OUTPUT

The output of the various compression techniques are indicated in Fig6.The delta and RLE compressed data is more efficient than the other methods.

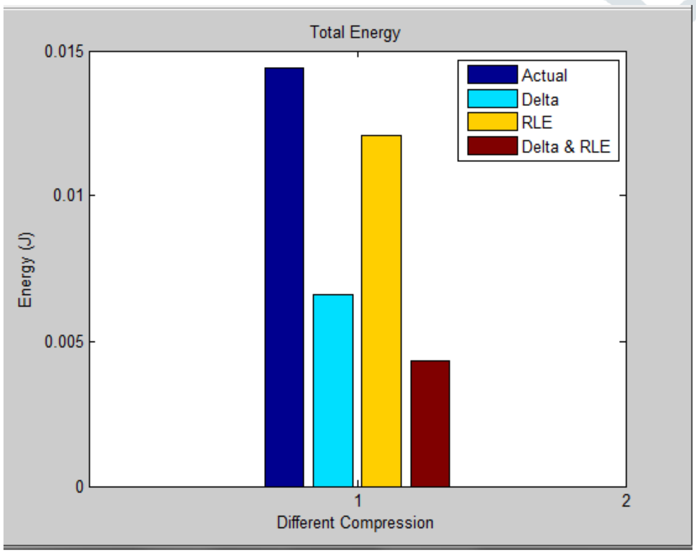
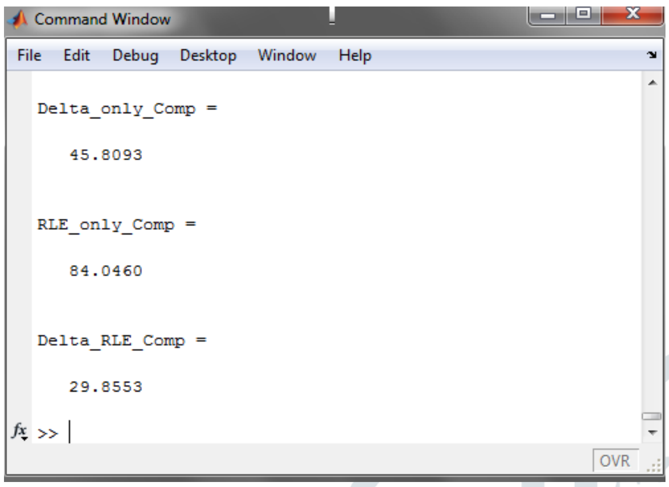


Fig 6 Energy output of the compressed data

DELTA AND RUN LENGTH COMPRESSION

The data values of delta, run length and combined delta –run length compression techniques are displayed in the command window Fig 7



```
Delta_only_Comp =

   45.8093


RLE_only_Comp =

   84.0460


Delta_RLE_Comp =

   29.8553

fx >> |
```

Fig 7 Data output in the command window.

I. Conclusion

The project has been completed successfully and the output results are verified. The results are in line with the expected output. The project has been checked with software testing tools. In this work the software "**MATLAB**" is chosen has been proven to be more appropriate for the intended application. The project is having enough avenues for future enhancement. The project is a prototype model that fulfils all the logical requirements. The project with minimal improvements can be directly applicable for real time applications. Thus the project contributes a significant step forward in the field of "**DOMAIN**", and further paves a road path towards faster developments in the same field. The project is further adaptive towards continuous performance and peripheral up gradations. This work can be applied to variety of industrial and commercial applications.

REFERENCES

1.S.Lee, S. Pattem, M. Sathiamoorthy, B. Krishnamachari, and A. Ortega,(2009) "Spatially-localized compressed sensing and routing in multi-hop sensor networks," in GSN '09,pp.

2.C. Luo, F. Wu, J. Sun, and C. W. Chen, (2010) "Efficient measurement generation and pervasive sparsity for compressive data gathering," IEEE Trans. Wireless Commun., vol. 9, no. 12, pp.3728–3738.

3.J. Luo, L. Xiang, and C. Rosenberg,(2010) ,"Does compressed sensing improve the throughput of wireless sensor networks?" in ICC2010, pp 1–6.

4. J. Wang, S. Tang, B. Yin, and X.-Y. Li, (2012), "Data gathering in wireless sensor networks through intelligent compressive sensing, "in INFOCOM 2012, pp. 603 –611.

5.B. Zhang, X. Cheng, N. Zhang, Y. Cui, Y. Li, and Q. Liang,(2011), "Sparse target counting and localization in sensor networks based on compressive sensing," in INFOCOM 2011, pp. 2255–2263.

6. M. Abu Alsheikh, P. K. Poh, S. Lin, H.-P. Tan, and D. Niyato, "Efficient data compression with error bound guarantee in wireless sensornetworks," inProceedings of the 17th ACM International Conferenceon Modeling, Analysis and Simulation of Wireless and Mobile Systems,.ACM, 2014, pp. 307–311.

7 A sensor networks: A survey and M. Razzaque, C. Bleakley, and S. Dobson, "Compression in wireless comparative evaluation,"ACM Transactions on Sensor Networks, vol. 10, no. 1, p. 5, 2013

8 E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In network aggregationtechniques for wireless sensor networks: A survey,"IEEE WirelessCommunications , vol. 14, no. 2, pp. 70–87, 2007.

9. Y. Liang, "Efficient temporal compression in wireless sensor networks, "inProceedings of the 36th IEEE Conference on Local ComputerNetworks. IEEE, 2011, pp. 466–474

10. M. F. Duarte, G. Shen, A. Ortega, and R. G. Baraniuk, "Signalcompression in wireless sensor networks,"Philosophical Transactionsof the Royal Society Series A, vol. 370, no. 1958, pp. 118–135, 2012.

11 L. Zhao, N. Chen and Y. Jia, An improved energy efficient routing protocol for heterogeneous wireless sensor networks, International Journal of Innovative Computing, Information and Control, vol.13,no.5, pp.1637-1648, 2017.

12 L. A. Villas, A. Boukerche and D. Guidoni, A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks, Ad Hoc Networks, vol.12, no.1, pp.69-85, 2014.

13 Y. Bai, S. Liu, Z. Zhang and J. Yuan, EBTM: An energy-balanced topology method for wireless sensornetworks, International Journal of Innovative Computing, Information and Control, vol.13,no.5, pp.1453-1465, 2017