2018 21st International Conference of Computer and Information Technology (ICCIT), 21-23 December, 2018

A Data Compression and Storage Optimization Framework for IoT Sensor Data in Cloud Storage

Kaium Hossain*, and Shanto Roy†
Department of Computer Science and Engineering*†
Green University of Bangladesh, Dhaka-1207, Bangladesh
Email: hossainkm1234@*gmail.com, shantoroy†@ieee.org

***Abstract*—The paper presents a multi-layered data compression framework that reduces the amount of data before being stored in cloud. At present, Internet of Things (IoT) has gained noticeable attention due to the approaches and advancements towards smart city aspects. With increasing number of devices and sensors connected to the Internet, tremendous amount of data is being generated at every moment which requires volumes of storage space to be stored. However, Data compression techniques can reduce the size of the data and the storage requirement by compressing the data more efficiently. In this article we introduced a two layered compression framework for IoT data that reduces the amount of data with maintaining minimum error rate as well as avoiding bandwidth wastage. In our proposed data compression scheme, we got an initial compression at the fog nodes by 50% compression ratio and in the Cloud storage we have compressed the data up to 90%. We also showed that the error is varied from the original data by 0% to 1.5% after decompression.**

keywords- IoT Data Compression, Efficient data management, Storage Optimization Framework

I. Introduction

IoT is a network paradigm that connects things (e.g. smart phones, smart TV, home appliance, online healthcare [1] etc) to the Internet. In recent years the number of IoT devices increased 31% year-over-year to 8.4 billion in 2017 and it is estimated that there will be 30 billion devices by 2020 [2]. As a result enormous data is continuously being generated from these devices. To store and process these large volume data, IoT is integrated with Cloud Computing which has virtually unlimited storage space and processing capability. Cloud computing eases the work-flow with unlimited resources and helps to build efficient frameworks such as storing a large volume of images for further processing [3]. But the large volume data needs larger amount of storage space and larger amount of energy during transmission through the network. So here comes up the need of data compression to reduce the storage requirement of IoT data. There are two types of data compression techniques- lossless and lossy data compression. In lossless compression techniques, the compressed data can be retrieved exactly to the original data. The LempelZiv(LZ) compression methods are one of the most popular algorithms for lossless compression [4]. On the other hand in lossy compression system, the decompressed data is not exact to the original data and there might be little variation in error rates. For example JPEG image compression is a well known lossy compression technique that works in part by cutting off less important bits from information [5]. IoT data are heterogeneous and different in characteristics considering different environments and needs. To compress these data a two layered lossy data compression approach is introduced that can be used for any kind of IoT environment. This means the data will be compressed in two steps- initially, at the fog node and later in the cloud storage. Fog Computing is a highly virtualized platform that provides computation, storage, and networking services between end devices and traditional Cloud Computing Data Centers, typically, but not exclusively located at the edge of network [6]. As we are compressing data in Fog initially, less energy will be consumed during the data transmission from Fog to Cloud. Because we know that big size of data required more energy during transmission. Another aspect is, due to this compression bandwidth consumption will become less during data transmission from Fog to Cloud. A conceptual diagram of the proposed scheme is shown in **Figure 1**.

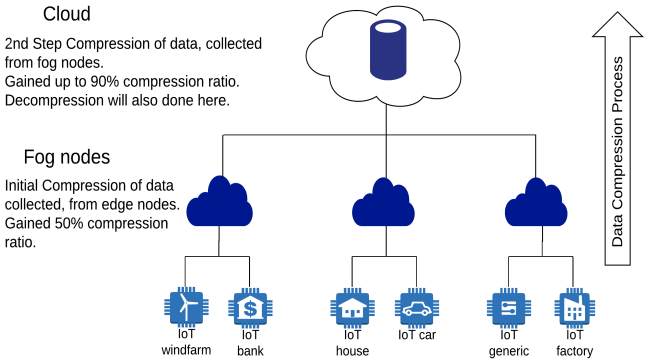

Fig. 1. Basic Block Diagram of proposed Framework

The primary contributions of this work are as follows:

- Designing a compression framework for sensor data.
- Minimizing the error rate in a lossy compression.

The rest of the paper is organized as follows: **Section II** presents an overview, scopes and limitations of previous works related to our proposed framework. **Section III** discusses the overall compression and decompression work-flow. Then **Section IV** describes the implementation process of our proposed work and results thereby. Finally, the significance, scopes, prospects and future works are reviewed in **Section V**, followed by a concluding section.

978-1-5386-9242-4/18/$31.00 © 2018 IEEE

II. Background Overview

A number of research works have been performed regarding the architecture and algorithms that compress various type of IoT data. However, all the proposed works are based on different IoT environment and requirements. Some of them works discussed the compression of multimedia (audio, video, image etc) data; some other work involves string data and others relate numeric data. In addition to that, most of these research works provide efficiency considering different type of IoT data or environment.

[7] proposed a novel lightweight approach capable of alleviating both aspects by leveraging on the advantages offered by classification methods to optimize communications and by enhancing information transmission to simplify data classification. In [8], they have presented a novel data compression algorithm PCVQ based on principal component analysis (PCA) and vector quantization (VQ). In [9], the performance of the data compression schemes is compared among each other, showing the compression capabilities of each of them under different scenarios.

[10] provided an overview of related research issues and challenges in the context of emerging big data compression research. Several points have been addressed, and critically discussed, along with possible research efforts to be considered in the near future. [11] proposed a sensor data compression and power management approach that compresses the sensor data and stops the operation of the highly power consumption sensor when the change of sensor input data is continuously small. They have showed that the wakeup count of the sensor hub core and the amount of sensor data transmitted to the core are reduced by about 78% compared to the conventional sensor data buffering structure. the proposed compression and power management scheme reduced the power consumption by about 56%.

In [12], authors proposed a novel data compression and transmission scheme for power reduction in IoT enabled wireless sensors. Both lossy and lossless techniques meant to be used here to compress data so that it can enable hybrid transmission mode, support adaptive data rate selection and save power in wireless transmission. Therefore the power reduction is demonstrated using a Bluetooth transceiver. It was found to be power reduced to 18% for lossy and 53% for lossless transmission respectively.

[13] proposed a tokenization based model that not only compresses IoT data but also provides data security in cloud storages. To achieve this, authors built an intermediate master node that clusters data for a particular period and then calculates statistical data. Then the data are sent and mapped to distributed storages as token values. The compression basically happens while calculating statistical data and tokenizing using token vaults. In this article authors ensured both security and compression but the process is a bit more complex.

[14] provided a context-sensitive data compression approach that is adaptable to balance accuracy and storage cost. The main idea of this approach is to reduce the size of the videos by discarding the less changeable sequential frames. This approach has achieved the video compression of the data size down to 60% with 96.5% precision for a sampled IoT video streaming application. Here authors made an approach to compress only video data which will not work for any other data. In [15], they obtained the compression ratio (CR) as 50% for the compression method used in the data logger. The memory capacity of the data logger limited size can increase the size by using the method of data compression. Compression method used in this device is lossless. This method can restore the size of data that has been compressed to the size of the original. They have calculated the compression ratio. Time sampling that can be set in data logger is more than 1518 milliseconds. This article has introduced a framework that only works on data logger.

There are a lots of frameworks and architectures on data compression for IoT data for different IoT environment. But most of them are developed for a particular IoT environment. One framework only works for one particular environment. No such common standard for IoT data compression are available as well. A common architecture of data compression is required. Our main concern is to compress IoT numerical data with a better compression ratio and with tolerable error rate. We are using a lossy data compression technique. This architecture can be used in any kind of smart IoT environment.

III. Methodology

The architecture is designed primarily to compress the numerical sensor data of IoT. This data compression architecture is divided into three sub blocks. Firstly the data taking from sensor readings is compressed in the Fog nodes. Then the compressed data is sent to the cloud. Secondly the data coming from the fog nodes is compressed secondly in the cloud. At last the compressed data is decompressed in the cloud before releasing the data on request. The system flowchart is shown respectively in the Figure 2, Figure 3 and Figure 4.

A. Compression in Fog

In the first step of compression, Fog node collects data from the IoT sensor and checks the authenticity of the device. If the device is unauthorized then data packet is discarded. Otherwise the data is stored in an array for a certain period. Further, the data is sorted in ascending order by using a sorting algorithm. Later, a calculation is happened in which the mean values of every two consecutive values of the data set are calculated. So here we are getting one value for every pair of values. Thus the data values are reduced by 50% initially in the Fog which is our primary goal. While calculating, mean values are rounded as to remove the fractional parts of mean values. After doing so, a file is generated by writing the mean values in it. Finally, the file is sent to the cloud. At last the Fog removes the temporary array records from itself. One thing noticeable here if the total frequency of input data set in an odd number then the last value of the data set will remain same after compression, but it will be converted in an integer by using round function.

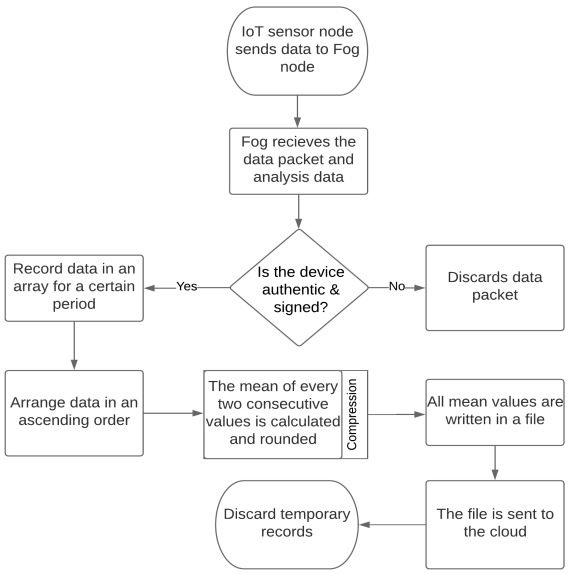

Fig. 2. Compression in Fog

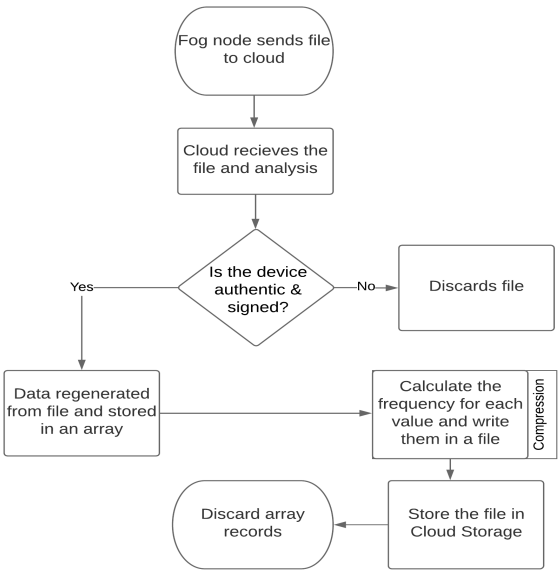

Fig. 3. Compression in Cloud

Algorithm 1: Algorithm for data compression in Fog node

Input: Generic Numerical Sensor Data
Output: Compressed Numerical data

System Initialization;
Receive data packets from end nodes;
if $Device_{ID}$ *and* IoT_{cert} = *True* **then**
  Set $Timeperiod$
  while *data receiving* **do**
    Take data in a certain range;
    Create array for input data set;
    for $Range_{each}$ **do**
      Sort the input data in ascending order;
      while $Timeperiod$ **do**
        Calculate the mean of each pair of values;
        Store mean values in a file;
      Send files towards the cloud storage;
else
  Discards Packet

Algorithm 2: Algorithm for data compression in Cloud

Input: Initially Compressed Sensor data
Output: Compressed Output data

System Initialization;
Receive file from Fog nodes;
if $Device_{ID}$ *and* IoT_{cert} = *True* **then**
  Set $Timeperiod$
  while *data receiving* **do**
    Regenerate data from file;
    Create array for regenerated data;
    while $Timeperiod$ **do**
      Calculate frequencies for each values;
      Store frequencies in a file;
else
  Discards File

B. Compression in Cloud

In this step the Fog sends the file that is initially compressed in the Fog. Cloud takes the file from Fog and regenerate the data values from the file. Then these regenerated values are stored in an array. Here the Cloud calculates the frequencies for all different values and writes them in a file in two columns. For example, if we take 100 values from 1 to 10 then after compression we will get only 10 values of frequencies here. Thus 90% compression ratio is achieved here. After compression the Cloud stores the file until a request is received for this data from the edge devices. The compressed data is decompressed after getting request from the end device. After finishing compression Cloud removes it's temporary records.

C. Decompression Process

In the decompression stage the Cloud at first received the request for data. Then Cloud gets the data file according to the request which contains the data values and their corresponding frequencies. After that all data values are generated in two times of their frequencies. We need to multiply the values by two, as initially we have reduced the data values by 50%. After generating all the values are stored in an array. Then the values are written in a file and sent the file to the request. Thus the decompression process is finished.

As we are compressing the data in a lossy compression technique, there remains some contradiction between the orig-

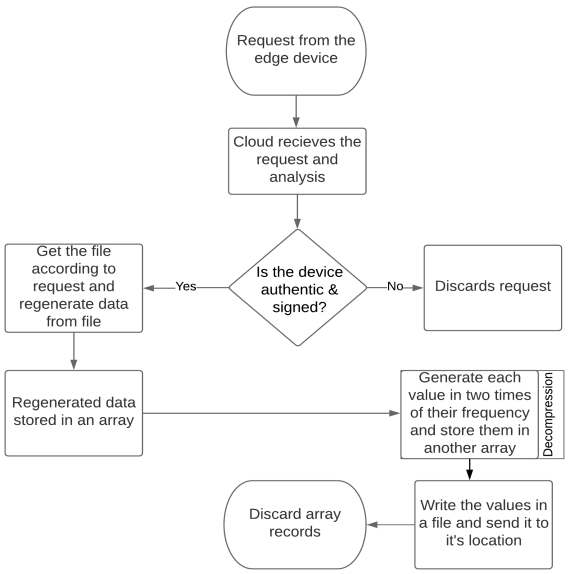

Fig. 4. Decompression Process

Algorithm 3: Algorithm for Decompression

Input: Compressed data
Output: Decompressed Output data

```
System Initialization;
Receive request from end device;
if Device_ID and IoT_cert = True then
    while Request_app do
        Get file according to request;
        Regenerate data from file Create array for
          regenerated data;
        for Range_each do
            Generate each value in two times of their
              frequency;
            Store values in a file;
        Send the file to it's location;
else
    Discards Request;
```

inal data and decompressed data which is called error. We have successful to increase the compression ratio as more as possible by getting as less as error rate possible.

IV. Implementation and Results

To measure the efficiency, we take 100 random values in the range from 1 to 10 by using "rand" function. Here we are dealing with fractional data values. A sample data set is as follows:

Data={7.71621,8.07692,8.44148,9.91319,8.59478,2.39258, 4.55604,3.59341, 8.77747,6.21731,9.80082,1.22775,1.24588, 5.27005,6.48736,7.92088,2.03956,5.90632,5.67143,3.66071, 9.22005,6.31566,4.15659,1.41731,5.24505,8.52637,8.11703, 4.4044,9.78791,5.11016,6.59313,5.69835,5.48764,5.00797, 7.66896,6.86374,4.72747,5.35852,2.57747,9.7294,7.91429, 9.76456,6.55604,2.98022,4.32253,8.87665,5.68571,5.72967, 9.03626,5.7283,3.54863,9.69918,7.98791,1.70082,5.93984, 8.19698,2.67115,7.54396,2.43434,7.53626,8.52445,4.33956, 8.10137,2.03791,9.51676,8.85357,8.11319,6.30027,1.99945, 6.11978,3.5739,8.85632,4.32473,2.72995,1.31511,3.91593, 3.25549,8.68544,1.94808,1.32363,1.73819,3.23104,8.96484, 9.38517,4.083,7.60769,5.15247,7.72637,3.71923,5.06676, 9.13049,3.86126,8.70962,2.1239,5.24121,1.58324,6.61923, 7.40494,3.64011,1.86923}

A. Compression in Fog

According to our framework this given data set is rounded first and later arranged in an ascending order. We are using the most popular Bubble Sort algorithm to sort the data set. After sorting the data set the initial compression is started. Here the program calculates the mean of every pair of data values. After calculating the mean values, all the values is rounded by using the "round" function. Here the initial compression is completed. This process will complete in the Fog node actually. After this compression the data is looking like as follows

Initially Compressed Data = {1,1,2,2,2,2,2,2,3,3,3,4,4,4,4,4, 4,4,5,5,5, 5,5,6,6,6,6,6,6,7,7,7,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,10, 10,10,}

We can see that we have got 50 values after the initial compression.

1) Optimization in Fog: We have taken 100 values from 1 to 10 as a sample data set. And we have got 50 output mean values for the input data set. Therefore, we find out, that we are optimizing the storage almost 50% for only a particular data set initially in the fog node.

$$\textit{Storage Optimization in Fog} = \frac{100-50}{100} * 100\% = 50\%$$

B. Compression in Cloud

For final compression, the program counts the frequencies of each values from the data getting after initial compression. After counting the frequencies it is stored in an array. And finally our compressed data is looked like this

Final Compression = {(1,2),(2,6),(3,3),(4,7),(5,5),(6,6),(7,3), (8,8),(9,7),(10,3)}

Here we have got 10 pair of values where the first value of every pair represents the numbers and second value represents their corresponding frequencies.

1) Optimization in Cloud: The main focus of our approach is to decrease the size of the big data that are generated from the IoT sensor device. In this purpose we are cutting down additional data from the data set and keeping only frequency values.

We have taken 100 values from 1 to 10 as a sample data set. And we have got 10 output frequency values for each data value. Therefore, we find out, that we are optimizing

the storage almost 90% for only a particular data set which exposes the high optimization level of our work.

$$\begin{aligned} \textit{Storage Optimization in Cloud} &= \frac{100-10}{100} * 100\% \\ &= 90\% \end{aligned}$$

That is how we can get 90% compression ratio by using this framework.That's all about the compression process. Now let's go through the decompression process.

C. Decompression Process

In the decompression process the data values is generated two times of their corresponding frequencies. Two times of their frequencies is because as we have taken 100 values for compression, we have to get 100 values after decompression also. So after decompression process we have got a decompressed data set is shown in the following

Decompressed Data = {1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,4,4,4,4, 4,4,4,4,4,4,4,4,4,4,5,5,5,5,5,5,5,5,5,5,6,6,6,6,6,6,6,6,6,6,6,7,7,7,7,7, 7,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,9,9,9,9,9,9,9,10,10,10,10, 10,10}

D. Error Calculation

To calculate the error we have measured the mean of both original data and decompressed data. And by calculating the deviation between the mean we have measured the error rate. Let's assume the mean before mining is x, the mean after mining is y, error rate is E_r.
For data set 1

$$\begin{aligned} \textit{Error Rate, } E_r &= \frac{|x-y|}{x} * 100\% \\ &= \frac{|5.74015-5.8|}{5.74015} * 100\% \\ &= 1.04268\% \\ &\approx 1\% \end{aligned}$$

Thus we have got different error rate for different data sets. Such as we have got error rate for second data set 0.17%, for third data set 0.02%, for fourth data set 1.1%, for fifth data set 0.71% etc.
So, from the upper examples we can conclude that error rate can be minimized up to almost 0% in consideration of best case and maximized to 1.5% in consideration of the worst case.

E. Comparison between original and decompressed data

For the data that we have taken the sample size is 100 and the range is 1-10. The deviation between original data and decompressed data is shown in **Figure 5**.

F. Space Optimization

To show the space optimization between original and compressed data we have taken 50 sample data set of different range and size and applied our scheme for every data set. After that we have counted the byte size for both original and compressed data file. Therefore we have plotted both

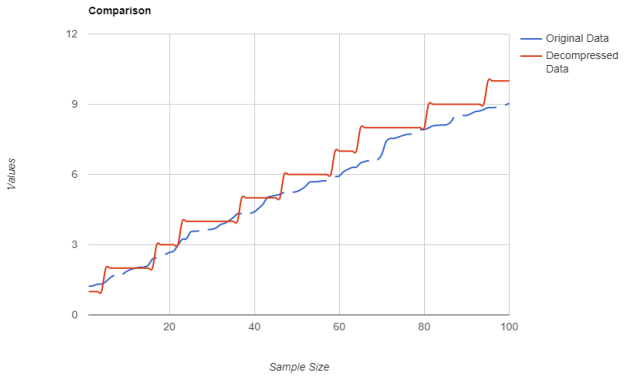

Fig. 5. Deviation between original data and decompressed data

original and compressed data in a graph to show the space optimization of our proposed framework. **Figure 6** shows the space optimization.

Fig. 6. Space Optimization

For original data set 886 bytes of space is required to store in a file for a data sample of size 100.
Our procedure always produce 51 Bytes for the same sample size

$$\begin{aligned} \textit{Space Optimization} &= \frac{886-51}{886} * 100\% \\ &= 94.24\% \end{aligned}$$

G. Comparison With Existing Schemes

Figure 4.5 shows the comparison of our developed system with the existing data compression schemes on the basis of data compression rate. We can see from the figure that [15] have obtained the compression rate of 50% and [11] have reduced the data up to 78% where we have achieved up to 90% compression rate in our approach.

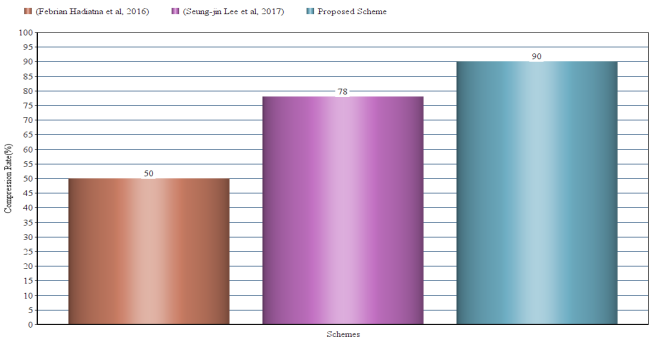

Fig. 7. Comparison With Existing Schemes

V. Discussion

A. Significance of the work

1) ***Storage Optimization:*** For increasing volume of sensor data, storage capability is one of the major concern. By our scheme we are able to reduce the storage requirement of IoT numerical data by 90%.
2) ***Easy to Implement:*** The framework that we have developed is easier to implement. Because it is developed by using several easy computation methods.
3) ***Error Rate:*** Generally if we want to increase the compression ratio, error rate increases accordingly. But in this approach, our error rate is at best 1% for certain range of values that is comparatively little compared to the compression ratio.
4) ***Energy Consumption:*** It is an another big issue for big data. As we initially compress the data in the Fog node by half of the original data, we have got a benefit of less energy consumption. We think that it will reduce the energy consumption by from 45% to 50%.

B. Scope and Limitations

The primary limitation of our data compression approach is that it can compress only the numerical data. Again, it is applicable for smart systems where it can tolerate the errors that occur during compression. If the data value range increases then error rate may vary. But still it is acceptable compared to the optimization in cloud storage.

C. Future Work

In future we plan to implement it in real fog and test with the real time data. The system can be combined with any other lossless compression for a better performance. It can be also combined with a security framework for better secure transmission of data.

VI. Conclusion

Big data issue seems terrible with the increasing number of IoT devices. Present research works are less advanced compared to the need. Therefore, our proposed scheme compresses the IoT numerical data with a better compression ratio as well as lower error rate. Here, we used a lossy compression technique to mine the IoT data in cloud storage. With initial compression in Fog node, we succeed to reduce energy consumption and bandwidth wastage as well. We achieved on around 90% compression ratio with around 1% error rate that is minimum and indicates the efficiency of the process while compressing the homogeneous structured data created by IoT sensor networks where approximate values are needed for further mining. In this regard, a better storage optimization, less energy consumption and less bandwidth wastage can lead towards an efficient management of smart city aspects.

Acknowledgements

This work has been partially supported by Green University of Bangladesh research fund.

References

[1] S. Roy, A. Rahman, M. Helal, M. S. Kaiser, and Z. I. Chowdhury, "Low cost rf based online patient monitoring using web and mobile applications," in *Informatics, Electronics and Vision (ICIEV), 2016 5th International Conference on*. IEEE, 2016, pp. 869–874.
[2] A. Nordrum, "Popular internet of things forecast of 50 billion devices by 2020 is outdated," *IEEE Spectrum*, vol. 18, 2016.
[3] A. R. Shovon, S. Roy, T. Sharma, and M. Whaiduzzaman, "A restful e-governance application framework for people identity verification in cloud," in *International Conference on Cloud Computing*. Springer, 2018, pp. 281–294.
[4] R. Naqvi, R. Riaz, and F. Siddiqui, "Optimized rtl design and implementation of lzw algorithm for high bandwidth applications," *Electrical Review*, vol. 4, pp. 279–285, 2011.
[5] C. Arcangel, "On compression," *Retrieved 6 March 2013*, 2013.
[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.
[7] M. Danieletto, N. Bui, and M. Zorzi, "Improving internet of things communications through compression and classification," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*. IEEE, 2012, pp. 284–289.
[8] Y. Li, S. Xi, H. Wei, Z. Zhang, and C. Zhang, "A data compression algorithm for the sea route monitoring with wireless sensor network," in *Information Science and Cloud Computing (ISCC), 2013 International Conference on*. IEEE, 2013, pp. 153–159.
[9] S. A. Awwad, C. K. Ng, N. K. Noordin, B. M. Ali, and F. Hashim, "Second and subsequent fragments headers compression scheme for ipv6 header in 6lowpan network," in *Sensing Technology (ICST), 2013 Seventh International Conference on*. IEEE, 2013, pp. 771–776.
[10] A. Cuzzocrea, "Big data compression paradigms for supporting efficient and scalable data-intensive iot frameworks," in *Proceedings of the Sixth International Conference on Emerging Databases: Technologies, Applications, and Theory*. ACM, 2016, pp. 67–71.
[11] S.-j. Lee, M. Kee, and G.-H. Park, "Sensor data compression and power management scheme for low power sensor hub," *IEICE Electronics Express*, vol. 14, no. 23, pp. 20 170 974–20 170 974, 2017.
[12] C. J. Deepu, C.-H. Heng, and Y. Lian, "A hybrid data compression scheme for power reduction in wireless sensors for iot," *IEEE transactions on biomedical circuits and systems*, vol. 11, no. 2, pp. 245–254, 2017.
[13] S. Roy, A. R. Shovon, and M. Whaiduzzaman, "Combined approach of tokenization and mining to secure and optimize big data in cloud storage," in *Humanitarian Technology Conference (R10-HTC), 2017 IEEE Region 10*. IEEE, 2017, pp. 83–88.
[14] C.-C. Hsu, Y.-T. Fang, and F. Yu, "Content-sensitive data compression for iot streaming services," in *Internet of Things (ICIOT), 2017 IEEE International Congress on*. IEEE, 2017, pp. 147–150.
[15] F. Hadiatna, H. Hindersah, D. Yolanda, and M. A. Triawan, "Design and implementation of data logger using lossless data compression method for internet of things," in *System Engineering and Technology (ICSET), 2016 6th International Conference on*. IEEE, 2016, pp. 105–108.