

Design an efficient internet of things data compression for healthcare applications

Ahmed Najah Kadhim, Mehdi Ebady Manaa

Department of Information Network, College of Information Technology, University of Babylon, Al Hillah, Iraq

Article Info

Article history:

Received Mar 4, 2022

Revised May 9, 2022

Accepted May 17, 2022

Keywords:

Data compression

Fog computing

IoT

Raspberry Pi

Zstandard algorithm

ABSTRACT

The internet of things (IoT) is an ecosystem of connected objects that are accessible and available through the internet. This "thing" in the IoT could be a sensor such as a heart monitor, temperature, and oxygen rate in the blood. These sensors produce huge amounts of information that lead to congestion and an effect on bandwidth in the IoT network. In this paper, the proposed system is based on the Zstandard compression algorithm to compress the sensor data to minimize the amount of data transmitted from the IoT level to the fog level and decrease network overloading. The proposed system was evaluated using compression ratio, throughput, and latency time for healthcare applications. The result showed better calculation through decreased response time and increased throughput for transmitted data compared with the case of non-compressed data. It showed the compression data ratio about 70% of original data, maximum number of IoT sensor reads as 100, throughput is 85.43 B/ms, and fog processing delay is 6.25 ms.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Ahmed Najah Kadhim

Department of Information Network, College of Information Technology, University of Babylon

60th Street, Al Hillah, Babylon, Iraq

Email: ahmedn.net.msc@student.uobabylon.edu.iq

1. INTRODUCTION

The internet of things (IoT) is a platform where virtual and physical goods are connected and interact with one another, according to its most basic description. The internet of things make use of a variety of technologies, including cloud computing, wireless sensor networks, and embedded intelligence, to accomplish its goals [1], [2]. Environmental data is collected by internet of things devices via the use of radio frequency identifier (RFID), cameras, sensors, and other technologies [3], [4]. Some of the complex capabilities offered with these systems include remote monitoring, online analytics, and remote management. Remote monitoring applications based on the IoT are being utilized in several disciplines, ranging from healthcare to smart manufacturing, as well as smart homes, smart cities, smart agriculture, and among other areas [5], [6].

Fog computing is an expanded architecture of cloud computing that leverages edge devices to provide computation, storage, analysis, and communication between smart end devices and cloud computing data centers. As a result, processing and intelligence are brought closer to the point of origin [7]. Fog computing is becoming more important for IoT because of its ability to provide quick response and low latency between the cloud and IoT levels. Fog computing may be comprised of a large number of nodes with a widely dispersed geographical distribution and it is distinguished by its mobility and spatial awareness [8].

Data compression is a critical component of contemporary digital communication; it reduces the number of bits required to encode data from its original size [9], [10]. Data transmission is greatly reduced at

the source end and then recovery is assisted by this method. Using this method, a significant amount of energy and bandwidth is saved [11], [12]. There are many ways to compress data, but they may be broadly categorized into two types: lossless techniques and lossy methods. By finding and removing statistical redundancy, lossless compression minimizes the number of bits needed to encode a given file size. Lossless compression ensures that no data is lost. By deleting unneeded or less essential data, lossy compression decreases the number of bits [13], [14]. IoT environment produce huge amounts of information and IoT networks do not have sufficient bandwidth due to constrained resources [15]. To save IoT network bandwidth and prevent network congestion issues, and network overhead, so the IoT data must be compressed prior to transmission to decrease size of transmitted data [16], [17].

To meet the latency, traffic reduction, privacy, and bandwidth requirements of IoT applications, the IoT network must be able to respond quickly to data from these apps. The smart healthcare application, for example, requires strong security and privacy for patient data, a rapid reaction to emergency status, and high bandwidth for uploading the enormous quantity of daily sensed patient data over the IoT network, among other requirements [18]. The present IoT network is unable to meet these demands due to bandwidth restrictions, an unavoidable lengthy delay, and the high cost of uploading such a large volume of information [19]. Therefore, the proposed system works in an efficient way to minimize the amount of data transmitted from the IoT level to the fog level and decrease network overloading using one of data compression algorithm in real experiment.

There are different related works associated with the IoT healthcare data compression as follow: a novel lossless electroencephalographic (EEG) data compression strategy has been developed in [20], it is enabled by fog computing. The method is being developed to decrease the amount of EEG patient data that is received at the fog gateway before it is sent to a cloud platform. The EEG data compression technique is divided into two phases: clustering and compression. First, using agglomerative hierarchical clustering, the incoming data is organized into groups of similar data. In the second step, the Huffman encoding is applied to each of the clusters that have been generated. The compressed data from smaller clusters are concatenated and sent from fog to cloud at the end of the process. Using hierarchical clustering and Huffman encoding (HCHE), the researchers discovered that the proposed strategy performed much better in terms of EEG data reduction size and compression ratio by taking advantage of similarities between data from different clusters.

Management of data traffic has been proposed in [21] for compression and minimum description length (MDL). The goal of the proposed is to reduce the energy consumption of each node and prolong the wireless sensor network's (WSN's) lifespan while maintaining acceptable data accuracy. In the first place, a lightweight lossless compression method based on differential encoding and Huffman techniques are used, which is especially useful for IoT nodes that monitor the environment and have limited computational and memory resources. A hierarchical clustering approach was used in the second level of MDL to cluster datasets that had been collected in the first level. The performance of the sensors is evaluated based on the real data collected by the sensors and the use of the objective modular network testbed in C++ (OMNeT++) simulator. In terms of results, a reduction in the volume of data submitted to the GW of up to 79-93.3 percent was achieved, as was a reduction in the energy expended of up to 76-82 percent. A compression ratio of up to 87-93 percent was achieved.

The compression-based data reduction (CBDR) approach has been proposed in [22] to operate at the level of internet of things sensor nodes. The approach, which is being developed to reduce data transmissions in the internet of things sensor networks, has the potential to save a significant amount of energy and prolong its lifetime. It incorporates two phases of compression: a lossy symbolic aggregate approximation (SAX) quantization stage that minimizes the dynamic range of sensor data readings and a lossless Lempel–Ziv–Welch (LZW) compression step that compresses the output of the loss quantization stage. Another modification to the CBDR approach has been proposed, which is the addition of a dynamic transmission component (DT-CBDR). It is utilizing the OMNeT++ simulator in conjunction with actual sensory data collected at Intel Lab. Its findings revealed that CBDR and DT-CBDR protocols outperformed PFF and Harb protocols in terms of workload reduction, with up to 79 percent and 80 percent reductions in the amount of data collected, 74 percent to 80 percent reductions in the amount of data transmitted, and 78 percent reduction in the amount of energy used, while CBDR and DT-CBDR techniques achieve high compression ratios (above 95 percent).

Passos *et al.* [23] proposed energy-efficient local data compression for transmission over wireless body area networks (WBAN) (GROWN) as a hybrid technique to local data compression. The technology is being developed to reduce information redundancy during data transmission, conserve energy, and extend the lifespan of wearable devices in wireless body networks, among other things. It is called GROWN because it coordinates compression with the kind of data signal gathered, minimizing redundant data transmission and so lowering energy consumption by devices in the network. A threshold is used to restrict the allowable data losses that may occur during transmissions and lossless compression is used by encoding tables, which normalize the data

format to reduce how much information is sent over a network connection. GROWN has achieved an energy efficiency of up to 53.73 percent with a maximum delay of 55 milliseconds, according to the company.

Hossain and Roy [24] proposed a two-layered data compression architecture for the internet of things data is presented, which decreases the quantity of data transmitted while maintaining a low error rate and avoiding bandwidth wasting. A lossy compression framework has been developed for sensor data and attempts have been made to reduce the error rate in the lossy compression process. A pair of algorithms have been developed that operate primarily at two layers: the fog and the cloud nodes. A collection of IoT sensor data is collected by the system, which then compresses and saves it in the cloud. When the user requests the data, it decompresses and performs specific operations to dispense the right information to him or her. The results show that at the fog nodes, the data was compressed by 50%. In cloud storage, we have compressed the data up to 90%. Even after decompression, we found that there is a small difference between the original data and the data after it has been uncompressed.

Shrividhiya *et al.* [25] propose hybrid compression, which is named as “data compression algorithm utilizing LZW framework based on Huffman technique to offer better compression ratio. The Huffman algorithm is used to compress the data, and the output of this process is transferred to the LZW algorithm, which produces the final result (Huffman-based LZW for data compression). The final compressed ratio and saving for all of the input data is determined at the conclusion of two compressions. The output shows five distinct inputs with varying storage capacities ranging from a few bytes to 5 kilobytes. This hybrid technique achieves a compression ratio of 2.417, a compression factor of 0.425, and a saving percentage of 57.7 percent as a result. The findings were compared to the well-known LZW and Huffman algorithms, and the results revealed that the results were more efficient and less susceptible to mistakes than these two algorithms. Furthermore, this hybrid method performs better when dealing with large amounts of data.

In addition to the introduced of this paper, the used system based on a real-time compression method for internet of things data gathered by many IoT devices using fog computing has been adopted. The Zstandard compression algorithm is used for IoT sensor data. It will compress data and send to the fog for decompression. It evaluated using compression ratio, throughput, and latency time for healthcare applications. The paper can be outlined in the following way: 1) introduction, 2) the proposed approach, 3) method, 4) results and discussion, and 5) conclusion.

2. THE PROPOSED APPROACH

The proposed approach is based on the Zstandard algorithm. It is mainly used for compression purposes in the proposed system. It works at the IoT sensors level to compress their readings in an efficient way to minimize the amount of data transmitted from the IoT level to the fog level and decrease network overloading. The general view of the proposed system showed in Figure 1. The flowchart of the system is explained in Figure 2. The main steps of the proposed approach are as shown in:

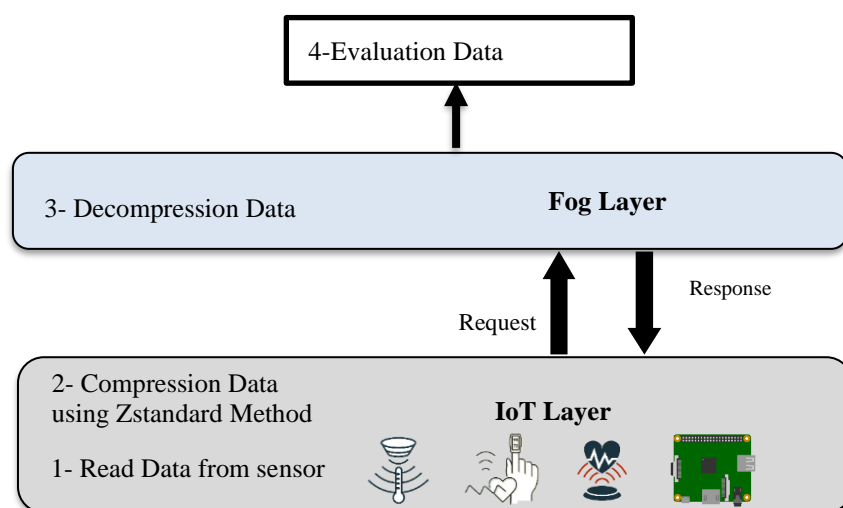


Figure 1. The general view of the proposed system

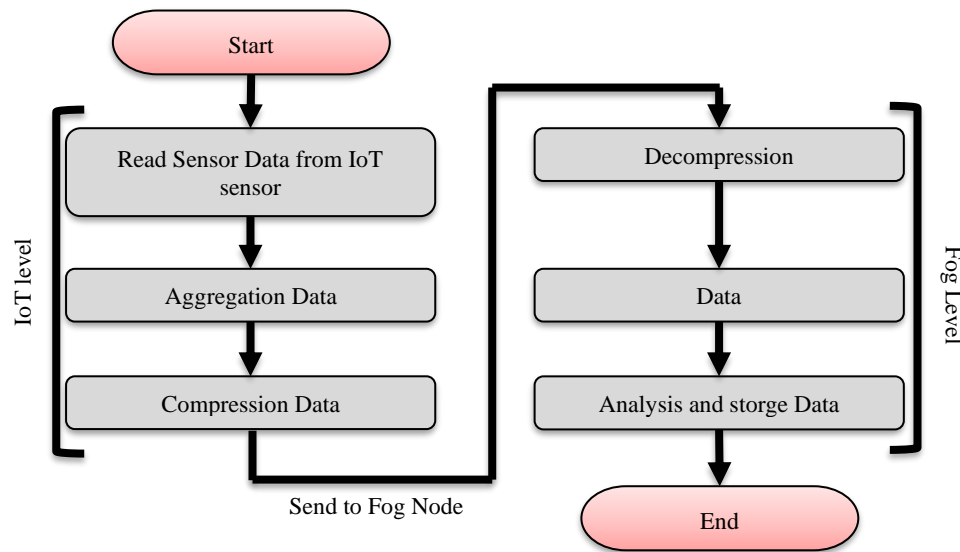


Figure 2. The flowchart of the proposed system

2.1. System implementation steps

The implementation steps of the proposed approach are sorted as shown in:

a. Read sensor data

Initializing the sensors (temperature, heart rate, oxygen level) read data from the patient's body. The Raspberry Pi (RPi) received data from sensors, and aggregation sensing information as data messages from IoT sensors in the Raspberry Pi such as temperature, heart rate, and oxygen level sensors.

b. Compression phase (IoT layer using Zstandard algorithm)

The compression process in Raspberry Pi compresses the data collected by sensors using Zstandard algorithm to result in compression data (CD). The compression data on the Raspberry Pi will be sent through a wireless connection by the socket (IP address) to the server, as shown in Figure 3 implementation of the proposed system.

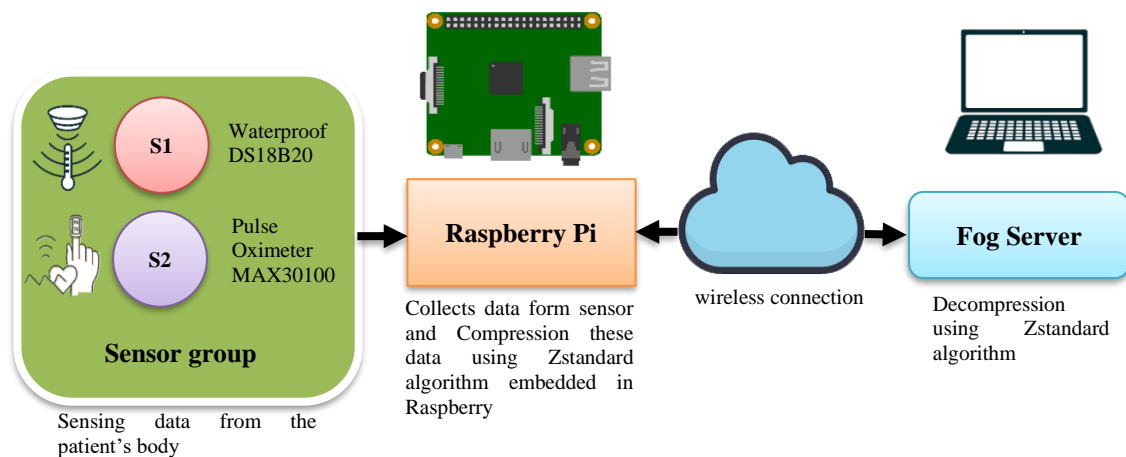


Figure 3. Implementation of the proposed compression algorithm

c. Decompression phase (fog layer using Zstandard algorithm)

The server will receive a CD form the Raspberry Pi. It will run decompression process to decompress the CD by using a Zstandard algorithm that returns data message to its original form. The data is the analysis and store the data in the server.

2.2. System specifications

The proposed system components include the IoT devices as Raspberry Pi, a temperature sensor, a heart rate sensor, an oxygen sensor of the network data devices, and the management device as a fog server (computer). Table 1 contains information on the specifications of the used network devices. The proposed system is based on the Raspberry Pi as a fundamental construction component due to its characteristics: it is tiny, inexpensive, powerful, and completely adaptable. In addition, it's a programmable tiny computer board that comes pre-loaded with support for a wide variety of input and output ports, as well as a network connection. It is used in the proposed system for monitoring and controlling sensing devices. It has the process of compression of data gathered by IoT sensors.

Besides, the temperature sensor in the proposed system senses the temperature between -55 and +125 degrees. Furthermore, the heart rate sensor is used for sensing heart pluses and capturing them as data messages. Alongside, the oximeter sensor is used to sense the rate of oxygen and heart rate in the human body, in the last the fog server device is used for monitoring and controlling the entire network traffic. Furthermore, all sensing information from IoT sensors (temperature degree, heart rate, and oxygen level) is collected on the Raspberry Pi, which then compresses this sensing information and sends it to the server.

The proposed system implemented with Python programming language, it is used on both the Raspberry Pi and the server devices. Python is a computer language that is useful for a wide range of tasks and is both adaptable and powerful. Because it is succinct and simple to understand, it makes an excellent for fog computing implementations. Python can do any task you set for it. Python is a programming language that may be used for a variety of tasks, including resource management, data analysis, data compression, and different approaches for network data analysis.

Table 1. Specification of devices nodes

Device type	Device model	Supply voltage	Product size
Raspberry Pi	4 model B 8 GB	DC 5 V/2.5 A	82 mm x 56 mm x 19.5 mm, 50 g
Temperature sensor	DS18B20 waterproof digital thermal probe 1 m	3.0 V to 5.5 V	6*50 mm
A sensor of oxygen and heart rate	Pulse Oximeter (SPO2) sensor MAX30100	3.3 V	5.6 mm x 2.8 mm x 1.2 mm
Fog server	Dell Inspiron 5559	19.5 V	

2.3. System evaluation metric

The proposed system is evaluated with the main metrics as shown in:

- Data compression ratio (DCR): it is used in the proposed system to for measured how much data representation can be compressed before being transferred. The most common way to describe this is as the ratio of compressed to uncompressed data size.

$$DCR (\%) = \left(\frac{\text{size of data} - \text{size of Compressed data}}{\text{size of data}} \right) * 100\% \quad (1)$$

- Throughput: it is used in the proposed system to measure maximum quantity of data that can be sent from source to destination in a particular length of time through an internet connection.

$$Throughput = \left(\frac{\text{Number of send data}}{\text{Time}} \right) \quad (2)$$

- Latency (delay): it is used in the proposed system to measurement of the amount of time it takes for data to travel across a network to its destination. As a result, latency has a significant influence on the overall performance of the network. It is customary to quantify latency in milliseconds (ms).

3. METHOD

This algorithm is kind of a lossless compression algorithm; it ensures that all data is sent without any loss. It is a high-compression-ratio method that provides both quick and efficient features. It also has a particular mode for small amounts of data, known as "dictionary compression," that may be used. There is a very broad range of speed/compression trade-offs available in the reference library, which is supported by a very fast decoder. Algorithm 1 describes the process of compressing data readings. Besides the decompression messages process showed in Algorithm 2.

Algorithm 1. Zstandard algorithm compression

```

Initialize table with single character strings
P = first input character
WHILE not end of input stream
    C = next input character
    IF P + C is in the string table
        P = P + C
    ELSE
        output the code for P
        add P + C to the string table
        P = C
    END WHILE
output code for P

```

Algorithm 2. Zstandard algorithm decompression

```

Initialize table with single character strings
OLD = first input code
output translation of OLD
WHILE not end of input stream
    NEW = next input code
    IF NEW is not in the string table
        S = translation of OLD
        S = S + C
    ELSE
        S = translation of NEW
    output S
    C = first character of S
    OLD + C to the string table
    OLD = NEW
END WHILE

```

4. RESULTS AND DISCUSSION

The proposed fog computing system result is based on the main network structure, as it analyzed between the IoT, and fog layers of the fog computing. Furthermore, the proposed system results are based on the two case studies as the first case study is based on the without compressed data while the second one is based on the case of compressed data with the Zstandard algorithm.

4.1. The 1st case study of (without compressed) method

In this case, the proposed system was implemented with Python programming language to evaluate the fog network performance based on the throughput parameter as shown in Table 2. The 1st case study with number of IoT sensor reads as 20, 50, and 100 with different input data sizes as 273 B, 671 B, and 1338 B respectively. It showed that the throughput increased with increased number of reads. Table 3 shows the fog processing delay for each IoT sensor reads. Figure 4 shows the 1st case study among evaluation metrics.

Table 2. Throughput of the 1st case study

Number of IoT sensor reads	Data size (Byte)	Execution's time (msec)	Throughput (Byte/msec)
20	273	12.28	22.23
50	671	18.77	35.74
100	1338	39.45	33.91

Table 3. Latency of the 1st case study

Number of IoT sensor reads	Fog processing delay (msec)
20	6.25
50	7.81
100	9.37

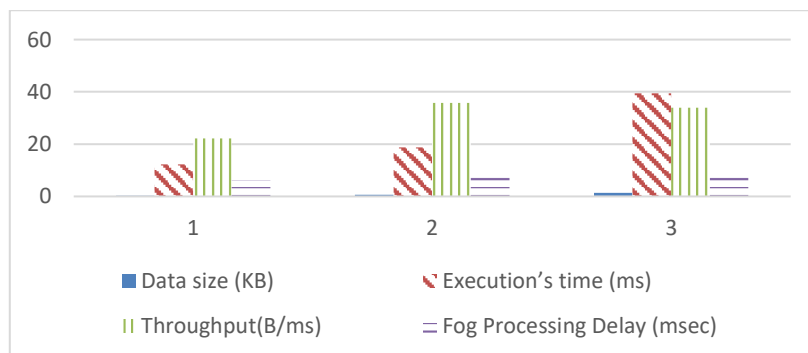


Figure 4. System parameters for the 1st case study

4.2. The 2nd case study of (with compression) method

In this case study the proposed system is based on the Zstandard compression algorithm to compressed input sensor data due to the IoT limitation device and fewer resources so the overloaded in the network will affect directly on the network performance, so this algorithm decrease data size and network delay through transmitting data from the source IoT sensors to the final destination (data management server). Table 4 shows the data compression ratio of the 2nd case study with a different number of reads. Table 5 showed the throughput value of the 2nd case study based on the compressed sensor data. In addition, Table 6 showed the latency of the data processing in the fog system is calculated depending on the different input reads.

Table 4. Data compression ratio of the 2nd case study

Number of IoT sensor reads	Size of original data (Byte)	Size of compression data (Byte)	DCR
20	273	115	57%
50	671	219	67%
100	1338	393	70%

Table 5. Throughput of the 2nd case study

Number of IoT sensor reads	Data size (Byte)	Execution's time (ms)	Throughput (Byte/ms)
20	115	3.9 ms	29.48
50	219	4.3 ms	50.93
100	393	4.6 ms	85.43

Table 6. Latency of the 2nd case study

Number of IoT sensor reads	Fog processing delay (ms)
20	6.25 ms
50	6.25 ms
100	6.25 ms

Figure 5 showed the 2nd case study among evaluation metrics. The Zstandard compression algorithm used in the proposed system due to the flexibility of use, easy to setting, and less overhead, as well as less computation power (CPU, RAM, and execution time) resources to executed and it was the better choice to work in IoT resource limitation environment compared with other compressed algorithms such as LWZ algorithm which effects on the system resource through executing processes. It showed decrease response time and increased throughput for transmitted data compared with the case of non-compressed data.

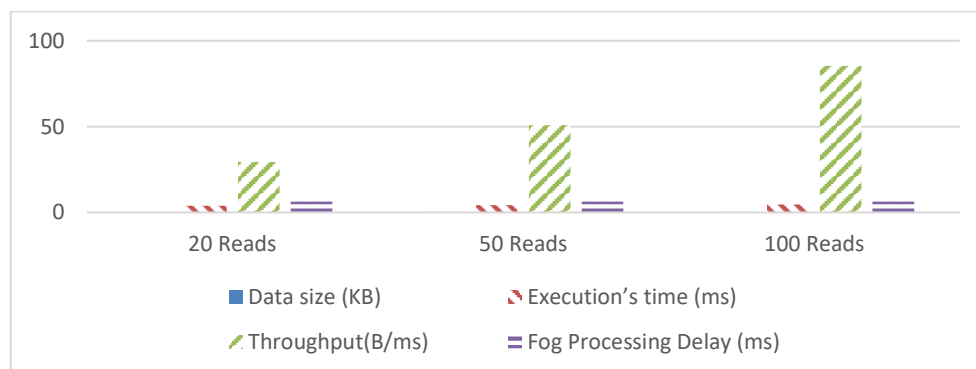


Figure 5. System parameters for the 2nd case study

4.3. The 3rd case study of (system comparison)

The proposed system compared with different related work based on the compression of IoT sensing data and it showed that the proposed system provides better results in compressed data size and ratio value, as compression data size is 881 B, and the compressed ratio is 70.6 B. Table 7 showed comparison with other related work.

Table 7. The comparison with other related work

Refrences.No	Programming language	Algorithm	Original data size in bytes	Compression data size in bytes	Ratio
2021 [25]	Python	Huffman based LZW	3000	1072	64%
The proposed system	Python	Zstandard	3000	881	70%




5. CONCLUSION

By using the Zstandard compression technique, the suggested system may reduce the quantity of data sent from the IoT device to the fog server significantly. The hardware used in this proposal is the Raspberry Pi 4 model B and two types of sensors: DS18B20 (temperature sensor-waterproof) and MAX30102 (pulse oximeter and heart-rate sensor). These sensors read information and send data to the Raspberry Pi. The programming language used in the paper is Python on the server and IoT devices used the Zstandard compression algorithm in the proposed system due to the flexibility of use, ease of setting, and less overhead, as well as fewer computation power (CPU, RAM, and execution time) resources required to execute. The proposed system was evaluated using compression ratio, throughput, and latency time for healthcare applications. The compression ratio is set when a file size of 3000 bytes is compressed to 881 bytes at a rate of 70%, which provides better results compared with different related work. For further study, we suggest combine data compression with cryptographic for more security and speed in transmitted data.




REFERENCES

- [1] M. Al-Qurabat and A. Kadhun, "A Lightweight Huffman-based Differential Encoding Lossless Compression Technique in IoT for Smart Agriculture," *International Journal of Computing and Digital System*, vol. 11, no. 1, pp. 117–127, Jan. 2022, doi: 10.12785/ijcds/110109.
- [2] I. Sokol and P. Hubinský, "Internet of things-nonstandard data compression," *Journal of Electrical Engineering*, vol. 71, no. 4, pp. 281–285, 2020, doi: 10.2478/jee-2020-0038.
- [3] J. Azar, A. Makhoul, M. Barhamgi, and R. Couturier, "An energy efficient IoT data compression approach for edge machine learning," *Future Generation Computer Systems*, vol. 96, pp. 168–175, Jul. 2019, doi: 10.1016/j.future.2019.02.005.
- [4] S. A. Abdulzahra, A. K. M. Al-Qurabat, and A. K. Idrees, "Data Reduction Based on Compression Technique for Big Data in IoT," *2020 International Conference on Emerging Smart Computing and Informatics*, 2020, pp. 103–108, doi: 10.1109/ESCI48226.2020.9167636.
- [5] 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," *2016 6th International Conference on System Engineering and Technology (ICSET)*, 2016, pp. 105–108, doi: 10.1109/ICSEngT.2016.7849632.
- [6] J. Park, H. Park, and Y. -J. Choi, "Data compression and prediction using machine learning for industrial IoT," *2018 International Conference on Information Networking (ICOIN)*, 2018, pp. 818–820, doi: 10.1109/ICOIN.2018.8343232.
- [7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," *MCC '12: Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, Aug. 2012, doi: 10.1145/2342509.2342513.
- [8] H. F. Atlam, R. J. Walters, and G. B. Wills, "Fog computing and the internet of things: A review," *big data and cognitive computing*, vol. 2, no. 2, pp. 1–18, Apr. 2018, doi: 10.3390/bdcc2020010.
- [9] S. Hamdan, A. Awaian, and S. Almajali, "Compression Techniques Used in Iot: A Comparitive Study," *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, 2019, pp. 1–5, doi: 10.1109/ICTCS.2019.8923112.
- [10] A. P. Sridhar, "An Efficient Lossless Medical Data Compression using LZW compression for Optimal Cloud Data Storage," *Annals of the Romanian Society for Cell Biology*, vol. 25, no. 6, pp. 17144–17160, May 2021.
- [11] T. N. Gia, L. Qingqing, J. P. Queralta, H. Tenhunen, Z. Zou, and T. Westerlund, "Lossless Compression Techniques in Edge Computing for Mission-Critical Applications in the IoT," *2019 Twelfth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, 2019, pp. 1–2, doi: 10.23919/ICMU48249.2019.9006647.
- [12] S. K. Routray, A. Javali, A. Sahoo, W. Semunigus, and M. Pappa, "Lossless compression techniques for low bandwidth io ts," *Proc. 4th Int. Conf. IoT Soc. Mobile, Anal. Cloud, ISMAC 2020*, pp. 177–181, 2020, doi: 10.1109/I-SMAC49090.2020.9243457.
- [13] B. Vijayalakshmi and N. Sasirekha, "Comparative Analysis of Lossless Text Compression Methods with Novel Tamil Compression Technique," *International Journal of Research in Engineering and Science (IJRES)*, vol. 9, no. 7, pp. 38–44, 2021.
- [14] G. Signoretti, M. Silva, P. Andrade, I. Silva, E. Sisinni, and P. Ferrari, "An evolving tinyml compression algorithm for iot environments based on data eccentricity," *Sensors*, vol. 21, no. 12, pp. 1–25, Jun. 2021, doi: 10.3390/s21124153.
- [15] N. A. Khairi and A. Bahari Jambek, "Study on data compression algorithm and its implementation in portable electronic device for Internet of Things applications," *EPJ Web of Conferences*, vol. 162, p. 01073, Nov. 2017, doi: 10.1051/epjconf/201716201073.
- [16] G. Giorgi, "Lightweight Lossless Compression for SNS -Dimensional Data in Multi-Sensor Systems," in *IEEE Sensors Journal*, vol. 19, no. 19, pp. 8895–8903, Oct. 2019, doi: 10.1109/JSEN.2019.2922666.
- [17] K. L. Ketshabetswe, A. M. Zungeru, B. Mtengi, C. K. Lebekwe, and S. R. S. Prabakaran, "Data Compression Algorithms for Wireless Sensor Networks: A Review and Comparison," in *IEEE Access*, vol. 9, pp. 136872–136891, 2021, doi: 10.1109/ACCESS.2021.3116311.
- [18] A. Gopinath and M. Ravisankar, "Comparison of Lossless Data Compression Techniques," *2020 International Conference on Inventive Computation Technologies (ICICT)*, 2020, pp. 628–633, doi: 10.1109/ICICT48043.2020.9112516.
- [19] A. Chatterjee, R. J. Shah, and K. S. Hasan, "Efficient Data Compression for IoT Devices using Huffman Coding Based Techniques," *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 5137–5141, doi: 10.1109/BigData.2018.8622282.
- [20] S. K. Idrees and A. K. Idrees, "New fog computing enabled lossless EEG data compression scheme in IoT networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, pp. 3257–3270, Mar. 2021, doi: 10.1007/s12652-021-03161-5.
- [21] A. K. M. Al-Qurabat, Z. A. Mohammed, and Z. J. Hussein, "Data Traffic Management Based on Compression and MDL Techniques for Smart Agriculture in IoT," vol. 120, no. 3, pp. 2227–2258, May 2021, doi: 10.1007/s11277-021-08563-4.
- [22] S. A. Abdulzahra, A. K. M. Al-Qurabat, and A. K. Idrees, "Compression-based data reduction technique for IoT sensor networks," *Baghdad Science Journal*, vol. 18, no. 1, pp. 184–198, 2021, doi: 10.21123/bsj.2021.18.1.0184.
- [23] C. Passos, C. Pedrosa, A. Batista, M. Nogueira, and A. Santos, "GROWN: Local Data Compression in Real-Time To Support Energy Efficiency in WBAN," *2020 IEEE Latin-American Conference on Communications (LATINCOM)*, 2020, pp. 1–6, doi: 10.1109/LATINCOM50620.2020.9282319.
- [24] K. Hossain and S. Roy, "A Data Compression and Storage Optimization Framework for IoT Sensor Data in Cloud Storage," *2018 21st International Conference of Computer and Information Technology (ICCIT)*, 2018, pp. 1–6, doi: 10.1109/ICCITECHN.2018.8631929.
- [25] G. Shrividhiya, K. S. Srujana, S. N. Kashyap, and C. Gururaj, "Robust Data Compression Algorithm utilizing LZW Framework based on Huffman Technique," *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 2021, pp. 234–237, doi: 10.1109/ESCI50559.2021.9396785.

BIOGRAPHIES OF AUTHORS

Ahmed Najah Kadhim    obtained the BS degree in Information Networks, College of Information Technology, University of Babylon, Iraq in 2012-2013. He is currently pursuing the M.E. degree in the Information Networks Department, College of Information Technology, Babylon University, Iraq. Research interests include the internet of things, wireless network security and data compression. He can be contacted at email: ahmedn.net.msc@student.uobabylon.edu.iq.



Mehdi Ebady Manaa    is currently an assistant professor in the Department of Information Network, College of Information Technology, University of Babylon. He received his bachelor's degree from the University of Babylon, college of science in 1999-2000. His master of science from University Utara Malaysia (UUM), Malaysia in 2012. He received his Ph.D in Computer Science and in the field Network Security and Data Mining using Cloud Computing from the University of Babylon, College of Information Technology in 2016. He is currently focusing on the detection of the attacks. The main interesting fields are data mining techniques (clustering and classification), communication software, network security, cloud computing, internet of things, and unstructured data. He can be contacted at email: it.mehdi.ebady@itnet.uobabylon.edu.iq.