Spring 2021

Data and resource management in wireless networks via data compression, GPS-free dissemination, and learning

Xiaofei Cao

Department: Computer Science

DATA AND RESOURCE MANAGEMENT IN WIRELESS NETWORKS VIA DATA COMPRESSION, GPS-FREE DISSEMINATION, AND LEARNING

by

XIAOFEI CAO

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

2021

Approved by

Sanjay Madria, Advisor
Nan Cen
Sajal Das
Tony Luo
Maciej J. Zawodniok

PUBLICATION DISSERTATION OPTION

This dissertation consists of the following seven articles, formatted in the style used by Missouri University of Science and Technology:

Paper I: Pages 31-56 have been published in 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS).

Paper II: Pages 57-113 have been published in the journal Distributed and Parallel Databases 2019.

Paper III: Pages 114-148 have been published in 2019 IEEE 18th International Symposium on Network Computing and Applications (NCA).

Paper IV: Pages 149-205 have been published in journal IEEE Transactions on Sustainable Computing 2020.

Paper V: Pages 206-239 have been submitted to journal Peer-to-Peer Networking and Applications 2020.

Paper VI: Pages 240-245 have been published in 2017 14th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON).

Paper VII: Pages 246-255 have been published in 2019 38th IEEE Symposium on Reliable Distributed Systems (SRDS).

ABSTRACT

This research proposes several innovative approaches to collect data efficiently from large scale WSNs. First, a Z-compression algorithm has been proposed which exploits the temporal locality of the multi-dimensional sensing data and adapts the Z-order encoding algorithm to map multi-dimensional data to a one-dimensional data stream. The extended version of Z-compression adapts itself to working in low power WSNs running under low power listening (LPL) mode, and comprehensively analyzes its performance compressing both real-world and synthetic datasets. Second, it proposed an efficient geospatial based data collection scheme for IoTs that reduces redundant rebroadcast of up to 95% by only collecting the data of interest. As most of the low-cost wireless sensors won't be equipped with a GPS module, the virtual coordinates are used to estimate the locations. The proposed work utilizes the anchor-based virtual coordinate system and DV-Hop (Distance vector of hops to anchors) to estimate the relative location of nodes to anchors. Also, it uses circle and hyperbola constraints to encode the position of interest (POI) and any user-defined trajectory into a data request message which allows only the sensors in the POI and routing trajectory to collect and route. It also provides location anonymity by avoiding using and transmitting GPS location information. This has been extended also for heterogeneous WSNs and refined the encoding algorithm by replacing the circle constraints with the ellipse constraints. Last, it proposes a framework that predicts the trajectory of the moving object using a Sequence-to-Sequence learning (Seq2Seq) model and only wakes-up the sensors that fall within the predicted trajectory of the moving object with a specially designed control packet. It reduces the computation time of encoding geospatial trajectory by more than 90% and preserves the location anonymity for the local edge servers.

ACKNOWLEDGMENTS

Throughout the writing of this dissertation I have received a great deal of support and assistance.

I would first like to thank my supervisor, Dr. Sanjay Madria, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would like to thank my parents for their wise counsel and sympathetic ear. You are always there for me. I would like to thank for my family for their strong support.

Finally, I could not have completed this dissertation without the support of my friend, Azharul Islam, who provided algorithm discussions as well as happy distractions to rest my mind outside of my research.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

Figure Page

SECTION

PAPER I

PAPER II

PAPER IV

PAPER V

LIST OF TABLES

SECTION

1. INTRODUCTION

The Internet of Things (IoT) facilitates fast access, process, and utilization of the big data created by the 'things' surrounding many applications such as environmental data monitoring, disaster management, or battlefield monitoring. As IoT is developed for a plethora of emerging applications in a wide range of disciplines, there are roughly 9.5 billion connected IoT devices reported at the end of 2019. It is also expected that the total number of connected IoT devices will reach 28 billion by 2025. Wireless sensor networks (WSNs) as an important sensing organ of the IoT family, contribute most of the increasing population due to its economic efficiency. Lots of WSNs applications gathering researchers' attention. For example, there are near real-time sensor cloud applications [1] to perform multi-modal sensing tasks. Some military applications like tracking hostile objects or monitoring intruders use multiple sensing units to provide precise location and speed by applying multi-sensor data fusion. The unmanned vehicles [2] need GPS and accelerometer to locate themselves, and to track distance and height of objects using the camera, laser range meter, or radar. By fusing these multi-modal sensor data, they can also predict the moving trajectory of the nearby objects. Similarly, the environmental monitoring applications [3] [4] [5] need temperature, wind direction, humidity, CO2 levels, etc. Many of these multi-modal sensor applications require data integrity (lossless data) as well as high-streaming rate, and therefore, cannot tolerate high latency due to limited bandwidth of IoT sensing networks. The WSNs can also responsible for content distribution, resources management, or reflecting on physical world decisions made at software level. The following subsection will introduce some applications using WSNs in detail and list challenges these applications face. Thereafter, we introduce our proposed data and resource management approaches,

which include efficient data collection framework, data compression algorithm, secure object tracking framework, to counter the challenges those real-world WSNs applications will encounter. At last, we briefly describe the performance metrics we used to evaluate the effectiveness of the proposed data and resource management approaches.

1.1. APPLICATIONS AND CHALLENGES FOR WSNS

There are many WSNs applications. We classify them into the following three categories: Environmental data monitoring applications, mobile sensor applications, and object tracking applications. They have different features and challenges which are elaborated in detail below.

1.1.1. Environmental Data Monitoring Applications with Static Nodes. Environmental data monitoring is the most common application where WSNs are used for both indoor and outdoor environmental data monitoring using static sensors and static sinks. For example in [3], the researchers deployed five eZ430 wireless sensors sensing temperature and humidity in a greenhouse. The authors found after 5 days, the battery capacity of the sensors is lower than 80% which indicates after 30 days the sensors will run out of battery. The experiments show that by reducing the number of communication between sensors could improve the lifetime of the sensors. However, the trade-off is that the sensing frequency needs to be lowered which may be affect accuracy of the data reported.

In another work [6], a real-time environmental monitoring cyber-infrastructure with WSNs was proposed. It monitors the outdoor soil moisture with flexible spatial coverage and resolution using wired and wireless sensors. It provides remote, near-real-time monitoring, long-term, and autonomous publicly available web services for sensor data visualization and dissemination. Besides, it achieves remote system monitoring and maintenance for system development, debugging, and maintenance purposes. To save energy and prolong the lifetime of the WSNs, a data aggregation method and a duty cycle approach are used to reduce the communication load and redundant overhearing.

Another research in [7] proposed an indoor temperature monitoring and regulation application. They use wireless sensors to monitor and control heating bodies, such as central heating radiators, electric radiators, or fans, and air conditioners, which can be based on a fan. The regulator can increase/decrease the cooling volume to reach a certain temperature. Third, airflow, such as central airflow ventilation which can regulate the degree of air circulation. Next, they also control window shutters, such as outside curtains. By raising/lowering the curtains the influence of solar energy can be regulated. The wireless sensors show their flexibility and easy deployment advantage in these applications.

These tiny low-cost wireless sensors suffer a lot of constraints including low computational power, less memory and storage space, and less energy capacity. Since batteries are the typical power source for wireless sensors, the limited energy budget is another primary constraint in the design of multi-modal WSNs. To address these problems and make WSNs more energy-efficient and bandwidth-efficient as well as to meet the QoS (quality of service) requirements, researchers studied the sensor's power model and proposed algorithms to optimize the WSNs from low-level (MAC) multimedia access control protocol to high-level data collection schemes. Many research efforts, like [8] and [9], have shown that radio communication, including data transmission and channel listening, is the predominant factor among all the energy consumption metrics of the WSNs. In a contentious WSN, the network congestion caused by overwhelmed channel load is another challenge that risks the QoS including throughput, delay, and data integrity [10]. Some carrier sense multimedia access (CSMA) MAC protocols have poor performance under this condition and their back-off mechanism will deplete the sensors' battery faster than the time divisor multimedia access (TDMA) MAC approaches do. Privacy is another challenge in WSNs. Sensor nodes with computational power are vulnerable to eavesdropping, hijacking, and can be easily compromised. The adversary could easily eavesdrop the communication of the WSNs and get the privacy information like the location of the users.

1.1.2. Applications with Mobile Nodes. In many WSN applications, like wild animal protection, mobile crowd-sourcing, and vehicluar network, the sensor nodes or sinks may have mobility. For example, in the ZebraNet project [11], the sensors are collared on zebras' neck. It tracks the moving trajectory of a zebra herd using GPS sensors. The sink is also mobile as any unguarded base-station or cellular station will attract the attention of wild animals. As the project was for one year, energy was a critical challenge. The bandwidth was also a challenge when the herd is far from the mobile sinks. However, the latency was not a problem though the time-stamp was still needed to be attached to the data packet.

In recent crowdsourcing applications [12] [13] [14], by utilizing the mobile devices' GPS data, air pollution data can be gathered efficiently. Another interesting application in Japan [15] uses garbage trucks to collect pollution data thus get a more accurate pollution map. The researchers also Innovatively use the vibration of the garbage truck to estimate the garbage volume of an area. Also, with the emerging of 5G techniques, researchers are discovering possible applications in Vehicular ad-hoc networks (VANETs) [16] [17], and [18].

An efficient message dissemination framework is the key factor in mobile WSNs as it involves real-time identification of a secure routing path based on network topology, the application's requirements, and a user's privacy preferences. However, three notable challenges are unaddressed in this regard. First, the location privacy can be exposed when messages are disseminated in an unsafe IoT network such as network containing third party devices, eavesdroppers, and malicious adversaries. Second, to be generic and scalable, the message dissemination framework must be able to virtualize and abstract spatial-temporal message dissemination services, such as location encoding, router selection, and duty-cycle management, from the physical routing infrastructures like cluster, grid, and table-based routers. Third is the trade-off between Quality of Service (QoS) and energy efficiency, which are two conflicting requirements, difficult to balance for applications that have dynamic

QoS requirements. For example, battlefield WSNs pose to conceal their radio footprint while detecting enemy targets to avoid being detected by the adversary. Conversely, a soldier requesting crucial data from a battlefield will give a much higher preference to the confidentiality and less to latency at the cost of intense radio communication even though it increases energy consumption, bandwidth usage, and risk of being detected. As such, hard-coded routing rules can't fit the diverse needs of real-world network security and QoS features. Further, provisioning spatial-temporal message dissemination without GPS is necessary but challenging for particular applications such as an underwater sonar network that can't access GPS signal and a battlefield WSN where GPS signal can be spoofed.

1.1.3. Object Tracking in IoT Environment. Moving object tracking is one of the important applications of wireless sensor networks and is widely used in both civil, research, agriculture, and military applications. For example, the military can use wireless sensor networks to track military vehicles [19]. The smart city uses WSNs to fetch the trajectory of every vehicle and use the information to guide other commuters or detect abnormal driving behavior [20]. Underwater robot localization and tracking [21] is another important application that could requires cooperation of multiple type of sensors.

In these target tracking applications, different sensors monitoring the targets continuously during their mobility and thus faces several challenges. First, not all sensors contribute equally to target tracking. Unnecessary sensing by the sensors which are off the targets could cause excessive energy consumption. Second, detecting targets with long sensing period in a WSN could be a challenge due to limited battery power. Given low power listening (LPL)[22] and other energy-efficient MAC protocols, most of the real-world WSN applications can put the sensors into the wake/sleep cycle, and only wake up some of them for a small period for sensing and communication. However, in LPL mode, sensors may face a high risk of losing the target because of the low sensing and communication frequency. Third, for some military applications, sensors also need to prevent being detected by other enemy targets. For example, once the enemy targets detect the presence of nearby sensors

through the radio signals, they may perform some adverse actions which may increase the detection difficulty. The enemy targets may also start jamming the radio transmissions in the area [23]. Fourth, obtaining the enemy targets' location is also a challenge. The sensors detecting the target need to estimate the target's location. However, due to the low cost of the WSNs, most of the sensors have no GPS modules. Though virtual coordinates [24],[25] and [26] could calculate the approximate location of sensors, fetching all the hop information of the sensors in a large scale WSN and calculating the location centrally can cause excessive energy consumption. Lastly, location anonymity is also a challenge as the sensors are vulnerable to be eavesdropped and can be compromised, thus directly transmitting the location information is not secure.

1.2. THE PROPOSED DATA AND RESOURCE MANAGEMENT APPROACHES

To address the energy and bandwidth challenge for static WSNs that affect the radio communication QoS and network lifetime, the dissertation first proposes an efficient sensor data compression algorithm [27] [28] that reduces the number of packets need to transmit between the intermediate nodes of the WSNs. Thus, energy and bandwidth are saved and the lifetime of the network is extended.

When disseminating data in mobile WSNs, reducing redundant data transmission could not only save energy but also save bandwidth thus increase the throughput and reduce delay when the channel is busy. This dissertation presents a novel data collection scheme for mobile IoT edge networks [29] [30] that reduces the redundant re-transmission and save both bandwidth and energy.

Last but not the least, to tackle the challenge of predicting and tracking object in IoT environment, we proposed a sequence to sequence (Seq2Seq) model that takes the target's previous estimated locations as input and outputs a geometric constraint that allows only the sensors within the predicted trajectory to wake up, detect and track the target and report the results to the nearest local edge server. A set of these constraints creates

a path constraint that covers all the areas of the target's previous and predicted trajectory. The proposed framework directly predicts the routing constraints that cover the target's predicted trajectory. It is much faster than predicting the sequence of future target's location first and then encoding the predicted trajectory. Compared to the cluster-based target tracking approach, the geometric constraints based routing protocol reduces the overhead of cluster generation and maintenance. The location anonymity is preserved as no GPS data are used and transmitted. The performance evaluations show the effectiveness of the proposed scheme over other competitive schemes.

1.2.1. Sensor Data Compression. To further compress the collected data, we propose a Z-order [31] encoding based data compression scheme. The Z-order encoding called Z-compression can compress multi-modal sensing data at each leaf node as well as at the intermediate nodes efficiently in near real-time. The Z-compression algorithm can encode multi-modal sensor data like precipitation, water level, and wind speed (needed to detect a flood risk in a region) into a binary stream. It is a lossless compression algorithm, i.e., data can be decompressed at the sink without any loss of accuracy. Using our Z-compression algorithm in a WSN with a hierarchical topology [32], the nodes with limited bandwidth can tolerate higher-stream data rates coming from upstream nodes by concatenating compressed sensor data into the reduced number of packets which may be as large as permissible by the network protocol. The proposed Z-compression algorithm also uses temporal and spatial data locality and delta encoding for better performance. Instead of using Huffman style coding which requires extra bits for each delta values, we use Z-order encoding to compress the delta values of all attributes of the input data into a binary stream. When decoding we use the predefined decoding rules to decode the Z-values and extract all the values of attributes. We also proposed an Optimized Z-compression algorithm which further increases the performance when compressing multi-modal sensor datasets. In a tracking application, longitude and latitude values of a vehicle equipped with GPS, Z-compression first transfer

data into delta values [33]; change between the current and previous sensed values. Then, the delta values of longitude and latitude are further compressed using Z-order encoding for better compression.

1.2.2. Data Request and Data Routing in WSNs. We propose a spatial data collection scheme that has both low latency and less overhead of redundant broadcasting. Instead of using the exact nodes' location information from GPS as in [34][35][36], we use a vector of the minimal distance of hops (DV-Hops) to all the anchor nodes selected by the secure fog server as a dictionary or virtual coordinate. The area of the position of interest (POI) can be represented as a list of hop constraints to the anchor nodes. Our routing message only contains two basic geometric shapes: hyperbola and ellipse segments. Each shape is encoded with simple hop constraints (e.g., size of the ellipse and the hyperbola and the start and end of the segment). The sensor nodes could avoid complex geometric computing, which makes it suitable for a WSN that have low-power and low-computing resources. Besides, the proposed scheme provides location anonymity by avoiding using and transmission of the GPS location information. To decode the POI of the client, the adversary has to have the encoded message as well as the location of the anchor nodes, which are stored in the secure fog server. To address the mobile nodes issues in heterogeneous networks, we proposed a DV-Hop updating algorithm that updates the DV-Hop of the moving nodes. We also address the broadcast storm issue by integrating the counter-based broadcasting mechanism.

1.2.3. Object Prediction and Tracking in IoT Environment. Lastly, our study will therefore focus on solving the following challenges. First, to locate and track the moving target without using any GPS-based sensors. Second, to deliver the wake-up and reset message to the area around the target's path quickly and energy efficiently. Last but not the least, to preserve the location anonymity of the sensors tracking objects.

In this thesis, a Seq2Seq learning model based trajectory prediction framework is proposed. We offload the trajectory encoding task to the local edge server and add a trajectory prediction functionality that predicts the moving event's future trajectory based on its previous locations. Instead of querying the remote cloud to predict the target's trajectory and to generate the data collection messages that cover the target's trajectory, the framework will choose the nearest local edge server to perform the trajectory prediction. Thus, the multi-hop radio communication delay is decreased. Also, the Seq2Seq learning-based encoding model accelerates the trajectory encoding algorithm by more than 100 times. This model will learn from the previous event moving patterns and directly predict the shape constraints of the future trajectory from the location points of the previous trajectory. For a large-scale IoT network, we divide it into small overlapped grid cells. We train the trajectory prediction model for each grid cell based on their network topology and assign these models to the local-edge-servers of each cell. The model-based trajectory prediction framework allows the third party local-edge-servers to predict the target's future trajectory without leaking the location information of the anchors, the event's trajectory, and the user's location.

1.3. PERFORMANCE METRICS

To evaluate the effectiveness of a compression algorithm and a trajectory-based routing algorithm, we make comparisons between other schemes to understand certain performance metrics. In the survey on data compression in WSNs [37] and survey of counter-based broadcast protocol [38], the following metrics are commonly used and many past proposals have used them to compare their contributions against other state-of-the-art approaches. To evaluate the efficiency of the data dissemination in the WSNs, the average delay and the redundant rebroadcast ratio are used. Lastly, to evaluate the effectiveness of the proposed seq2seq model for trajectory prediction and encoding, two types of accuracy metrics are defined which are discussed below.

1.3.1. Compression Ratio. The compression ratio CR in the performance metrics is defined as the compressed data length over the size of uncompressed data as shown in Equation 1.1. A base station is needed to collect all n compressed packets. $\sum_1^n L_{compressed}$ is the compressed data size and $\sum_1^n L_{original}$ is the original data size. For a compression algorithm or a trajectory encoding algorithm, the higher the compression ratio the better performance is.

$$CR = \frac{\sum_1^n Length_{original}}{\sum_1^n Length_{compressed}} \tag{1.1}$$

1.3.2. Normalized Compression Overhead. When comparing prefix encoding based compression algorithms, the normalized overhead $Overhead_{norm}$ is the ratio of the prefix bits required to code the data to the actual data bits (binary format of the data value). It can be represented as in Equation 1.2. For a prefix encoding based compression algorithm, the normalized compression overhead changes when the length of the input data changes. By analyzing the normalized compression overhead of a compression algorithm we can estimate the overall compression performance with the input data distribution.

$$Overhead_{norm} = \frac{len(Prefix)}{len(DataValue)} \tag{1.2}$$

1.3.3. Average Delay. Routing/broadcasting delay is an important metric measuring the QoS of the WSNs. The average delay is the time when the sink receives the data minus the time when the source reports the data. Routing hops, duty cycle, and waiting time due to data aggregation or concatenation are three key factors that determine the average delay.

1.3.4. Redundant Rebroadcast Ratio. In energy-efficient broadcast and routing protocols, redundant rebroadcast is an important metric that determines their energy efficiency. As shown in Equation 1.3, $Redundant_{rebroadcast}$ is the ratio of the number of

rebroadcasting that happens outside of the position of interest (POI) over the total number of the nodes in the POI.

$$Redundant_{rebroadcast} = \frac{NumBroadcast_{outside}}{Nodes_{inside}} \tag{1.3}$$

1.3.5. Accuracy Metric for Trajectory Prediction Models. Two types of accuracy metrics are defined. One is called the valid coverage score (ACS) which is defined in Equation 1.4. The other is called the false activation rate (FAR) which is defined in Equation 1.5. As shown in Figure 1.1, $Area_{Trajectory}$ is the area of the desired target trajectory. $Area_{covered\_by\_constraints}$ is the area calculated by testing every pixel of the working area with the DV-Hop constraints. All the pixels valid for the hyperbola constraints are counted. Therefore, the high ACS score means more areas of the target trajectory are predicted and covered. The higher FAR rate means more false areas are predicted which will incur redundant rebroadcast and waste sensors' energy.

$$ACS = \frac{Area_{covered\_by\_constraints} \cap Area_{Trajectory}}{Area_{Trajectory}} \tag{1.4}$$

$$FAR = \frac{Area_{covered\_by\_constraints} - Area_{Trajectory}}{Area_{covered\_by\_constraints}} \tag{1.5}$$

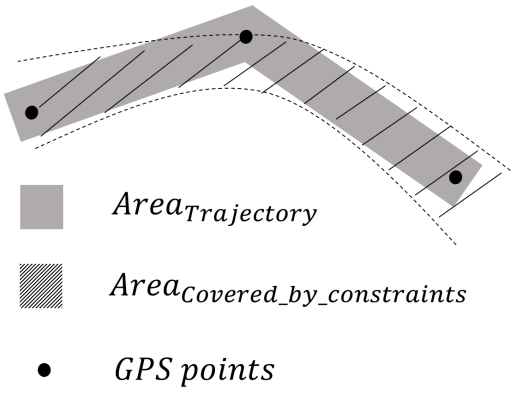


Figure 1.1. An example of the definition of $Area_{covered\_by\_constraints}$ and $Area_{Trajectory}$

2. LITERATURE REVIEW

As we have discussed above that radio communication, including the data transmission and channel listening, is the predominant factor among all the energy consumption metrics of the WSNs. The best strategy to save energy and bandwidth is to reduce the redundant radio transmission by only collecting and transmitting compressed data in the location of interest (trajectory). In this section, we will review some of the proposed works related to sensor data compression, trajectory/geometric based routing, duty cycle protocol, data aggregation, and simulation and data processing techniques used for this research.

2.1. SENSOR DATA COMPRESSION ALGORITHMS

In this section, we will discuss several popular sensor data compression algorithms including LEC, TinyPack, Huffman-coding, FELACS, S-LZW, and many other model-based compression algorithms.

2.1.1. Prefix Coding Based Lossless Compression Algorithm. Huffman coding [39] is a classic prefix coding algorithm which is commonly used for lossless data compression. It assigns shorter prefix code for higher frequency input data while assigns longer prefix code to lower frequency input data. For example, suppose we have an input string "Huffman coding is awesome" which has 15 distinct symbols. By counting each symbol's appearance frequency, we can create the weight table for these symbols as shown in the Table 2.1.

Then we can build the Huffman coding tree from the input string as shown in the Figure 2.1. where each edge represents a binary bit, each leaf node represents the distinct symbol, and each internal node represents the sum of its children's weight. Then, all the distinct symbols can be represented with the bit code from root nodes to the leaf nodes. The length of the encoded string is 97 bits.

Table 2.1. Weight table of input string "Huffman coding is awesome"

symbol	frequency	weight
space	3	0.12
a	2	0.08
c	1	0.04
d	1	0.04
e	2	0.08
f	2	0.08
g	1	0.04
H	1	0.04
i	2	0.08
m	2	0.08
n	2	0.08
o	2	0.08
s	2	0.08
u	1	0.04
w	1	0.04

A good Huffman coding algorithm should minimize the total number of bits of the encoded data in less computational expenses. The above example using a priority queue(heap) to greedily group pair of least weight nodes into an internal node, which weights the sum of the two nodes, and pushed the internal node back to the priority queue. Repeating the previous step until only one internal node left which if the root of the Huffman tree. The time complexity of this Huffman tree building algorithm is $O(n \log(n))$ and the memory complexity is $O(n)$, where n is the number of distinct symbols. Based on Shannon's entropy Equation $Entropy\ H(X) = -\sum p(X) \log p(X)$ [40], for the above example, the theoretic average bits per symbol of the import string is 3.8137 bits. The average number of bits for the encoded string is 3.88 bits.

The decompression procedure requires the decoder to have the Huffman tree/code dictionary of the symbols. By mapping the key of the code dictionary, the decode speed can be as fast as O(n) where is the number of bits in the encoded string. Figure 2.2 shows an example of decoding the encoded string with a given Huffman tree.

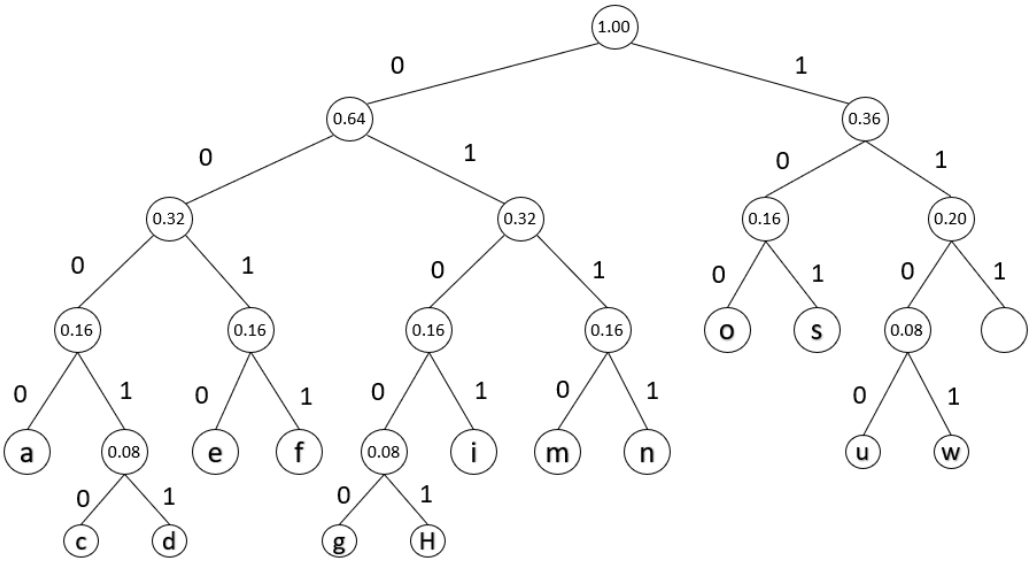


Figure 2.1. Huffman tree of input string "Huffman coding is awesome"

The drawback of directly using Huffman coding in a wireless sensor network is the overhead of creating and transmitting the Huffman coding tree. For WSNs, the distinct symbols are the value of the sensing data which can vary from 0 to 2^b where b is the length of bits of the sensing attributes. Building and maintain the Huffman tree with such big symbol pools is not practical in WSNs as sensors usually lack computation and memory resources. Moreover, as the Huffman coding tree is required for decoding, it has to be transmitted along with the encoded message. However, considering the sensing data is small, the overhead

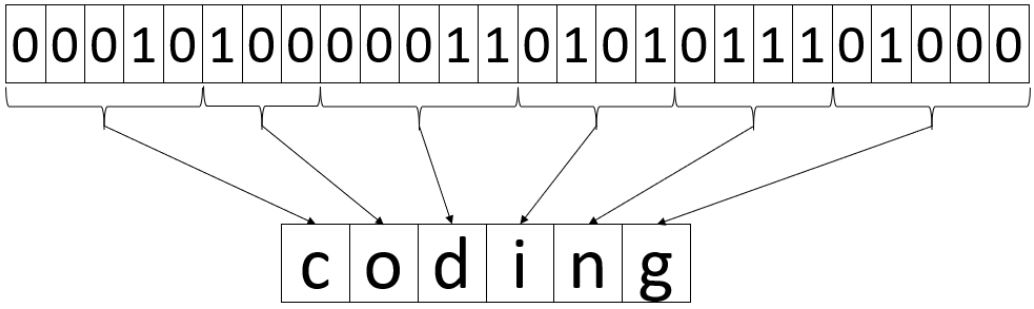
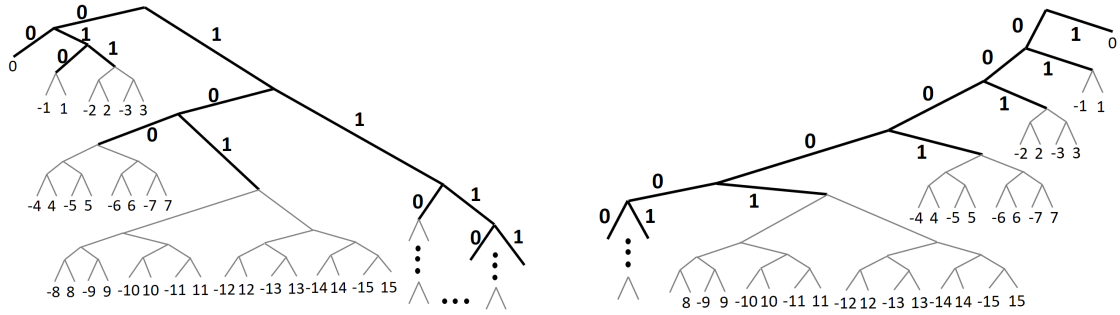


Figure 2.2. Decoding example of Huffman coding algorithm"

of storing and transmitting the Huffman tree is bigger than the redundant bits the encoding algorithm saves. To address these problems, many adapted Huffman coding algorithms for WSNs are proposed like LEC [41], TinyPack [33].

LEC [41] and TinyPack [33] are both Huffman coding based compression algorithm. First step in their adaption is to use delta value, which is the difference of two consequent sensing value, as the input symbol rather than using the actually sensing values. In their improved approaches to save memory and to make them computationally efficient, instead of adapting Huffman coding to each symbol, LEC and TinyPack create the Huffman coding for the length of the delta value in a predefined pattern that matches the frequency of the data length group which they assume is decreasing when the length of delta value increases. Thus, in both LEC and TinyPack, the length of Huffman coded prefix increases with the increasing length of the delta values.



(a) Huffman tree of LEC code (b) Huffman tree of TinyPack's code

Figure 2.3. Huffman tree of the compression code of LEC and TinyPack

The Huffman tree of LEC is shown in Figure 2.3-a and the TinyPack code is shown in Figure 2.3-b. These two coding focused on the different distribution of the delta values of the dataset. The TinyPack has better performance when majority of the input data are zeros while LEC optimized the performance when most of the data are less than five bits long. For example, if a dataset contains many data of only one-bit length, according to Figure 2.4,

TinyPack will have better compression ratio than LEC in terms of normalized compression overhead (ratio of the extra bits required to code the delta value to the data bits) whereas, for other delta values larger than 3, LEC will have better performance than TinyPack.

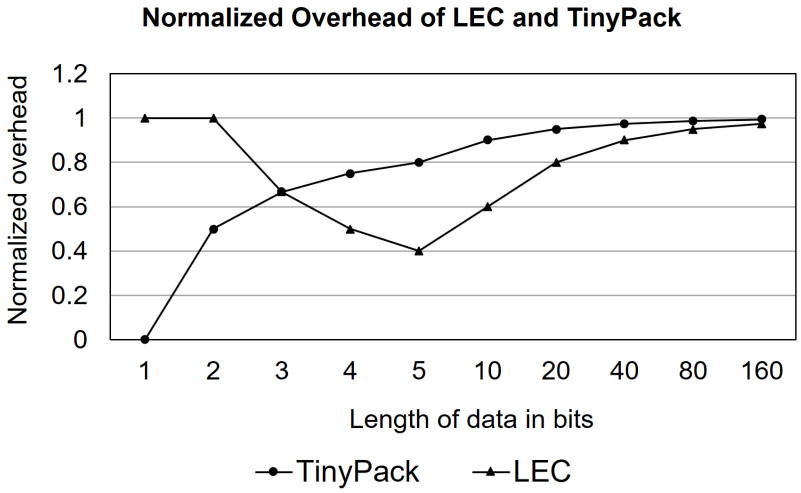
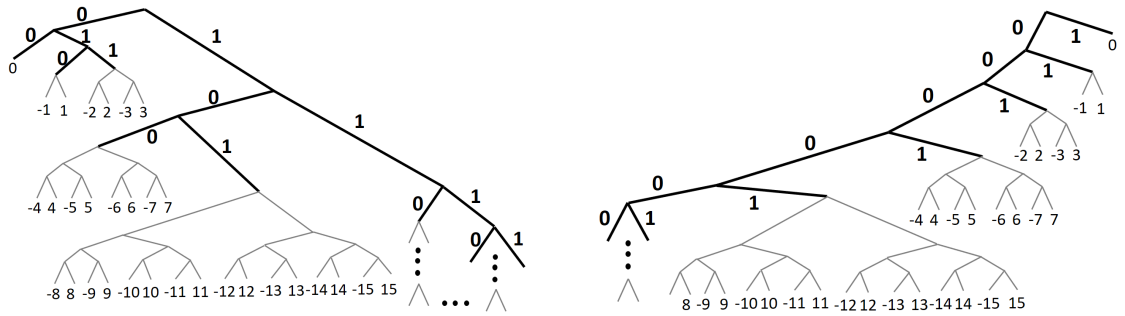


Figure 2.4. Normalized compression overhead of LEC and TinyPack

However, as we have discussed before, the distribution of the delta values can be varied in different applications. Even in the same application at different time periods or at different sensor nodes, the distribution may vary. Thus, in these situations, LEC and TinyPack do not show good performances as they have fixed code. To solve this problem, Adaptive-LEC [42] and TinyPack-DP(TP-DP)[33] can adapt the prefix code when the length of the distribution of the frequency of the delta value changes. Figure 2.5 shows the predefined rotating dictionary.

Adaptive-LEC will adapt its prefix code for every new data while the TP-DP only changes its prefix code at the beginning of each new frame. They define the length of delta value with the most frequency as the frequency center of data and define the length of delta value with the minimum prefix code as the code center. When the current frequency center of data is drifted from the previous code center, the prefix code will shift to meet



(a) Single rotate dictionary of the adaptive LEC (b) Double rotate dictionary of adaptive LEC

Figure 2.5. Single rotate and double rotate dictionary of adaptive LEC

the current frequency center of the data. For some datasets with the two frequency centers, they proposed the double rotate initial code with the two code centers and two adaptive segments where the prefix codes will be adapted independently based on the segments.

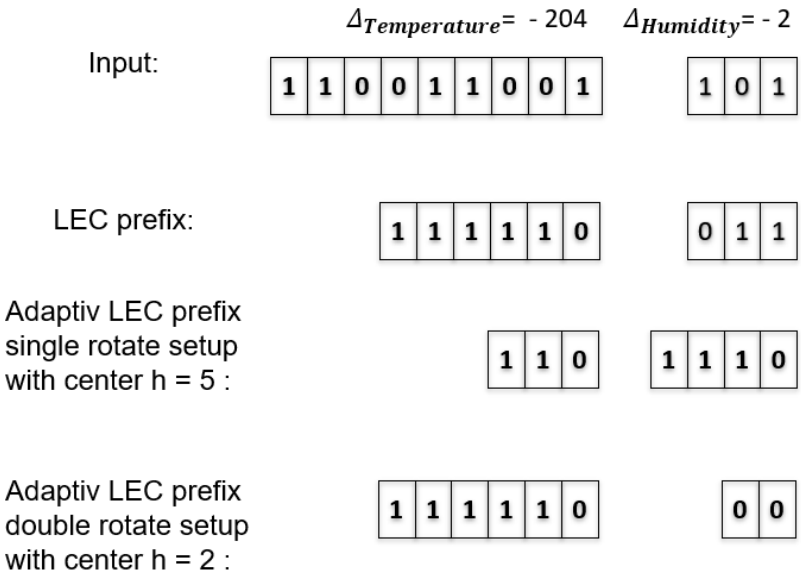


Figure 2.6. An encoding example of LEC and adaptive LEC

Figure 2.6 shows the procedure of encoding input data with two sensing attributes' delta value using LEC and adaptive LEC. It shows when the center is optimized, the adaptive LEC could provide a better compression ratio than LEC. However, the drift correction (set

code center to be optimized), always behind the actual center drift, happens with a very high delay. Thus, it reduces the performance of the compression algorithm. Also, different sensor nodes may have different code centers in WSNs.

The adaptive TinyPack [43] algorithm takes advantage of spatial locality between the two or more nodes and performs collaborative data compression. However, in WSNs, sensors may be sensing multidimensional data independently, and waiting for other sensors to dispatch their different attribute values will increase latency in processing and delays in transmission in an intermittent environment. Thus, this approach trades the response time for a better compression ratio.

2.1.2. Fast Efficiency Lossless Adaptive Compression Schema (FELACS). FELACS [44] gives every compressing data a fixed b bits. Every value smaller than 2^b will be filled with '0's at the front to reach b bits, added '1' bit at the front then directly appended to the output stream. Every value larger than or equal to 2^b will be cut into two sections. The higher bits section is encoded using unary coding and is appended to the output stream. The lower bits section has the length b directly appended to the output stream. The fixed bits b is generated by calculating the average number of the input data.

FELACS adapts data packets by packets rather than sample by sample. It works for a single sensing attribute only. Also, it needs to wait for more data to achieve better compression performance as they have fixed indicator bits.

2.1.3. S-LZW Algorithm. S-LZW [45] an extension of LZW [46] that compresses data by encoding and representing a common sub-string with fewer bits used for the encoded value. The encoded common sub-strings are stored in the dictionary and represented as the dictionary index. The next subsequence of the encoding data are represented with the index of the longest matching subsequence in the dictionary. However, the dictionary of LZW is too large to store at sensor motes. Therefore, S-LZW sets the dictionary size to be 512 bytes, using a 32-byte mini-cache, and also, they tested the performance with a dataset of 528 bytes; the size of each block of buffered data. The authors believed that before the

data can be compressed, the entry in the dictionary is a good representation of the data being compressed. LZW and S-LZW usually need more data to create new words and update their library before compressing, which causes long delays. The average waiting time of S-LZW compression algorithm on compressing eight pages each with 256 bytes of Intel Berkeley Lab sensor data is about 2.8 times the time interval between two consecutive packets. Additionally, the average waiting time increases so LZW and S-LZW cannot be used for real-time wireless sensor applications as shown in our earlier work also [33].

2.1.4. Other Compression Algorithms. The model-based compression algorithms [47] such as APCA[48], PWLH[49], and SF[50] also have good compression ratios. They use the mathematic models to approximate a sequence of sensing data. For example the constant model approach APCA [48] use a constant line model while PWLH [49] uses a linear line model as shown in Figure 2.7. They can both present multiple data points with few model parameters thus achieve a good compression ratio. However, they can not work with applications requiring lossless data. The compressive sensing approaches like [51] have the drawback that the data may lose integrity thus, are not suitable for multi-modal lossless data compression.

2.2. ENERGY EFFICIENT DATA BROADCAST AND ROUTING ALGORITHMS

Efficient data dissemination protocol could not only reduce the delay but also save energy. In this section, we will discuss some classic data dissemination protocols including Counter-based broadcast, energy efficiency routing schemes, and trajectory-based routing and virtual coordinates.

2.2.1. Counter-based Broadcasting Schemes. Broadcasting is the fastest way to flood a message into the whole WSAN. However, limited bandwidth causes a delay in broadcasting a sequence of messages into the network. After a node receives a given packet, the counter-based broadcasting schemes [52][53][54][55] require a node to wait for a short period to listen (random access delay) to its neighbors and count how many times

the given packet has been rebroadcast. If the broadcast count of the given packet reaches the predefined threshold, it will drop the packet. Thus, only a few of the nodes in the network will rebroadcast the given packet which saves bandwidth and thus, alleviates the congestion. The predefined counter threshold (CTS) and the random access delay (RAD) are two hyperparameters that determine the performance of the broadcast. Based on the strategy of configuring these hyperparameters, we classify the counter-based broadcasting approaches into the three categories as follows: (1) Neighborhood-aware counter-based broadcast. (2) Topology-aware counter-based broadcast. (3) Energy residual aware counter-based broadcast.

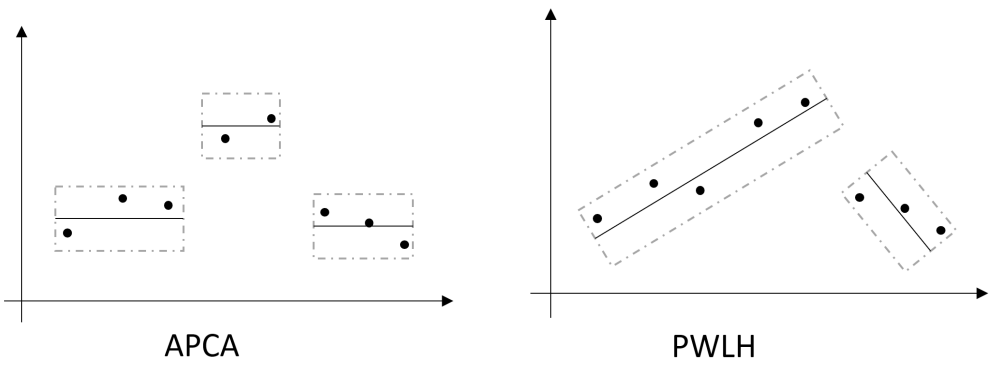


Figure 2.7. An example of APCA and PWLH constant model

Neighborhood-aware counter-based broadcast approaches, like [56], tweak the random access delay (RAD) based on the number of neighbors of the receiver. The nodes with less neighbors will has longer RAD that reduce the probability of been chosen to be the rebroadcast nodes. In contrary, nodes with more neighbors is more likely to rebroadcast the packets that could increase the overall coverage ratio of the broadcast.

$$RAD = rand[0, 1] \times T_{max} \times \frac{(R^2 - D^2)}{R^2} \tag{2.1}$$

Topology-aware counter-based broadcast approaches, like [53] and [54], set the RAD based on the receiver's location, network density, overlapping area. The approach in [53] use the distance between the broadcaster and receiver as the control parameter as shown in Equation 2.1 where R is the radio range, D is the distance between the broadcaster and the receiver, and T_{max} is the maximum listening time for a sensor node. The idea behind the Equation 2.1 is the father nodes will have high possibility to rebroadcast thus reduce the total number of hops to broadcast a message to the whole network.

The Level-based approach [57] exploit the topology of the network and select minimum nodes that cover the most of the network.

Energy residual aware counter-based broadcast considering the remaining battery of the receiver as a factor of choosing the RAD values. For example in the hybrid RAD decision Equation of [54], the battery factor is as shown in Equation 2.2 where $E_{residual}$ is the energy residual of the receiver and E_{max} is the maximum energy of new battery. The hot spot problem that some nodes deplete energy faster than others could be solved by considering the energy residual of the receiver.

$$RAD_B = rand[0, 1] \times (1 - \frac{E_{residual}}{E_{max}}) \tag{2.2}$$

2.2.2. Energy Efficiency Routing Schemes. Hierarchical grid-based routing is an energy-efficient method for routing of data packets [58]. With the mobile sink and predefined virtual grid, packets could bypass the congestion area of the grid and route to the mobile sink by fetching the updated mobile sink's location from the cell-center. The grid-based routing protocol can be classified into two categories, the query-based protocol (i.e. PANEL [59], Grid-Based Coordinated Routing [60], GMCAR [61], etc.) and the event-based protocol (i.t. TTDD [62], GMR [63], EAGER [64], etc.). For the query-based routing protocol, sensor nodes only sending the sensing data on request, while event-based protocol sensors will report events based on the pre-configuration of the event definition. Comparing to the

event-based routing protocol where each sensor report sensing data periodically, the query-based routing protocol greatly reduces the overhead of unnecessary sensing and routing, which saves both energy and bandwidth. However, efficiently disseminating the data request packets in a WSN without GPS is a challenge for the query-based routing protocol. Some works like [58] switches alternately grid-head states to overcome the energy and bandwidth overhead of flooding control packets. The work [65] uses the location information of the cell-header and their neighbors to forward the query towards the target cell and flood message only in the target cell. Though these works reduce the broadcast overhead, the grid-based routing still needs the GPS information and extra energy to maintain the grid topology.

Ring routing and nested routing [66][67] are proposed to solve the problem of routing packets to a mobile sink. The idea behind ring routing and nested routing is to store the current mobile sink's location in a ring or nested ring structure. The data source needs to query the nodes in the ring/nested ring structure to fetch the updated location of the mobile sink before routing the packets. Then the data source routes the packets to the mobile sink using the updated sink location. Ring routing and nested routing achieve good delay and energy performance because searching the ring structure is easier than searching the whole network.

$$R_i = (1 - c \times \frac{d_{max} - d_{base}}{d_{max} - d_{min}})R_0 \tag{2.3}$$

Some cluster based communication protocols like [68][69][70] could saves energy by only let the cluster heads to broadcast. Thus eliminate the redundant rebroadcast as well as broadcast storm effect. There are two major challenges for cluster based communication protocols. One is the hot-spot issue which is addressed in HEED [69] protocol. It proposed an adaptive clustering algorithm that elect a sensor node with high battery residual to the cluster head thus improved the total life-time of the WSNs. Some clustering algorithms tweaked the size of the clusters and also achieve good energy saving. For example in the work [70], the authors proposed a novel clustering algorithm that group sensor nodes into

clusters with unequal size which radius is defined in Equation 2.3 where c, d_{max}, d_{min}, R_0 are predefined parameters, d_{base} is the distance from that cluster head to the base station, and R_i is the radius of the i'th cluster. By doing so, the hot spots near the base station can be off load to more clusters. Second challenge is the overhead of maintaining the cluster topology. As we have discussed above, to mitigate the hot spot issue of WSNs, cluster head need to be re-elected if the current cluster head deplete too much energy. However, re-elect a cluster head introduce redundant message exchange which contribute to both the energy overhead and bandwidth overhead. It is a challenge to find a balance point that mitigate hot spot problem but not incur too much cluster head re-election overhead. Suppose in each round the WSN would re-select it's cluster head (CH). If a round lasts too long, the CHs will soon die. On the other hand, if a round is too short, the topology of the clusters will change frequently and considerable energy shall be wasted in the setup phase instead of spent on data communication. In [71], the author proposed a fuzzy interface system which can determine the optimized round period for the current status of the WSN. Thus it achieve a good energy saving compare to HEED.

2.2.3. Trajectory-based Routing and Virtual Coordinates. Trajectory based routing [34][35] is a paradigm that only the nodes near the given routing trajectory will forward the packets. It includes trajectory generating and encoding and the routing decision rules for each sensor node. It has the following challenges: First, the trajectory encoding algorithm should able to compress the trajectory as the encoded message will be included in the routing packets. Second, each compressed message should be able to route through the trajectory to the sink reliably. Third, the overhead of routing caused by redundant rebroadcast should be minimized. However, for low cost WSNs without GPS module, the additional challenge is to route through a trajectory without using any GPS-based location information.

A virtual coordinate system is an option for IoT WSNs without GPS. It can use local connectivity information such as the number of neighbors of each node and the perimeter nodes' locations as in [72]. It can also use the anchor nodes and the vector of minimum hop distance (DV-Hop), which shown in Figure 2.8, to the anchor nodes to estimate the distance between nodes.

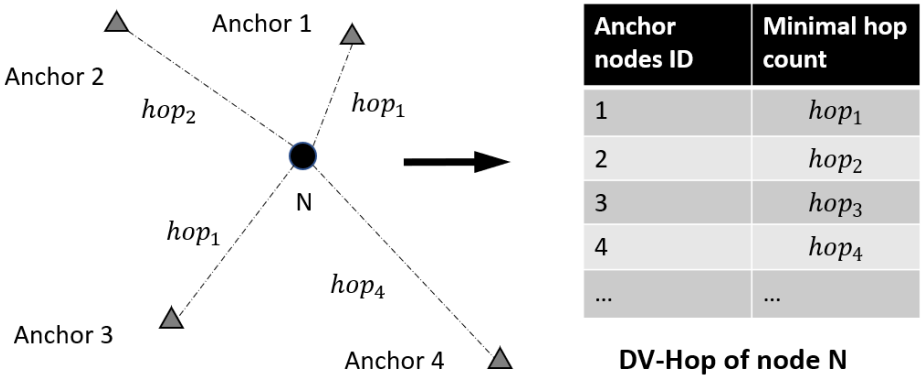


Figure 2.8. An example of DV-Hop of a sensor node N

To route through a trajectory with virtual coordinates using DV-Hop rather than GPS, the virtual coordinates should be able to reflect the sensors' real location precisely. Intuitively, increasing the number of anchor nodes will improve the precision of the virtual coordinates. This has also been proven by DV-Hop based localization algorithms such as in [73], [74], and [75]. The challenge, however, is to reduce computation and memory usage which are limited in sensor nodes. The naive combination of greedily checking the distance to the routing trajectory and the use of the virtual coordinate system with many anchor nodes is not practical. Because it not only requires computational resources, but also error-prone due to the use of the estimated location. According to the DV-Hop based localization algorithms, in the worst case, the error rate can be as large as 45% of the range of the radio [24], which could lead to routing failure.

2.3. ENERGY EFFICIENT DUTY CYCLE MECHANISMS

The Energy efficient duty cycle mechanisms saves energy of WSNs by putting sensors to sleep periodically. To lower the duty cycle, sensors should sleep majority of the time and awake up fast. In the active period, the sensors need to finish the optimized work fast to reduce the energy consumption. Also, to reduce the collision of the radio communication in the WSNs, sensors need to avoid transmitting data simultaneously in the same region. In the work S-MAC [76], T-MAC [77], and Z-Mac [78], they maintain the synchronized time slots which can be much bigger than normal time-division multiple access (TDMA) slots. S-MAC, T-MAC employ request to send (RTS)/clear to send (CTS) mechanism (Nodes maintain periodic duty cycle to listen for channel activities and transmit data) in case of synchronization failures. As these protocols use RTS/CTS, the overhead of the protocols is quite high because most data packets in sensor networks are small. To reduce the overhead, instead of using RTS/CTS, the Z-MAC [78] adopted a technique from RTP(real-time transport protocol) and developed a local synchronous protocol that the control message transmission rate is limited to a small fraction of session bandwidth and each session member adjusts its sending rate of control messages according to the allocated session bandwidth. In its local synchronization protocol, each data sender transmits a synchronization message containing its current clock value periodically. When a node receives a synchronization message, it updates its clock value by taking a weighted moving average of its current value and the newly received value.

B-Mac [79], which is a light weight carrier sense multiple access (CSSA) MAC protocol, is the default MAC protocol of Mica2 sensor. It also adopt low power listening (LPL) and engineer the clear channel sensing (CCA) technique to improve channel utilization. The LPL require the sender broadcast a preamble which lasts one cycle (sleep+active) before actual sending the data. After a receiver wake up and overhear the preamble message, it

keeps awake until it receives all the data. The clear channel sensing technique exploits the ambient noise changes depending on environment. If the channel is clear then the sender will transmit while the channel is busy it will backoff (wait for a small interval and retry).

2.4. SIMULATING WSNS

To demonstrate the effectiveness of a compression algorithm or a routing protocol in large scale WSNs, many researchers have simulated WSNs under different conditions to gather statistics on the performance. These simulations require input datasets that describe the time-evolving nature of the network topology. In this section, available datasets and models needed to simulate a WSN are listed and described.

2.4.1. Simulation System Properties of TOSSIM. The TOSSIM simulation system has the following four properties: Scalability, Completeness, Fidelity, and Bridging.

The scalability means the simulation tool need to be able to handle large scale networks. TOSSIM is an event driven simulator which handle network event in an asynchronous queue that is able to handle large scale network's message exchange simulation. Also, it loads the WSN topology by reading a directed graph file which can handle thousands of vertexes and links. The completeness requires the simulator to cover as many system interactions as possible. The TOSSIM could fully simulate the data linked layer. It also allow simulation of communication through PC to simulating WSNs. The user can also choose external radio to verify their own prototypes. The fidelity means the simulator is able to capture the behavior of the network at a fine grain. To improve the fidelity of the simulation, TOSSIM considering real-world condition that affect the radio communication like the environment noise and the radio contention caused in packet-level Interactions. The bridging is a property that the user can effortless to test and verify code in real hardware. The TOSSIM is designed for TinyOS and can simulate real-world nesC programming running in real sensor motes.

2.4.2. Energy Simulation and Energy Model. To simulate the energy usage, PowerTossim-Z [80] is used as a plugin for TOSSIM simulation. It use the micaZ power model that precisely simulate the energy usage including CPU power, and Radio receive and radio transmission power usage. It works as a plugin of TOSSIM and needs a data parser to parse the simulation outputs.

2.5. TARGET TRACKING

Target tracking is a complex task including target trajectory prediction, trajectory encoding, and real-time data dissemination.

2.5.1. Trajectory Prediction in WSNs. In a moving object tracking problem, the key objective is the dynamic sensor tracking schedule to predict the trajectory that ensures the real-time performance of object detection and tracking. When WSNs operate in low-power-listening (LPL) mode, the radio communication latency equals half of the duty cycle times hops counts as shown in the following Equation. $Delay_{send}^{rcv} = \frac{H_{send}^{rcv} \times T_{clc}}{2}$ where $Delay_{send}^{rcv}$ is the routing delay in sending a packet from a sender to a receiver, H_{send}^{rcv} is the hop counts from a sender to a receiver, and T_{clc} is the average duty cycle of the current scheduling protocol. The prediction-based methods are used to predict the location of the mobile object after $Delay_{send}^{rcv}$ time based on historical data. Therefore, the sensor states (active, sleep) can be prepared before the target enters/leaves the area. Linear prediction is a simple prediction approach, which depends only on the previous location of the target [81]. However, linear prediction suffers from low prediction accuracy.

To improve the prediction accuracy, particle filter [82] and Kalman filter [83], [84], [85] based prediction frameworks have been proposed. Kalman filter is a linear algorithm that exploits a series of data observed overtime to boost the prediction's precision. In paper [86], the authors proposed a Kalman filter based generalized regression neural network that not only reduced the prediction error but also improved the prediction speed by combining

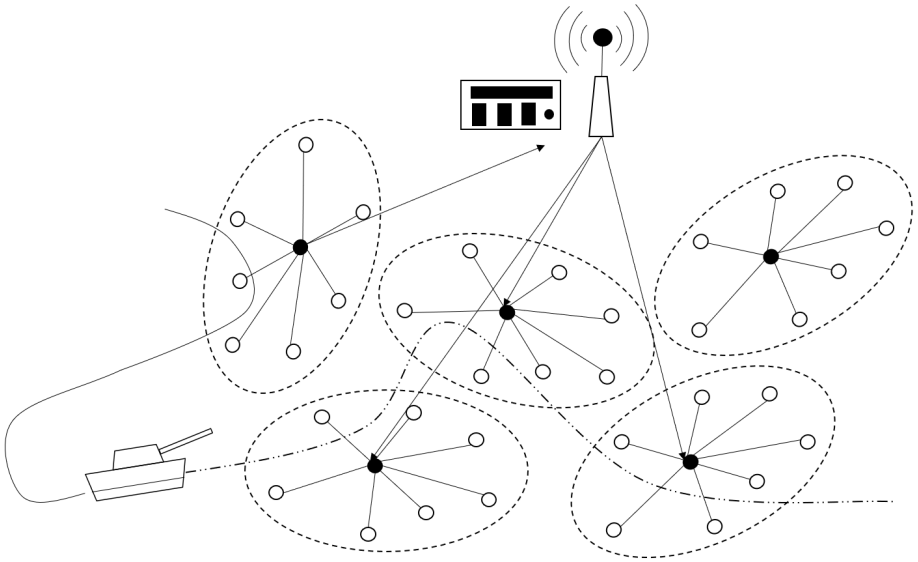


Figure 2.9. Cluster-based Object tracking protocol

the Kalman filter with neural networks. The particle filter based frameworks like [87] [88] [89] are also widely used in the target tracking as they are suitable for nonlinear systems. There are also works like [90] that combines both Kalman filter and Particle filter based frameworks to get reliable location prediction for real-world applications. Although the previous prediction based target tracking approaches could achieve good prediction speed and precision, they still rely on the GPS data or RSSI values, and can not directly generate the control message which can directly control the local sensors. The predicted trajectory needs to be processed by the server that has the knowledge of all the sensors' locations which has a higher risk of leaking the location privacy.

2.5.2. Cluster-based Object Tracking Algorithms. Cluster-based object tracking protocols are so popular that some researchers classified the previous works into only two groups: cluster-based and non-cluster-based. It is the most realistic solution that could control the message flow in large scale WSNs.

In cluster-based protocols, cluster heads, which are selected by different cluster algorithms, are responsible for collecting information from the nodes in their cluster, communicating with sinks, and propagating the control messages to their cluster members. In

this way, a large scale WSN is simplified to a small sink-cluster heads network with many small cluster head - slave sensors sub-networks. For object tracking applications, once slave sensors in any cluster detect the object they report the target's information to their cluster head. The cluster head then routes the information to the sink or the local mobile edge server.

For saving energy, as we have discussed before, the server predicts the target's trajectory and sends the control messages to the cluster heads which reside on the target's trajectory. Those cluster heads then propagate the active/sleep messages to all or some members when the target enters/leaves. Therefore, most of the sensors that far from the target's trajectory could fall into sleep to save energy and bandwidth.

For example, in Figure 2.9, the sink of a cluster-based WSN collects the target's current location from a cluster-head. After predicting the target's future trajectory, the sink sends control messages to the cluster heads that reside on the target's future path. The cluster-head then controls the related sensors for detecting the target. Although the linear prediction model could predict the target's movement well, the centralized cluster-based approaches can't deliver the wake-up (for tracking) and reset messages (for putting sensors back to low power listening (LPL) mode) efficiently.

The drawback of the cluster-based object tracking protocols is the overhead of generating the dynamic cluster [91] and maintaining the clusters. Also, for different applications, all the nodes in the WSN need to tune their program to meet specific routing and clustering requirements. However, it is usually not practical for large scale WSNs.

Considering the above drawbacks of the cluster-based object tracking protocols, we choose to use the DV-Hop (which stands for distance vector of hops) based packet routing protocol [29] that decouples the data plane (network layer) and the control plane of the IoT network. Like software-defined networks, the DV-Hop based routing rules are encapsulated in each routing packet. So different applications could share the same WSN by just creating their own routing rules.

2.5.3. Counter-based Broadcast. Broadcasting is the fastest way to flood a message into the whole WSN. However, limited bandwidth causes a delay in broadcasting a sequence of messages into the network. After a node receives a given packet, the counter-based broadcasting schemes [52][54][55] require a node to wait for a short period to listen to its neighbors and count how many times the given packet has been rebroadcasted. If the broadcast count of the given packet reaches the predefined threshold, it will drop the packet. Thus, only a few of the nodes in the network will rebroadcast the given packet which saves bandwidth and thus, alleviates the congestion.

PAPER

I. EFFICIENT Z-ORDER ENCODING BASED MULTI-MODALDATA COMPRESSION IN WSNS

Xiaofei Cao, Sanjay Madria, Takahiro Hara

ABSTRACT

Wireless sensor networks have significant limitations in available bandwidth and energy. The limited bandwidth in sensor networks can cause higher message delivery latency in applications such as monitoring poisonous gas leak. In such applications, there are multi-modal sensors whose values such as temperature, gas concentration, location and CO2 level need to be transmitted together for faster detection and timely assessment of gas leak. In this paper, we propose novel Z-order based data compression schemes (Z-compression) to reduce energy and save bandwidth without increasing the message delivery latency. Instead of using the popular Huffman tree style based encoding, Z-compression uses Z-order encoding to map the multidimensional sensing data into one-dimensional binary stream transmitted using a single packet. Our experimental evaluations using real-world data sets show that Z-compression has a much better compression ratio, energy saving, streaming rate than known schemes like LEC (and adaptive LEC), FELACS and TinyPack for multi-modal sensor data.

Keywords: Sensor network, Data compression, Z-order encode

1. INTRODUCTION

Wireless sensor networks (WSNs) are being developed for a plethora of emerging applications in wide range of disciplines. For example, there are near real-time sensor-cloud applications [1] to perform multi-modal sensing tasks. Some military applications of tracking hostile objects or monitoring intruders use multiple sensing units to provide precise location and speed by applying multi-sensor data fusion. The unmanned vehicles [2] need GPS and accelerometer to locate themselves, and to track distance and height of objects using the camera, laser range meter or radar. By fusing these multi-modal sensor data, they can also predict the moving trajectory of the nearby objects. Similarly, environmental monitoring applications need temperature, wind direction, humidity, CO2 level, etc. Many of these multi-modal sensor applications asking for rigorous data integrity as well as high data stream rate, and therefore, cannot tolerate high latency due to limited link bandwidth in WSNs. Since batteries are the typical power source for wireless sensors and cannot easily be changed, the energy consumption is another primary constraint in the design of multi-modal WSNs. Many research efforts have shown that radio communication is the predominant factor in all energy consumption metrics of the WSNs. Thus, there is a need for lossless data compression algorithms which could reduce the size of multi-modal sensing data thereby, decreasing the radio communication.

Some sensor data aggregation approaches [3][4] could save energy and bandwidth by reducing the number of packets to be transmitted. However, the data aggregating approaches cannot guarantee the data integrity because of the lossy process and outlier data. Also, when aggregating, the outlier detection is expensive and could introduce delay [5]. Model based compression algorithms [6] such as APCA[7], PWLH[8], SF[9] also have good compression ratio. However, they can not work on lossless applications as they approximate the data with temporal and spatial locality. The compressive sensing approaches have the same drawback that may lose data Integrity thus, not suitable for multi-modal lossless data compression.

Existing works [10][11][12] propose the lossless compression algorithms for sensors, using Huffman coding, that exploit temporal locality of the data of the WSNs. Instead of storing and transmitting the complete data, [10][11][12][13] use the difference in value, called delta value, between two adjacent timestamp readings and usually, it needs fewer bits to represent a delta value than a complete value. Based on Shannon entropy theory [14] and the Huffman coding[15], the most frequent values are assigned a shorter code than the less frequent values. Thus, if dataset is drawn from a smaller set of values, the entropy will be smaller. However, the standard Huffman and adaptive Huffman [16] coding have larger overhead on RAM in storing the Huffman trees generated dynamically based on the frequency of the data. Also, to decode the compressed data, every node in the wireless sensor network needs to have a copy of the tree. However, as we know that WSNs have limited bandwidth and energy, synchronizing the Huffman tree is impractical. LEC and adaptive LEC [10][11] successfully adapted Huffman coding for its static initial code library which is a predefined Huffman tree. That way, WSNs do not require transmitting the entire Huffman tree. Similarly, TinyPack [12] modified its initial code library based on LEC's and also proposed algorithms adapting library to different types of sensor applications.

To address the problem discussed above and improving the applicability and compression ratio of existing algorithms, we propose a Z-order encoding based compression algorithm. We also compare the performance of our work with LEC, Adaptive-LEC, FELACS and TinyPack (since they outperform other algorithms like ASTC [17], S-LZW [18] and GAMPS[19]). [20] proposed their coding dictionary based on a very specific dataset, thus, not considered in our experiments. In summary, this paper makes the following contributions:

- Design of an efficient multi-modal sensor data compression scheme that combined the Z-order encoding with delta compression. To our knowledge, this is first attempt to propose multi-modal lossless data compression algorithms for WSNs. Briefly, our main contributions are as follows.

- Improve Z-order encoding by integrating a static initial dictionary and an odd bit Z-order encoding for further performance improvement in WSNs.
- Design a data concatenation scheme which can concatenate leaf nodes data efficiently for compression.
- Perform extensive simulations using TinyOS and TOSSIM and compare the performance with most recent and popular sensor data compression algorithms referenced above. The results show that our schemes outperform these considering the compression ratio, streaming rate and energy efficiency as the metrics.

2. BACKGROUND AND RELATED WORK

2.1. LEC, TINYPACK, AND ADAPTIVE LEC

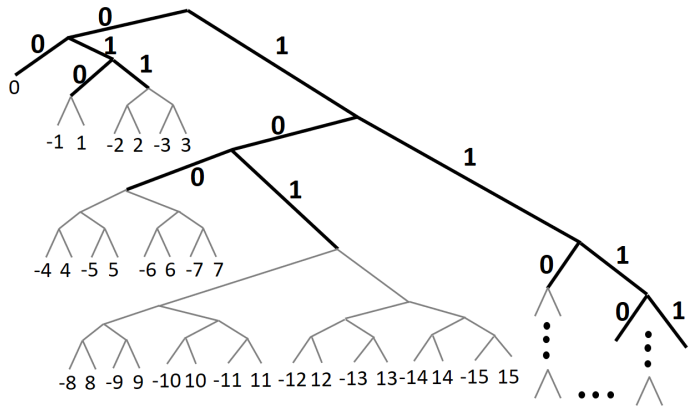
LEC[10] and TinyPack[12] are both Huffman coding based delta compression algorithm. The Huffman coding would represent higher frequency symbols with less number of bits. However, in sensor networks, delta values can range from 0 to more than 2^{16}. Thus, the memory limitation of nodes makes creating such large Huffman coding dictionary impossible. In their improved approaches to save memory and to make them computationally efficient, instead of adapting Huffman coding to each symbol, LEC and TinyPack create the Huffman coding for the length of the delta value in a fixed pattern that matches the frequency of the data length group which they assume is decreasing when the length of delta value increases. Thus, in both LEC and TinyPack, the length of Huffman coded prefix increases with the increasing delta values length. The Huffman tree of LEC is shown in Figure 1a and the TinyPack code is shown in Figure 1b. These two coding focused on the different distribution of the delta value of the dataset. For example, if a dataset contains many data

of only one-bit length, according to Figure 2, TinyPack will have better compression ratio than LEC whereas, for other delta values larger than 3, LEC will have better performance than TinyPack.

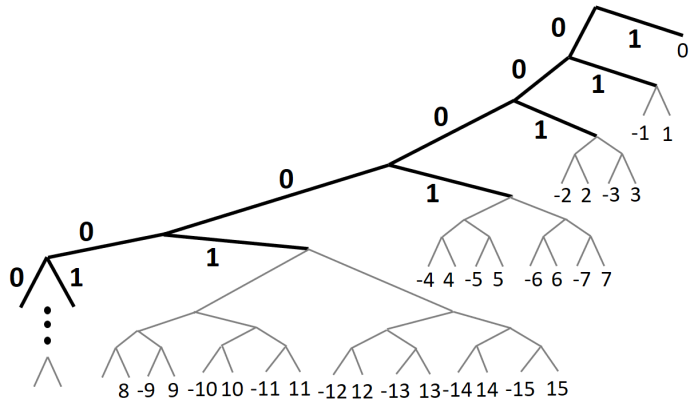
However, as we have discussed before, the distribution of the delta values can be varied in different sensor network applications. Even in the same application under different time periods or in different sensor nodes, the distribution is different. Thus, in these situations, LEC and TinyPack do not show good performance as they have fixed code. To solve this problem, Adaptive-LEC and TinyPack-DP(TP-DP) can adapt the prefix code when the distribution of the frequency of the delta value length changes. The Adaptive-LEC will adapt its prefix code for every new data while the TP-DP only changes its prefix code at the beginning of each new frame. The efficiency of Adaptive-LEC initial code is shown in Figure 2. They define the length of delta value with the most frequency as the frequency center of data and define the length of delta value with the minimum prefix code as the code center. When the current frequency center of data is drifted from the previous code center, the prefix code will shift to meet the current frequency center of the data. For some dataset with two frequency centers, they proposed another initial code with two code center and two adaptive sections where the prefix codes will be adapted independently based on sections.

2.2. FAST EFFICIENCY LOSSLESS ADAPTIVE COMPRESSION SCHEMA

The idea of fast efficiency lossless adaptive compression schema (FELACS) [13] is to give every compressing data a fixed b bits. Every value smaller than 2^b will be filled with '0' at the front to reach b bits, added '1' bit at the front then directly appended to the output stream. Every value larger than or equal to 2^b will be cut into two sections. The higher bits section is encoded using unary coding and is appended to the output stream. The lower bits section has the length b directly appended to the output stream. The fixed bits b is generated by calculating the average number of the input data.



(a) Huffman tree of LEC code



(b) Huffman tree of TinyPack initial code

Figure 1. Huffman tree of LEC and TinyPack initial code

3. PROPOSED Z-COMPRESSION APPROACH

Z-compression is a lossless compression algorithm that exploits temporal locality. The wireless sensor nodes use the delta value of each attribute as the input of the compression algorithm. The delta value is calculated using Equation 1 where V_c is the current delta value, V_p is the previous delta value and P is the resolution of the sensors.

$$d_\Delta = \frac{V_c - V_p}{P} \tag{1}$$

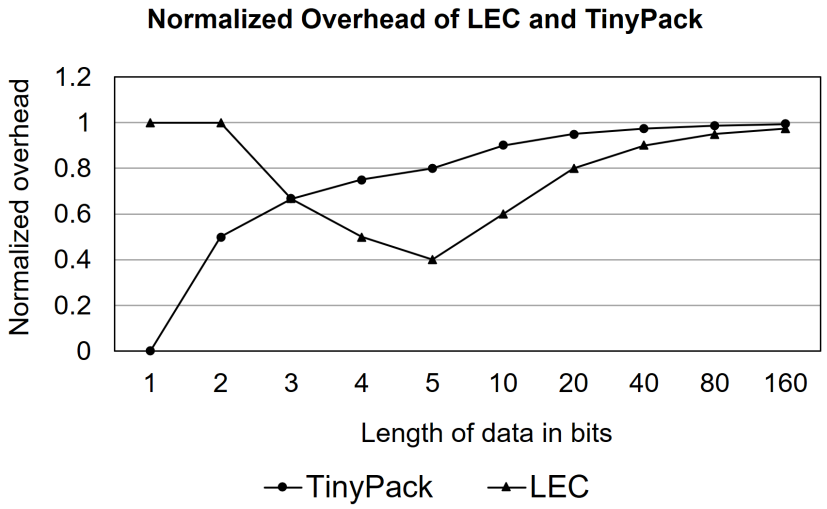


Figure 2. Normalized compression overhead of LEC and TinyPack

3.1. NAIVE MULTI-DIMENSIONAL Z-COMPRESSION FOR SENSOR VALUES

The Naive Z-compression uses Z-order encode [21] and all-is-well scheme [12]. As shown in Figure 3, the Z-order encode interleaves input data bit by bit and output a new binary number with a length double of the largest input.

$$V = \begin{cases} 2 \times V_{signed}, & \text{if } V_{signed} > 0 \\ 1, & \text{if } V_{signed} = 0 \\ 1 - 2 \times V_{signed}, & \text{if } V_{signed} < 0 \end{cases} \tag{2}$$

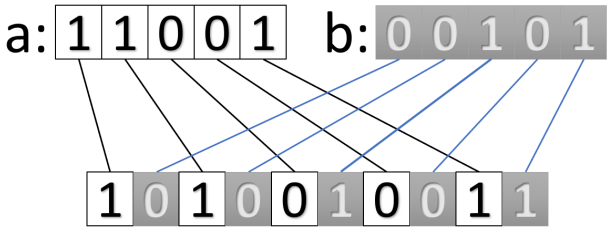


Figure 3. Procedure of Z order encoding

We choose the byte array as the data structure to store the encoded data. Also, we use unsigned integer V to represent both positive, negative and zero values $V_{signed}$by using Equation 2. Next, we add '1' at the front of the output to protect the possible '0'. We also integrate an all-is-well bit [12] with Z-order encode. It sets the compressed data to be zero if all delta values of the input attributes are zero. We set the result to be an all-is-well bit when nothing changes. The implementation is by directly checking all the input values. If all the inputs equal to '1' then output '1' where '1' equals to zero according to Equation 2.

Different from the Huffman tree based compression algorithms that need to know the probability distribution of the input data to achieve the best performance, Z-compression only exploits the relationship between the length of delta value of the attributes. We then compare the normalized overhead of Naive Z-compression, LEC, and TinyPack on compressing the uniformly distributed multi-modal random data as shown in Figure 4.

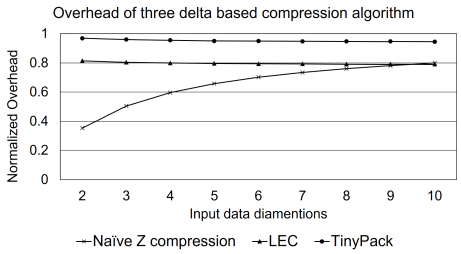


Figure 4. Normalized compression overhead compressing multi-dimensional data

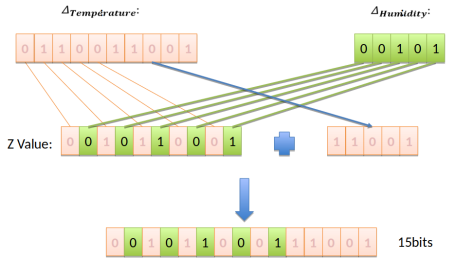


Figure 5. An example of optimized Z-compression

The result shows that Naive Z-compression performs well when the number of attributes is smaller. However, with the increase in number of attributes, the overhead of Z-encode increases. When the number of attributes is larger than eight, the overhead of Z-encode is greater than that of LEC. To address this drawback, we next propose an Optimized two-dimensional Z-compression scheme which guarantees to have better compression ratio than TinyPack and LEC when compressing two-dimensional data. Thus, for multi-dimensional data, we can split it into several groups with two or more attributes to minimize the overhead due to extra bits.

3.2. OPTIMIZED TWO-DIMENSIONAL Z-COMPRESSION ALGORITHM

Before improving the Naive Z-compression, we need to study two important properties of Two-dimensional Z-order encoding. Here, we do not consider the extra '1' bit added before the compression result.

- The number of bits in the output of Two-dimensional Z-order encoding is always even.
- There must be at least one of '1' in the first two bits of the output of the Two-dimensional Z-order encoding in binary format.

The first property indicates a way to improve the Naive Z-compression. We can use the value with odd length to represent the 'skewed' data. Here, we define a two-dimensional dataset as skewed when the number of bits of the larger delta value is more than two times of the number of bits of the smaller as in Equation 7. When the data is skewed, in order to make the length of Z-value odd, we divide the larger value V_L with length B_l into two values V_{L1} and V_{L2} of length B_{l1} and B_{l2} using Equation 6a,6b,6c. Note that the notation '<<' and '>>' means shift left and shift right for certain bits.

$$Mask_{l2} = (1 << B_{l2}) - 1, \tag{3a}$$

$$V_{L1} = V_L >> B_{l2}, \tag{3b}$$

$$V_{L2} = V_L \& Mask_{l2}; \tag{3c}$$

We can apply Z-order encoding on V_{L1} and V_S which give us the Z value Z_{half}. Then by appending V_{L2} on Z_{half}, shown in Equation 8, we get the Z value for the skewed data. The notation $\otimes$ means applying Z-order encoding and $\oplus$ means concatenating the two binary strings together. In the above procedure, we have to ensure that the Z value has odd length L_{odd} and each different two-dimensional data maps to a unique Z value.

Algorithm 1 shows the procedure of the Optimized two-dimensional Z-compression. It reduces the extra bits adding to the smaller delta value when the delta value pair is skewed. However, when the delta value pair is not skewed, Naive Z-encode is applied. For example, when encoding two attributes '101' and '11110000', we add two '0' bits to the first attribute '101' which makes it '00101' and set the end pointer of the second attribute to be 3 which divides it into two values, '11110' and '000'. Then we apply Naive Z-compression on '00101' and '11110' which produces '0101110110'. In the end, we append the remaining '000' to it and add '1' to the leftmost bit which gives us the encoded Z-value of '10101110110000' which is 14 bits long.

$$B_s < floor(B_l/2) \tag{4}$$

$$Z = Z_{half} \oplus V_{L2} = (V_{L1} \otimes V_S) \oplus V_{L2} \tag{5}$$

When decoding, we use the second property of Two-dimensional Z-order encoding to help us figure out which value within the two decoded data is larger. That is, the value with less '0's at the front is larger. Then we append the remaining data to it to get the final result. For example, when decoding '10101110110000', we first count the length of the Z value after the first '1' which give us 13. This odd value 13 indicates that the delta value pair is skewed. Then using $13\%6 = 1$, we know that it belongs to the first case in Algorithm

1 that B_L%4==0. And, using $n = 13/2 = 2$, we can divide the Z value into two parts at $2 \times (2n + 1) = 10$ bits from left. Interleaving the bits from the first part, we get '00101' and '11110'. In the end, we append the remaining '000' to the second value as it has less '0' bits at the front than the first one so the decoder will output two binary values '00101' and '11110000'.

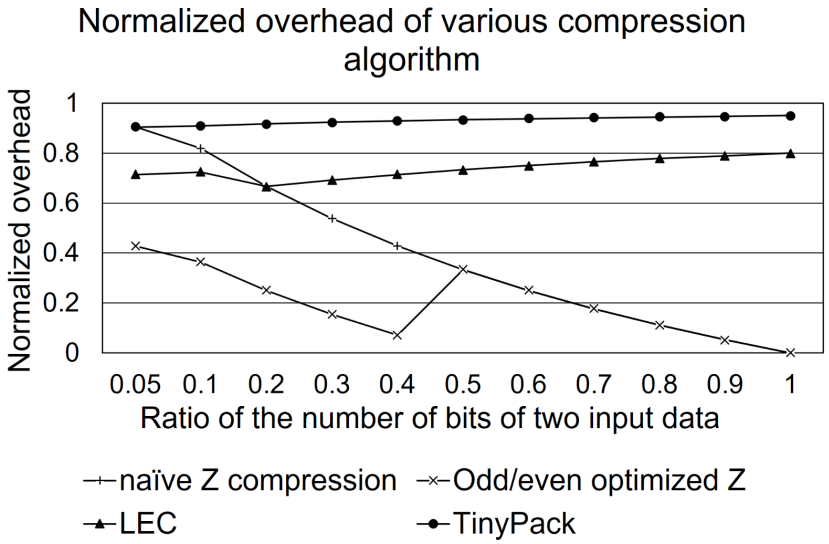


Figure 6. Normalized overhead in compressing two attributes where the largest attribute's data is 20 bits long

In Figure 6, We compare the normalized overhead of extra bits of four different compression algorithms which includes LEC, TinyPack, Naive Z-compression and Optimized Z-compression on compressing two-attributes data where the larger attribute is 20-bits long. The X-axis is the ratio of the number of bits of the input delta value pair. The Y-axis is the normalized overhead which is the extra bits over the total bits that the input delta value pair has. We can see that the Optimized Z-compression dwindle the extra bits significantly. The maximum normalized overhead is 0.43 which is half of the normalized overhead of

TinyPack. In Optimized Z-compression using odd/even bit optimization, we use two as the critical ratio of two attributes' length to trigger the Optimized Z-compression as it is easier to implement.

Algorithm 1: Optimized Two-dimensional Z-compression algorithm using odd/even bits optimization

input : Delta value V_L and V_S where $V_L > V_S$
output Z value
:

initialization $B_L \leftarrow Length(V_L)$, $B_S \leftarrow Length(V_S)$
if $B_S \geq floor(B_L/2)$ **then**
    $Z = V_L \otimes V_S$;
else
    if $B_L\%4 == 0$ **then**
        $n \leftarrow B_L/4$;
        Divide V_L into V_{L1} and V_{L2} where $V_L = V_{L1} \oplus V_{L2}$;
        $Length(V_{L1}) = 2n + 1$ and $Length(V_{L2}) = 2n - 1$;
        $Z = (V_{L1} \otimes V_S) \oplus V_{L2}$;
    else if $B_L\%4 == 2$ **then**
        $n \leftarrow B_L/4$;
        Divide V_L into V_{L1} and V_{L2} where $V_L = V_{L1} \oplus V_{L2}$;
        $Length(V_{L1}) = 2n + 1$ and $Length(V_{L2}) = 2n + 1$;
        $Z = (V_{L1} \otimes V_S) \oplus V_{L2}$;
    else if $B_L\%4 == 3$ **then**
        $n \leftarrow B_L/4$;
        Divide V_L into V_{L1} and V_{L2} where $V_L = V_{L1} \oplus V_{L2}$;
        $Length(V_{L1}) = 2n + 2$ and $Length(V_{L2}) = 2n + 1$;
        $Z = (V_{L1} \otimes V_S) \oplus V_{L2}$;
    else
        $n \leftarrow B_L/4$;
        $V_L =' 0' \oplus V_L$;
        Divide V_L into V_{L1} and V_{L2} where $Length(V_{L1}) = 2n + 1$ and $Length(V_{L2}) = 2n + 1$;
        $Z = (V_{L1} \otimes V_S) \oplus V_{L2}$;
return $Z =' 1' \oplus Z$;

We then compare another optimization scheme that can apply Optimized Z-compression on two input attributes with any ratio of lengths. That is, use a fixed small bits as a ratio indicator to indicate the actual ratio of two attributes' lengths. However, extra control bits

can cause extra overhead. For example, when using two bits indicator, four different ratio which are $\frac{3}{2}$, $\frac{3}{1}$,$\frac{2}{3}$, and $\frac{1}{3}$, can be indicated using '00', '01', '10', '11'. We can then interleave binary values in a fixed ratio. For example, when the ratio is set to be $\frac{3}{2}$, we should interleave 3 bits at a time for the first delta value and 2 bits at a time for the second delta value.

We changed the ratio indicator bits from 2 to 4 bits and tested it on a uniformly distributed random data set. Figure 7 shows that with the increasing length of input data, the average overhead of different optimized Z-compression also increases. However, the odd/even checking based optimization performs better than others when the input's maximum length is fewer than 45 bits which is larger than the size of most wireless sensor's data sensing output. Thus, when compressing real-time sensing data packets, we suggest only to use Algorithm 1.

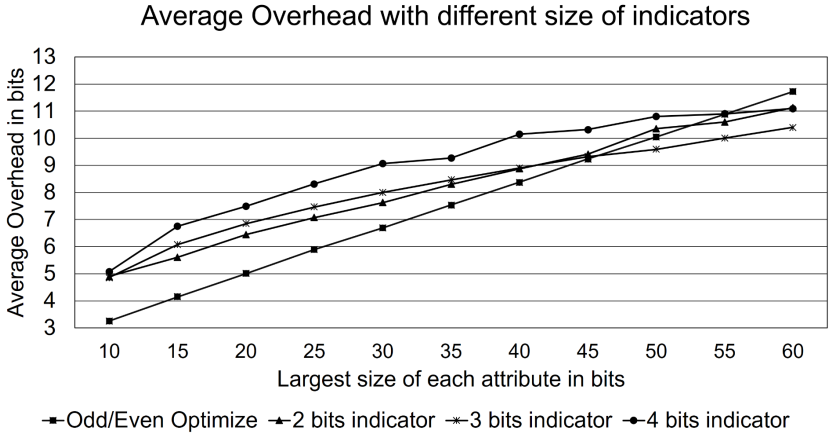


Figure 7. Average overhead in compressing data with two attributes

3.3. SMALL CODE LIBRARY ADD-ON

We can further improve the Optimized two-dimensional Z-compression scheme by integrating a small code library. This library will enhance the compression performance without affecting the correctness. The code library is shown in Table 1. For entries in the

small library, one of the input delta value pairs should have '0' or '1'. We assign the output a smaller value which never appears in the result of Naive or Optimized Z-compression. For example, when we compress two delta values where both the values are 0, the output of encoding value will be 1-bit '1' instead of 2-bits '11' giving 50% improvement. Also, this small code library will not affect the correctness of the encoding and decoding procedures in Algorithm 1 because Algorithm 1 will not generate Z values that the small library has. When decoding, we can check the small library before checking the length of Z value to get the correct decoding result.

Table 1. Initial small code library

Value 1	Value 2	Z value
0	0	1
0	1	11
1	0	10
0	-1	111
-1	0	110
0	V_2 >31 || V_2 <-31	10000⊕V
V_1 >31 || V_1 <-31	0	100000⊕V

3.4. OPTIMIZED N-DIMENSIONAL Z-COMPRESSION ALGORITHM

The Optimized N-dimensional Z-compression algorithm combines the procedures discussed in 3.A, 3.B, and 3.C using a predefined rule which is generated in Algorithm 3. When compressing multi-dimensional sensing data, We can either use Naive Z-encoding based compression on all the attributes or use Optimized Z-compression on the pairs of attributes and then merge the result. Testing all the combinations of the input attributes with above two encoding methods, Naive Z-compression and Optimized two-dimensional Z-compression is an NP-complete problem. We propose an approximate algorithm using two pointers and a local greedy approach to find the approximate combination result in

Algorithm 3. We use a two dimensional array as GroupMember to represent the attributes ID and their length. We use another array as Group to store the list of GroupMembers. The input of the algorithm is a list of GroupMembers which represent all the attributes. The output of the algorithm is a list of Groups which instructs the encoding and decoding procedure.

Algorithm 2: Rule Generation Algorithm

```
   input : List of GroupMembers: GML
   output List of Groups: GL
   :
 1 Sort(GML); //sorting based on the length of attributes
 2 LP← 0; //left pointer starting from the left end
 3 RP ← GML.getSize(); //right pointer initialization
 4 while (length(GML[LP])<length(GML[RP-1])/2&&LP<RP) do
 5 |   LP++,RP- -; //find pairs meeting Equation 2
 6 //add groups of the pairs to the output list
 7 while RP<GML.getSize() do
 8 |   GL.add(new Group{GML[RP++],GML[GML.getSize()-RP]}
 9 bufGroup=new  Group{ }//initial a new group
10 //add rest GroupMembers to the Group
11 for (i=LP;i<GML.getSize()-LP;i++) do
12 |   bufGroup.add(GML[i]);
13 GL.add(bufGroup);
14 return GL;
```

The encoder will use the Group information to help them encoding. If a Group contains more than three entries, the encoder will use the Z-order encoding to compress the attributes represented by the group. If a Group contains only two attributes, the encoder will use Algorithm 1 and Table 1 to compress them. At last, the sensor will further encode all the encoded values and output the result. For example, if there is a Group [{1,7},{2,5},{4,9}], the wireless sensor node will do Z-order encoding on the attributes with field ID 1,2 and 4. If there is another Group [{3,3},{5,10}], the sensor will do the optimized two-dimensional Z-compression on the attributes with field ID 3 and 5. Then the node will compress the two

encoded value using optimized Z-compression as there is only two group. After transmitting the compressed value to the decoder, the decoder will decode the Z value reversely based on the number of groups and the fields ID in each group.

Algorithm 3: Lossless data concatenating algorithm

```
   input : Q, N, b_max, b_min, sum, L
   output payload: out[]
   :
 1 Index=new byte[N], ind=1, prev=L[0][1];
 2 Sort(L); //sorting is based on L[][0]
 3 for (i=0;i<N;i++) do
 4     Index[L[i][1]]=ind;
 5     ind+=prev;
 6     prev=L[i][1];
 7 prev=L[0][1];
 8 for (i=0;i<N;i++) do
 9     buf=Q.poll();
10     for (j=prev-L[i][1];j<L[i][1];j++) do
11         out[Index[i] + j]=buf[j]
12     prev=L[i][1];
13 out[0]=N;
14 return out;
```

3.5. LOSSLESS DATA CONCATENATING ALGORITHM

The main reason that we need to concatenate the local compressed packets is to save energy and bandwidth usage. In our experiments and also, in the previous experimental analysis of radio performance[22] [23], we found that reducing the payload size of leaf nodes'packets will not give us much energy saving. Also, a leaf node is not the bottleneck in WSNs as there is not much traffic via them. However, in the experiment, we found that the intermediate nodes are the bottleneck in WSNs as they not only need to sense data but also need to route packets from the lower level nodes to the higher level nodes. The energy consumption rate and bandwidth occupation by the intermediate nodes especially the root is much more critical than the leaf nodes. Thus, to save energy and prolong the lifetime

Table 2. Fields of Experimental Dataset

data set name	number of fields	fields label
Intel Berkeley Lab environment data	6	epoch, node ID, Temperature, Humidity, Light, Voltage
Accelerator in moving car	5	epoch, node ID, X, Y, Z
ZebraNet data	5	epoch, node ID, Longitude, Latitude, Voltage
Vehicle trace data (V to V)	10	epoch, node ID, $Longitude$, $Latitude$, $Altitude$, $speed$ of two vehicles

of WSNs, we only need to consider the energy consumption of the node with the most radio load. To do that, we need to concatenate the upstream data to decrease the number of packets each intermediate node will transmit.

$$PacketSizeLimit < (sum + b_{max} - b_{min}) \tag{6}$$

Next, we discuss how we can concatenate the local compressed packets. The compressed data are in the byte array format with variable length. It has node ID and timestamp as their primary key. It is not practical to decode and re-encode all the data in the intermediate nodes as it will increase the RAM and CPU load as well as cause delays. Here, we propose a data concatenating algorithm which will concatenate byte array input data efficiently. The idea is that, in a packet of intermediate node, the compressed data with larger number of bytes is always in front of the compressed data with smaller number of bytes. However, the compressed data with smaller number of bytes will be filled with zero bytes to make them have the same number of bytes as their neighbor in front of them. At the end, we set the first byte of the output packet the number of bytes of the largest data. When decoding, we first read the length of the largest packets b_{max} at first byte. Then we read the following $2b_{max}$ bytes to extract the first and the second data. If there are empty bytes in front of the second data we need to update the b_{max} by subtract the number of empty byte from b_{max}. Next, we use the updated b_{max} to extract the third data and update the b_{max}

again. Repeating doing that until the last byte of the packet, we can extract all the samples in the packets. Note that we use a queue Q to store all the upstream packets' payload, a two-dimensional array L to store the size of each payload and their index in Q, an index array $Index$ to help us append the data in the queue to the output byte array. We also need to track the largest packet length b_{max}, the smallest packet length b_{min} and the total length of all the packets $sum = \sum_{i=1}^{n} b_i$ in the queue. Once the criteria in Equation 9 is satisfied, we use Algorithm 4 to concatenate the elements in the payload queue and create a large packet. The total number of data needs to be concatenate is 'N' which exclude the last item in the queue. Then the intermediate node will transmit the large packet to its downstream node.

4. EXPERIMENTS AND EVALUATIONS

4.1. EXPERIMENTAL SETUP AND CONFIGURATIONS

To demonstrate the effectiveness of our proposed Z-compression scheme in real-world situation, we tested it against different types of real-world multi-modal data sets such as GPS data [24], environmental data[25], Accelerometer data[26] and vehicle trace data[27] from real projects. The attributes in each dataset are shown in Table 4. Two common attributes which both data sets share are timestamp and node ID. These two attributes are used as the primary key of the local packet. We cannot compress the primary key because the intermediate nodes need to identify where the packet has come from and when the sampling starts. We assign the timestamp fixed two bytes and the node ID fixed one byte in the local packet, and use a variable length for the compressed sensing values. The compression ratio CR in the performance matrix is defined as the compressed data length over the size of uncompressed data shown in Equation 22. We use a base station to accumulate the number of bytes of all n compressed packets. $\sum_1^n L_{compressed}$ is the compressed data size and $\sum_1^n L_{original}$ is the original data size.

$$CR = \frac{\sum_1^n L_{original}}{\sum_1^n L_{compressed}} \tag{7}$$

To find out the energy cost of the intermediate nodes, we use PowerTOSSIM-Z to simulate the energy consumption in WSN. The tool will calculate the CPU cycle, and radio usage at each node. It then uses the predefined power model to generate the power consumption at each node in the experiment. Data are inserted into the leaf nodes using python script.

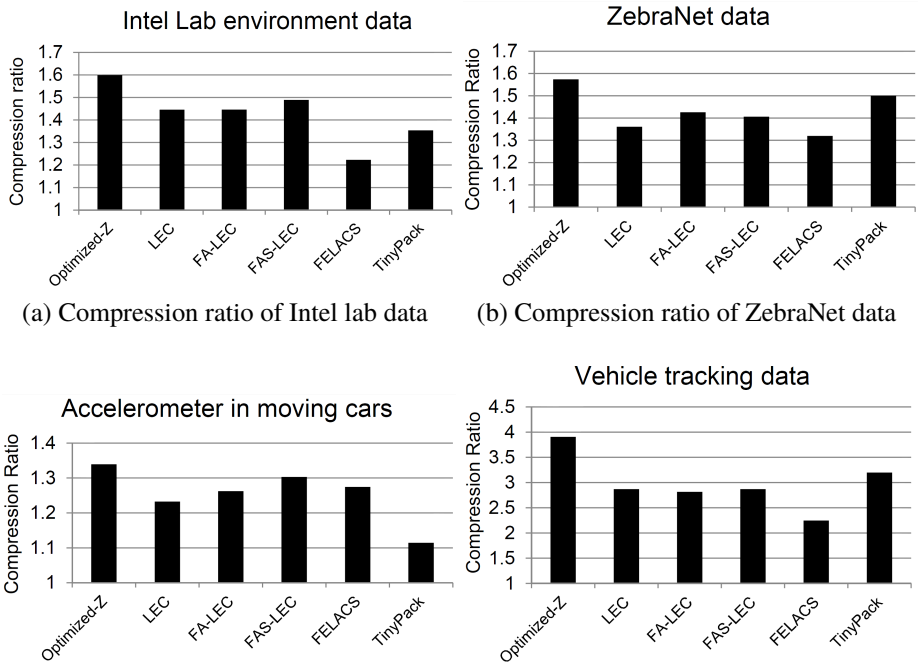
We notice that when we increase the sampling rate in the WSN, close to a certain interval of two consecutive sensing sample, the packet drops starts happening at some intermediate nodes and the sink node. Here, we define the sampling rate as the total number of sensing samples per second in a WSN. To find the effectiveness of the compression algorithms on the sampling rate and the maximum sampling rate a WSN can have, we define Equation 23 which output the approximate maximum sample rate of the WSN, where $T_{ap} = 30.31\%$ is the maximum experimental normalized throughput of IEEE 802.15.4 radio in application layer[28], $V_{ch} = 250kbps$ is the channel speed of the radio [29], S_{data} is the uncompressed size of an original sensing sample and CR is the compression ratio of respective compression algorithm.

$$sampleRate_{max} \approx \frac{CR \times T_{ap} \times V_{ch}}{S_{data}} \tag{8}$$

4.2. COMPRESSION PERFORMANCE COMPARISON

This experiment evaluates the average compression ratio in compressing 5000 data items from each of the four different datasets listed before. The leaf nodes will do the compression locally. Then the compressed packets are concatenated at the intermediate nodes using Algorithm 4. To generate the compression rules, a sequence of previous sensing data are studied. When using the Z-compression algorithm, we set the learning

period to be 100 continuously sensing samples. The evaluation results are shown in Figure 8. The compression ratio in Y axis is calculated using Equation 22. Note that greater the compression ratio means better compression performance.



(a) Compression ratio of Intel lab data (b) Compression ratio of ZebraNet data

(c) Compresseion ratio of Accelerometer data (d) Compresseion ratio of Vehicle tracking data

Figure 8. Compression ratio of real-time datasets

4.3. ENERGY USAGE COMPARISON

This experiment is done in the TOSSIM simulator using PowerTOSSIM-Z [30]. We inserted data at the leaf nodes using the python script again. For some datasets such as Intel Lab and vehicle trace dataset, we inserted about 20000 samples each time. After compression and concatenating, the number of packets sending to the sink is much smaller than the original 20000 sample packets. The result is shown in Figure 9. As the compression ratio of Z-compression is better than the other compression algorithms for all the four dataset, the Z-compression reduces more packets than all others and thus, saving more energy and also reduces the bandwidth usage in the network. Note that as we are comparing different

datasets and each dataset has a different number of samples, in the energy comparison results, we use the normalized energy instead the real energy cost to show the effectiveness of each compression algorithm. The normalized energy is the ratio of the energy consumed by compressing or concatenating data and transmitting the fused packets over the energy consumed by only transmitting the fused packets. The experimental results are shown in Figure 10; the result shows that Z-compression provides best energy saving for the WSN. It is because, with the better compression ratio, the intermediate nodes can concatenate much more leaf node data into a larger packet that reduces the radio usage. Also, the fact that the intermediate nodes do not perform the compression and the overhead due to concatenating leaf node's payload is negligible and thus, we don't show the CPU energy consumption in the result.

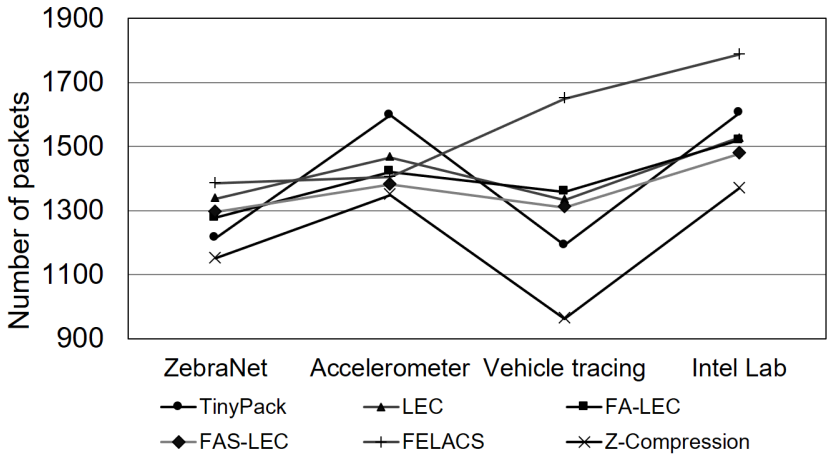


Figure 9. Total packets after compression and concatenating for 20000 sample data

4.4. APPROXIMATE MAXIMUM SAMPLING RATE

As we have discussed in Section 4.1, the maximum sampling rate is the rate that the maximum throughput of the sink node can support without dropping packets. It mainly depends on the compression ratio of the leaf nodes. The compression time will only

Table 3. Maximum approximate sampling rate using different compression algorithms with data concatenating

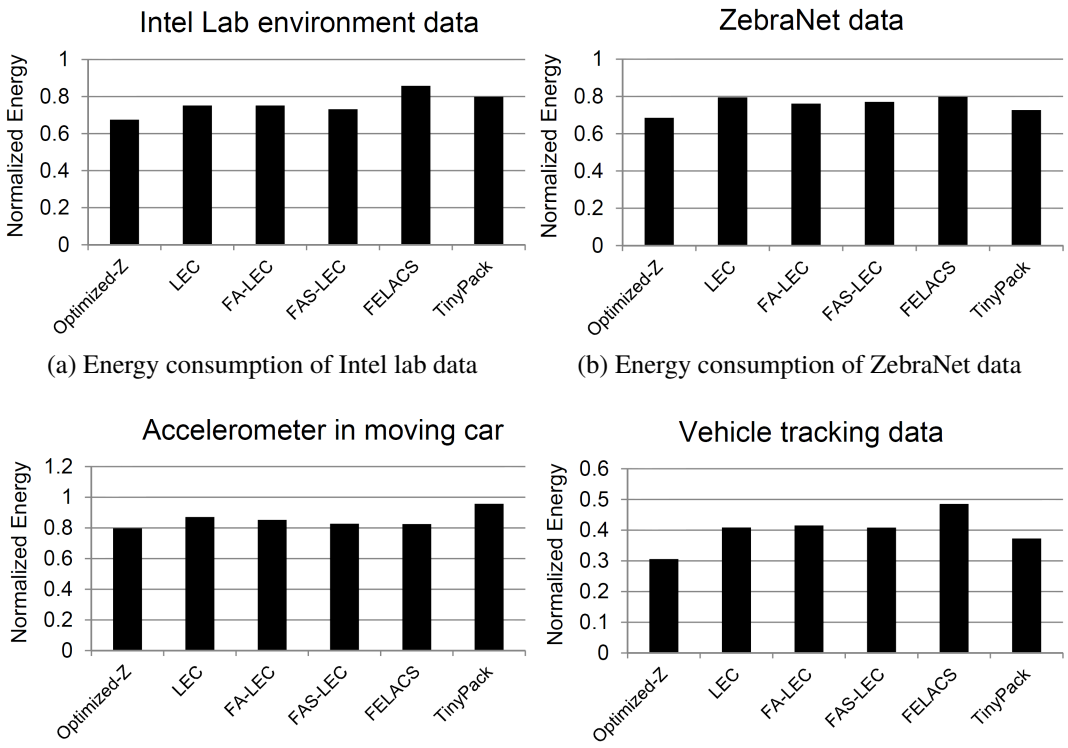
compression algorithms	Intel Lab environment data	Accelerator in moving car	ZebraNet data	Vehicle trace data (V to V)
Z-compression	1376	1409	1656	1947
LEC	1244	1297	1431	1430
FA-LEC	1245	1328	1500	1403
FAS-LEC	1282	1371	1479	1430
FELACS	1053	1341	1389	1119
TinyPack	1165	1172	1578	1594
No concatenating	131	136	136	117

determine the minimum sample interval of the leaf nodes in the WSN. The maximum sampling rate will not be affected by the compression time as we can increase the number of leaf nodes. Also, the time complexity of these 6 compressing algorithms are all O(n). It is about 30-60 milliseconds for compressing each sample including the sampling time.

Table 3 shows the approximate maximum sampling rate on compressing and concatenating different data sets versus direct forwarding data without compressing and data concatenating. The Z-compression has the best sampling performance comparing to LEC, FA-LEC, FAS-LEC, FELACS, and TinyPack. The approach without data concatenating has the worst sampling performance.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed Z-compression schemes based on Z-order encoding for lossless compressing of multi-modal sensor data in WSNs. We have performed several ToSSIM and TinyOS based simulation experiments using four real-world sensor datasets and measured the compression ratio, energy and sampling rate as performance metrics under different settings. The result shows that Z-order based compression algorithm with pre-defined rules has the robust performance across these metrics on compressing multi-



(a) Energy consumption of Intel lab data

(b) Energy consumption of ZebraNet data

(c) Energy consumption of Accelerometer data

(d) Energy consumption of Vehicle tracking data

Figure 10. Energy consumption of real-time datasets

modal real-world sensor data sets when compared with other existing schemes. Our scheme compresses multi-modal attributes into one Z value while avoiding using prefixes which always produce extra bits. Since the Z-order compression reduces the packets size, it allows the intermediate nodes to transmit less number of packets and thus, save energy and being able to mitigate the packet drops when streaming rate is high when compared with others.

In future, we plan to implement Z-order based compression algorithm in a sensor cloud [1], a paradigm of computation for wireless sensor networks which consist of wireless sensors from different owners and provides sensing as a service. With the help of lossless Z-compression, we can handle the maximum sensing request rate from many different clients.

REFERENCES

[1] Sanjay Madria, Vimal Kumar, and Rashmi Dalvi. Sensor cloud: A cloud of virtual sensors. *Software, IEEE*, 31(2):70–77, 2014.

[2] John Burgess, John Zahorjan, Ratul Mahajan, et al. CRAWDAD dataset umass/diesel (v. 2008-09-14), September 2008.

[3] Tung-Wei Kuo, Kate Ching-Ju Lin, and Ming-Jer Tsai. On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: Np-completeness and approximation algorithms. *IEEE Transactions on Computers*, 65(10):3109–3121, 2016.

[4] Chen-Xu Liu, Yun Liu, Zhen-Jiang Zhang, and Zi-Yao Cheng. High energy-efficient and privacy-preserving secure data aggregation for wireless sensor networks. *International Journal of Communication Systems*, 26(3):380–394, 2013.

[5] Dylan McDonald, Stewart Sanchez, Sanjay Madria, and Fikret Ercal. A survey of methods for finding outliers in wireless sensor networks. *Journal of network and systems management*, 23(1):163–182, 2015.

[6] Nguyen Quoc Viet Hung, Hoyoung Jeung, and Karl Aberer. An evaluation of model-based approaches to sensor data compression. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2434–2447, 2013.

[7] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM SIGMOD Record*, 30(2):151–162, 2001.

[8] Chiranjeeb Buragohain, Nisheeth Shrivastava, and Subhash Suri. Space efficient streaming algorithms for the maximum error histogram. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 1026–1035. IEEE, 2007.

[9] Hazem Elmeleegy, Ahmed K Elmagarmid, Emmanuel Cecchet, Walid G Aref, and Willy Zwaenepoel. Online piece-wise linear approximation of numerical streams with precision guarantees. *Proceedings of the VLDB Endowment*, 2(1):145–156, 2009.

[10] Francesco Marcelloni and Massimo Vecchio. An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks. *The Computer Journal*, 52(8):969–987, 2009.

[11] Massimo Vecchio, Raffaele Giaffreda, and Francesco Marcelloni. Adaptive lossless entropy compressors for tiny iot devices. *IEEE Transactions on Wireless Communications*, 13(2):1088–1100, 2014.

[12] Tommy Szalapski and Sanjay Madria. On compressing data in wireless sensor networks for energy efficiency and real time delivery. *Distributed and Parallel Databases*, 31(2):151–182, 2013.

[13] Jonathan Gana Kolo, S Anandan Shanmugam, David Wee Gin Lim, and Li-Minn Ang. Fast and efficient lossless adaptive compression scheme for wireless sensor networks. *Computers & Electrical Engineering*, 41:275–287, 2015.

[14] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[15] David A Huffman et al. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[16] Jeffrey Scott Vitter. Design and analysis of dynamic huffman codes. *Journal of the ACM (JACM)*, 34(4):825–845, 1987.

[17] Azad Ali, Abdelmajid Khelil, Piotr Szczytowski, and Neeraj Suri. An adaptive and composite spatio-temporal data compression approach for wireless sensor networks. In *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pages 67–76. ACM, 2011.

[18] Christopher M Sadler and Margaret Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 265–278, 2006.

[19] Sorabh Gandhi, Suman Nath, Subhash Suri, and Jie Liu. Gamps: Compressing multi sensor data by grouping and amplitude scaling. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 771–784. ACM, 2009.

[20] Henry Ponti Medeiros, Marcos Costa Maciel, Richard Demo Souza, and Marcelo Eduardo Pellenz. Lightweight data compression in wireless sensor networks using huffman coding. *International Journal of Distributed Sensor Networks*, 2014.

[21] Guy M Morton. *A computer oriented geodetic data base and a new technique in file sequencing*. International Business Machines Company New York, 1966.

[22] Dimitrios Lymberopoulos, Nissanka B Priyantha, and Feng Zhao. Towards energy efficient design of multi-radio platforms for wireless sensor networks. In *Information Processing in Sensor Networks. IPSN'08. International Conference on*, 2008.

[23] Bhaskar Krishnamachari, Deborah Estrin, and Stephen Wicker. The impact of data aggregation in wireless sensor networks. In *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*, pages 575–578. IEEE, 2002.

[24] Yong Wang, Pei Zhang, Ting Liu, Chris Sadler, and Margaret Martonosi. CRAWDAD dataset princeton/zebranet (v. 2007-02-14). Downloaded from http://crawdad.org/princeton/zebranet/20070214, February 2007.

[25] S Madden. Intel berkeley research lab data, 2003.

[26] Mohit Jain, Ajeet Pal Singh, Soshant Bali, and Sanjit Kaul. CRAWDAD dataset jiit/accelerometer (v. 2012-11-03), November 2012.

[27] Richard M. Fujimoto, Randall Guensler, Michael P. Hunter, Hao Wu, Mahesh Palekar, Jaesup Lee, and Joonho Ko. CRAWDAD dataset gatech/vehicular (v. 2006-03-15). Downloaded from http://crawdad.org/gatech/vehicular/20060315, March 2006.

[28] Nelson I Dopico, Carlos Gil-Soriano, Iñigo Arrazola, and Santiago Zazo. Analysis of ieee 802.15. 4 throughput in beaconless mode on micaz under tinyos 2. In *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*, pages 1–5. IEEE, 2010.

[29] TelosB Datasheet. Crossbow Inc. Downloaded from http://www.memsic.com/userfiles/files/Datasheets/WSN, 2013.

[30] Enrico Perla, Art Ó Catháin, Ricardo Simon Carbajo, Meriel Huggard, and Ciarán Mc Goldrick. Powertossim z: realistic energy modelling for wireless sensor network environments. In *Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pages 35–42. ACM, 2008.

II. MULTI-MODEL Z-COMPRESSION FOR HIGH SPEED DATASTREAMING AND LOW POWER SENSOR NETWORKS

Xiaofei Cao, Sanjay Madria, Takahiro Hara

ABSTRACT

Wireless sensor networks have significant limitations in available bandwidth and energy. The limited bandwidth in sensor networks can cause higher message delivery latency in applications such as monitoring poisonous gas leak. In such applications, there are multi-modal sensors whose values such as temperature, gas concentration, location and CO2 level need to be transmitted together for faster detection and timely assessment of gas leak. In this paper, we propose novel Z-order based data compression schemes (Z-compression) to reduce energy and save bandwidth without increasing the message delivery latency. Instead of using the popular Huffman tree style based encoding, Z-compression uses Z-order encoding to map the multidimensional sensing data into one-dimensional binary stream transmitted using a single packet. Our experimental evaluations using real-world data sets show that Z-compression has a much better compression ratio, energy saving, streaming rate than known schemes like LEC (and adaptive LEC), FELACS and TinyPack for multi-modal sensor data.

Keywords: Sensor network, Data compression, Z-order

1. INTRODUCTION

Wireless sensor networks (WSNs) are being developed for a plethora of emerging applications in a wide range of disciplines. For example, there are near real-time sensor cloud applications [1] to perform multi-modal sensing tasks. Some military applications

like tracking hostile objects or monitoring intruders use multiple sensing units to provide precise location and speed by applying multi-sensor data fusion. The unmanned vehicles [2] need GPS and accelerometer to locate themselves, and to track distance and height of objects using the camera, laser range meter or radar. By fusing these multi-modal sensor data, they can also predict the moving trajectory of the nearby objects. Similarly, the environmental monitoring applications need temperature, wind direction, humidity, CO2 levels, etc. Many of these multi-modal sensor applications require data integrity (lossless data) as well as high-streaming rate, and therefore, cannot tolerate high latency due to limited bandwidth in WSNs. Since batteries are the typical power source for wireless sensors, the energy consumption is another primary constraint in the design of multi-modal WSNs. Many research efforts have shown that radio communication, including the data transmission and channel listening, is the predominant factor among all the energy consumption metrics of the WSNs.

The power model of Micaz in [3] shows that the channel listening even consumes more power than the transmission. The duty cycle approaches are the most straightforward way to reduce the radio communication. The fundamental idea of these duty cycle schemes is to put the sensors to sleep periodically. When the sensors fall asleep, there is no radio communication at all which minimizes the energy consumption. We can divide these duty cycle approaches into two categories; synchronous and asynchronous. The synchronized duty cycle MAC protocols include [4], [5], and [6]. and asynchronous approaches include [7], [8], [9]. Among them, the low-power listening (LPL) scheme in TinyOS has proven its ability to reduce the duty cycles of the wireless sensor nodes. It works well for the applications that are without heavy loads. However, in LPL mode, sensors will wake up for a full period for sending messages. Thus, there is a need for lossless data compression algorithms which could reduce the size of multi-modal sensing data and thereby, decreasing the radio communication.

Sensor data aggregation approaches like [10][11] save energy and bandwidth by reducing the number of packets to be transmitted. However, they cannot guarantee the data integrity because of the lossy process and the outliers. Also, when aggregating, the outlier detection is expensive and introduce delays [12]. The model-based compression algorithms [13] such as APCA[14], PWLH[15], and SF[16] also have good compression ratios. However, they can not work with applications requiring lossless data as they approximate the data with temporal and spatial locality. The compressive sensing approaches like [17] have the drawback that the data may lose integrity thus, are not suitable for multi-modal lossless data compression.

Existing works [18][19][20] propose the lossless compression algorithms for sensor data using Huffman tree style coding that exploit the temporal locality of the data in WSNs. Instead of storing and transmitting the complete data, [18][19][20][21] use the difference in values, called delta value, between the two adjacent timestamp readings and usually, it needs fewer bits to represent a delta value. Based on the Shannon entropy theory [22] and the Huffman coding[23], the most frequent values are assigned a shorter code than the less frequent values. Thus, if a dataset is drawn from a smaller set of values, the entropy will be smaller. However, the standard Huffman and adaptive Huffman [24] coding have larger overhead on RAM in storing the Huffman trees generated dynamically based on the frequency of the data. Also, to decode the compressed data, every node in the wireless sensor network needs to have a copy of the tree. However, as we know that nodes in WSNs have limited bandwidth and energy, synchronizing the Huffman tree among nodes is impractical. LEC and Adaptive-LEC [18][19] successfully adapted Huffman coding for their static initial code library which is a predefined Huffman tree. That way, WSNs do not require transmitting the entire Huffman tree. Similarly, the TinyPack [20] modified its initial code library based on LEC's and proposed algorithms adapting library to different types of sensor applications. The static initial code cannot always give the best performance because

the distribution of the dataset can deviate from the optimized code tree. Further more, these schemes are designed to work for a single attribute value as the multi-dimensional sensing data can have different distributions for each attribute.

To address the problems discussed above and improve the applicability and compression ratios of existing works, we propose a Z-order [25] encoding based data compression scheme. The Z-order encoding called Z-compression can compress multi-modal sensing data at each leaf node as well as at the intermediate nodes efficiently in near real-time. The Z-compression algorithm can encode multi-modal sensor data like precipitation, water level, and wind speed (needed to detect a flood risk in a region) into a binary stream. Using our Z-compression algorithm in a WSN with a hierarchical topology [26], the nodes with limited bandwidth can tolerate higher-stream data rates coming from upstream nodes by concatenating compressed sensor data into the reduced number of packets which may be as large as permissible by the network protocol. The proposed Z-compression algorithm also uses temporal and spatial data locality and delta encoding for better performance. Instead of using Huffman style coding which requires extra bits for each delta values, we use Z-order encoding to compress the delta values of all attributes of the input data into a binary stream. When decoding we use the predefined decoding rules to decode the Z-values and extract all the values of attributes.

We conducted extensive experiments using skewed and unskewed real datasets. We found that Z-order encoding based compression performs better than Huffman tree based source coding approaches. We further optimized the original Z-order encoding, where, for skewed datasets, we proposed the initial code library to improve the compression performance further. Our experiments show that it has much better compression ratios for the multi-dimensional datasets than the previous Huffman coding based compression approaches like LEC [18], TinyPack [20], Adaptive-LEC [19] and FELACS [21]. The packet

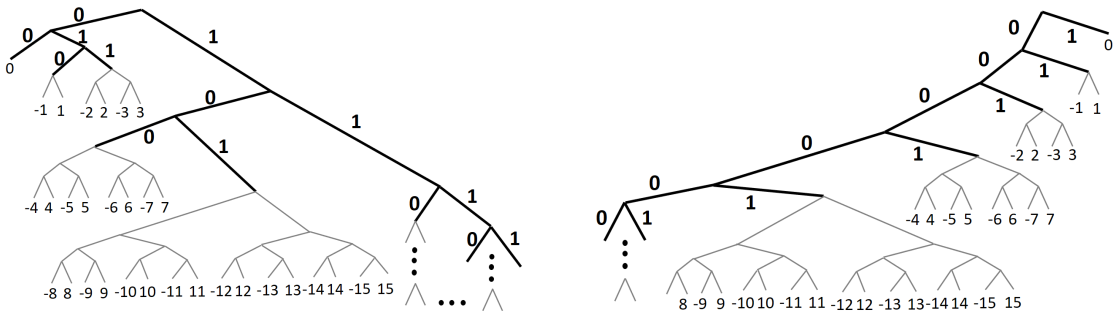
compression evaluation is done in a wireless sensor network using TelosB motes, which use the IEEE 802.15.4 radio and 8 MHz TI MSP430 microcontroller with 10KB RAM. The energy consumption rate evaluation is done in TOSSIM [27] using powerTossim-Z [28].

In this paper, we compare the performance of our work with LEC, Adaptive-LEC, FELACS and TinyPack (since they outperform other algorithms like ASTC [29], S-LZW [30] and GAMPS[31]). [32] proposed their coding dictionary based on a very specific dataset, thus, not considered in our experiments.

In summary, this paper makes the following contributions: Design of an efficient multi-modal sensor data Z-compression scheme that combines the Z-order encoding with delta compression. To our knowledge, this is the first attempt to propose a multi-modal lossless data compression algorithm for WSNs. Create a probability model of the sensing data which has the temporal and spatial locality such as the environmental data, location data and motion data. Design a data concatenation scheme which can concatenate leaf nodes data efficiently for compression. Also, improve Z-order encoding by integrating a small dictionary and an odd bit Z-order encoding for further performance improvement in WSNs. Integrate the proposed Z-compression scheme with low-power listening and high-streaming applications of WSNs and prove its effectiveness using the simulation experiments. Perform extensive simulations using TinyOS and TOSSIM and compare the performance with the recent and popular sensor data compression algorithms. The results show that our schemes outperform these other schemes in terms of compression ratio, handling high-streaming data and energy usage as the metrics.

2. BACKGROUND AND RELATED WORK

S-LZW [30] an extension of LZW [33] compresses data by encoding and representing a common sub-string with fewer bits used for the encoded value. The encoded common sub-strings are stored in the dictionary which usually is too large to store at sensor motes. LZW and S-LZW usually need more data to create new words and update their library before



(a) Huffman tree of LEC code (b) Huffman tree of TinyPack's code

Figure 1. Huffman tree of the compression code of LEC and TinyPack

compressing, which causes longer delays.Additionally, the average waiting time increases so LZW and S-LZW cannot be used for real-time wireless sensor applications as shown in our earlier work also [20].

LEC [18] and TinyPack [20] are both Huffman coding based delta compression algorithm. The Huffman coding would represent higher frequency symbols with less number of bits. However, in sensor networks, delta values can range from 0 to more than 2^{16}. Thus, the memory limitation of nodes makes creating such large Huffman coding dictionary impossible. In their improved approaches to save memory and to make them computationally efficient, instead of adapting Huffman coding to each symbol, LEC and TinyPack create the Huffman coding for the length of the delta value in a fixed pattern that matches the frequency of the data length group which they assume is decreasing when the length of delta value increases. Thus, in both LEC and TinyPack, the length of Huffman coded prefix increases with the increasing delta values length. The Huffman tree of LEC is shown in Figure 1a and the TinyPack code is shown in Figure 1b. These two coding focused on the different distribution of the delta values of the dataset. For example, if a dataset contains many data of only one-bit length, according to Figure 2, TinyPack will have better compression ratio than LEC in terms of normalized compression overhead (ratio of the extra bits required to code the delta value to the data bits) whereas, for other delta values larger than 3, LEC will have better performance than TinyPack.

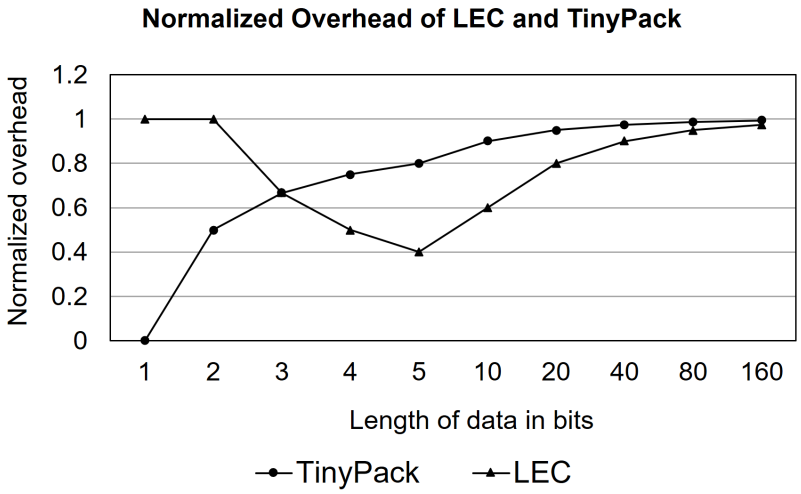


Figure 2. Normalized compression overhead of LEC and TinyPack

However, as we have discussed before, the distribution of the delta values can be varied in different applications. Even in the same application at different time periods or at different sensor nodes, the distribution may vary. Thus, in these situations, LEC and TinyPack do not show good performances as they have fixed code. To solve this problem, Adaptive-LEC [19] and TinyPack-DP(TP-DP)[20] can adapt the prefix code when the length of the distribution of the frequency of the delta value changes. Adaptive-LEC will adapt its prefix code for every new data while the TP-DP only changes its prefix code at the beginning of each new frame. They define the length of delta value with the most frequency as the frequency center of data and define the length of delta value with the minimum prefix code as the code center. When the current frequency center of data is drifted from the previous code center, the prefix code will shift to meet the current frequency center of the data. For some datasets with the two frequency centers, they proposed another initial code with the two code centers and two adaptive segments where the prefix codes will be adapted independently based on the segments. However, the drift correction, always behind the data drift, happens with a very high delay. Thus, it reduces the performance of the compression algorithm. Also, different sensor nodes may have different code centers which requires much overhead to handle the asynchronous WSNs.

[34] algorithm takes advantage of spatial locality between the two or more nodes, and performs collaborative data compression. However, in WSNs, sensors may be sensing multidimensional data independently and waiting for other sensors to dispatch their different attribute values will increase latency in processing and delays in transmission in an intermittent environment. Thus, compressing the multi-modal sensing data locally at each node becomes essential for real-time data transmission. Z-compression does not need to correlate data with other nodes. Therefore, there is no need to wait for more data to acquire before compressing.

FELACS [21] gives every compressing data a fixed b bits. Every value smaller than 2^b will be filled with '0's at the front to reach b bits, added '1' bit at the front then directly appended to the output stream. Every value larger than or equal to 2^b will be cut into two sections. The higher bits section is encoded using unary coding and is appended to the output stream. The lower bits section has the length b directly appended to the output stream. The fixed bits b is generated by calculating the average number of the input data.

FELACS adapts data packets by packets rather than sample by sample. It works for a single sensing attribute only. Also, it needs to wait for more data to achieve better compression performance as they have fixed indicator bits.

3. PROPOSED Z-COMPRESSION APPROACH

In this section, we describe the efficient Z-order encoding based multi-modal data compression scheme, Z-Compression. It is a lossless compression algorithm that exploits temporal and spatial locality that only consider the value difference between two adjacent data points. Also, it compresses every single sensing data sample without waiting for more to arrive as only one previous data point is considered. The wireless sensor nodes use the delta value of each attribute as the input of the compression algorithm. The delta value is calculated using Equation 1 where V_c is the current delta value, V_p is the previous delta

value and P is the resolution of the sensors.

$$d_{\Delta} = \frac{V_c - V_p}{P} \tag{1}$$

3.1. NAIVE MULTI-DIMENSIONAL Z-COMPRESSION FOR SENSOR VALUES

Naive Z-compression uses Z-order encode [25] and all-is-well bit like in [20]. As shown in Figure 3, the Z-order encode interleaves input data bit by bit and output a new binary number with a length that double of the largest input.

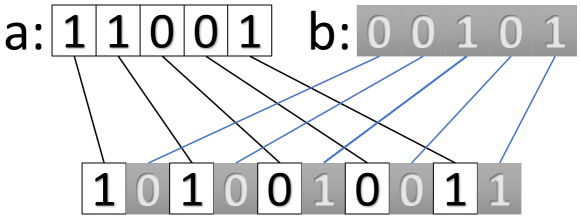


Figure 3. Procedure of Z order encoding

$$V = \begin{cases} 2 \times V_{signed}, & \text{if } V_{signed} > 0 \\ 1, & \text{if } V_{signed} = 0 \\ 1 - 2 \times V_{signed}, & \text{if } V_{signed} < 0 \end{cases} \tag{2}$$

We choose the byte array as the data structure to store the encoded data. Also, we use unsigned integer V to represent both positive, negative and zero values $V_{signed}$by using the Equation 2. Next, we add '1' at the front of the output to protect the possible '0'. For example, consider three dimensional array, '10', '110' and '1'. The largest attribute is '110'

which has 3 bits. So first, we need to add zeros to the other two attributes. Then applying Z-encoding on '010', '110', and '001' to get the Z-value, '010110001'. Then, we add '1' which gives the final result of '1010110001'.

In wireless sensors, as the computational and memory resources are limited, it is better to use bitwise operator rather than converting the encoding data into string format when doing Z-order encoding. We use bitwise 'shift', 'or'(|), 'and'(&) for interleaving a certain bit from the input data and append to the output. Note that the notation '<<' and '>>' means shift left and shift right for certain bits. We first need to find out the length of the largest input data as B_l. Then the length of the output array, L_Z, can be calculated using the Equation 3. In the same way, the output length of TinyPack and LEC on compressing the two sensing attributes can be calculated using Equation (4) and (5) where B_i represent the number of bits of the ith attribute which needs to be compressed.

$$L_Z = N \times B_l + 1 \tag{3}$$

$$L_{TP} = \sum_{i=1}^{N} (2B_i - 1) + 1 \tag{4}$$

$$L_{LEC} = \sum_{i=1}^{N} (2B_i - T) + 1, T = \begin{cases} 0, & \text{if } 0 < B_i < 3 \\ 1, & \text{if } B_i = 3 \\ 2, & \text{if } B_i = 4 \\ 3, & \text{if } B_i = 5 \\ 4 & \text{if } B_i > 5 \end{cases} \tag{5}$$

As stated, we also integrate an all-is-well bit [20] with Z-order encoding. It sets the compressed data to be zero if all the delta values of the input attributes are zero. In Z-compression, we set the result to be an all-is-well bit when nothing changes. It is done by directly checking all the input values. If all the inputs equal to '1' then output '1' where '1' equals to zero according to the Equation 2.

Different from the Huffman-tree based compression algorithms that need to know the probability distribution of the input data to achieve the best performance, Z-compression only exploits the relationship between the length of delta values of the attributes. We then compare the normalized overhead (in terms of extra bits over total bits) of Naive Z-compression, LEC, and TinyPack on compressing the uniformly distributed multi-modal random data by using the Equations (3), (4) and (5). The result is shown in Figure 4.

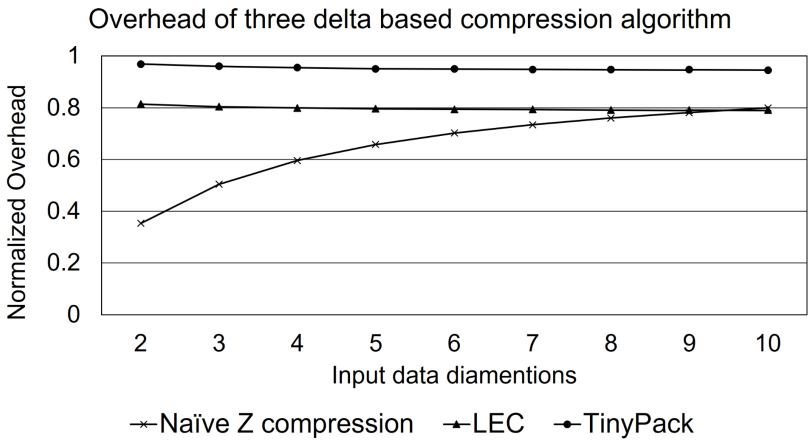


Figure 4. Normalized compression overhead compressing multidimensional Data

The result shows that Naive Z-compression performs well when the number of attributes is small. However, with the increasing number of attributes, the overhead of Z-encode increases. When the number of attributes is larger than eight, the overhead of Z-encode is greater than that of LEC. To address this drawback, we next propose an Optimized two-dimensional Z-compression scheme which guarantees to have a better compression ratio than TinyPack and LEC when compressing two-dimensional data. Thus, for multi-dimensional data, we can split it into several groups with two or more attributes to minimize the overall overhead due to extra bits.

3.2. OPTIMIZED TWO-DIMENSIONAL Z-COMPRESSION ALGORITHM

Before improving Naive Z-compression, we need to study two important properties of two-dimensional Z-order encoding. First, the number of bits in the output of Two-dimensional Z-order encoding is always even. Second, there must be at least one of '1' in the first two bits of the output of the two-dimensional Z-order encoding in binary format. Here, we do not consider the extra '1' bit added before the compression result. The first property indicates that We can use the value with odd length to represent the 'skewed' data. Here, we define a two-dimensional dataset as skewed when the number of bits of the larger delta value is more than two times the number of bits of the smaller as in Equation 7. When the data is skewed, in order to make the length of Z-value odd, we divide the larger value V_L with length B_l into two values V_{L1} and V_{L2} of length B_{l1} and B_{l2} using the Equations 6a, 6b and 6c.

$$Mask_{l2} = (1 << B_{l2}) - 1, \tag{6a}$$

$$V_{L1} = V_L >> B_{l2}, \tag{6b}$$

$$V_{L2} = V_L \& Mask_{l2}; \tag{6c}$$

We can apply Z-order encoding on V_{L1} and V_S to get the Z-value Z_{half}. Then by appending V_{L2} on Z_{half}, shown in Equation 8, we get the Z-value for the skewed data. The notation $\otimes$ means applying Z-order encoding and $\oplus$ means concatenating the two binary strings together. In the above procedure, we have to ensure that the Z-value has odd length L_{odd} and each different two-dimensional data maps to a unique Z-value.

Algorithm 1 shows the procedure of the Optimized two-dimensional Z-compression. It reduces the extra bits adding to the smaller delta value when the delta value pair is skewed. However, when the delta value pair is not skewed, Naive Z-encode is applied.

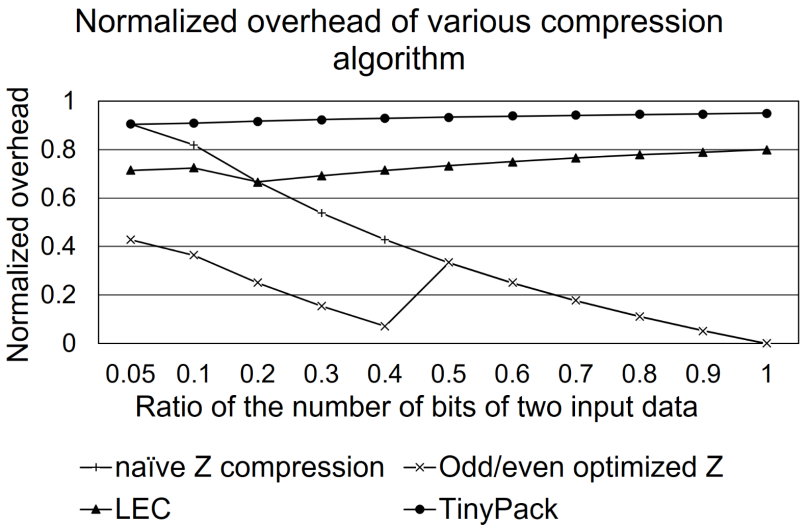


Figure 5. Normalized overhead in compressing two attributes where the largest attribute's data is 20 bits long

For example, when encoding two attributes '101' and '11110000', we add two '0' bits to the first attribute '101' which makes it '00101' and set the end pointer of the second attribute to be 3 which divides it into two values, '11110' and '000'. Then we apply Naive Z-compression on '00101' and '11110' which produces '0101110110'. At the end, we append the remaining '000' to it and add '1' as the leftmost bit which gives us the encoded Z-value of '10101110110000' which is 14 bits long.

$$B_s < floor(B_l/2) \tag{7}$$

$$Z = Z_{half} \oplus V_{L2} = (V_{L1} \otimes V_S) \oplus V_{L2} \tag{8}$$

When decoding, we use the second property of two-dimensional Z-order encoding to help us figure out which value within the two decoded data is larger. The value with less '0's at the front is larger. Then we append the remaining data to it to get the final result. For example, when decoding '10101110110000', we first count the length of the Z-value after the first '1' which give us 13. This odd value 13 indicates that the delta value pair is skewed. Then using 13%6 = 1, we know that it belongs to the first case in Algorithm

1 that B_L%4==0. And, using $n = 13/2 = 2$, we can divide the Z-value into two parts at $2 \times (2n + 1) = 10$ bits from left. By interleaving the bits from the first part, we get '00101' and '11110'. At the end, we append the remaining '000' to the second value as it has less '0' bits at the front than the first one so the decoder will output two binary values '00101' and '11110000'.

Algorithm 1: Optimized Two-dimensional Z-compression algorithm using odd/even bits optimization

input : Delta value V_L and V_S where $V_L > V_S$
output Z value
:

initialization $B_L \leftarrow Length(V_L)$, $B_S \leftarrow Length(V_S)$
if $B_S \geq floor(B_L/2)$ **then**
    $Z = V_L \otimes V_S$;
else
    if B_L%4 == 0 **then**
        $n \leftarrow B_L/4$;
        Divide V_L into V_{L1} and V_{L2} where $V_L = V_{L1} \oplus V_{L2}$;
        $Length(V_{L1}) = 2n + 1$ and $Length(V_{L2}) = 2n - 1$;
        $Z = (V_{L1} \otimes V_S) \oplus V_{L2}$;
    else if B_L%4 == 2 **then**
        $n \leftarrow B_L/4$;
        Divide V_L into V_{L1} and V_{L2} where $V_L = V_{L1} \oplus V_{L2}$;
        $Length(V_{L1}) = 2n + 1$ and $Length(V_{L2}) = 2n + 1$;
        $Z = (V_{L1} \otimes V_S) \oplus V_{L2}$;
    else if B_L%4 == 3 **then**
        $n \leftarrow B_L/4$;
        Divide V_L into V_{L1} and V_{L2} where $V_L = V_{L1} \oplus V_{L2}$;
        $Length(V_{L1}) = 2n + 2$ and $Length(V_{L2}) = 2n + 1$;
        $Z = (V_{L1} \otimes V_S) \oplus V_{L2}$;
    else
        $n \leftarrow B_L/4$;
        $V_L =' 0' \oplus V_L$;
        Divide V_L into V_{L1} and V_{L2} where $Length(V_{L1}) = 2n + 1$ and $Length(V_{L2}) = 2n + 1$;
        $Z = (V_{L1} \otimes V_S) \oplus V_{L2}$;
return $Z =' 1' \oplus Z$;

In Figure 5, we compare the normalized overhead of extra bits of four different compression algorithms LEC, TinyPack, Naive Z-compression and Optimized Z-compression on compressing two-attributes data where the larger attribute is 20-bits long. The X-axis is the ratio of the number of bits of the input delta value pairs. The Y-axis is the normalized overhead which is the extra bits over the total bits that the input delta value pair has. We can see that Optimized Z-compression dwindle the extra bits significantly. The maximum normalized overhead is 0.43 which is half of the normalized overhead of TinyPack. In Optimized Z-compression using odd/even bit optimization, we use two as the critical ratio of two attributes' length to trigger the Optimized Z-compression as it is easier to implement.

Algorithm 2: Optimized Z-compression algorithm with input length ratio indicator

input : Delta value V_L and V_S where $V_L > V_S$
output Z value
:
initialization $B_L \leftarrow Length(V_L)$, $B_S \leftarrow Length(V_S)$
if $CalculateOverhead(B_L, B_S)$ *prefer Naive* **then**
    $Z =' 1' \oplus V_L \otimes V_S$;
else
    $Z =' 0'$;
    $Z = Z \oplus FindBestRatio(B_L, B_S)$; $Z = Z \oplus FindZValue(V_L, V_S, ratio)$;
return $Z =' 1' \oplus Z$;

Another optimization scheme that can apply Optimized Z-compression on two input attributes with any ratio of lengths is compared. We use a fixed small bit as a ratio indicator to indicate the actual ratio of two attributes' lengths. However, extra control bits can cause extra overhead. For example, when using two bits indicator, four different ratio which are $\frac{3}{2}$, $\frac{3}{1}$,$\frac{2}{3}$, and $\frac{1}{3}$, can be indicated using '00', '01', '10', '11'. We can then interleave binary values in a fixed ratio. For example, when the ratio is set to be $\frac{3}{2}$, we should interleave 3 bits at a time for the first delta value and 2 bits at a time for the second delta value.

We changed the ratio indicator bits from 2 to 4 and tested it on a uniformly distributed random data set. Figure 6 shows that with the increasing length of input data, the average overhead of different optimized Z-compression also increases. However, the odd/even checking based optimization performs better than others when the input's maximum length is fewer than 45 bits which is larger than the size of most wireless sensor's data sensing output. Thus, when compressing real-time sensing data packets, we suggest only to use Algorithm 1.

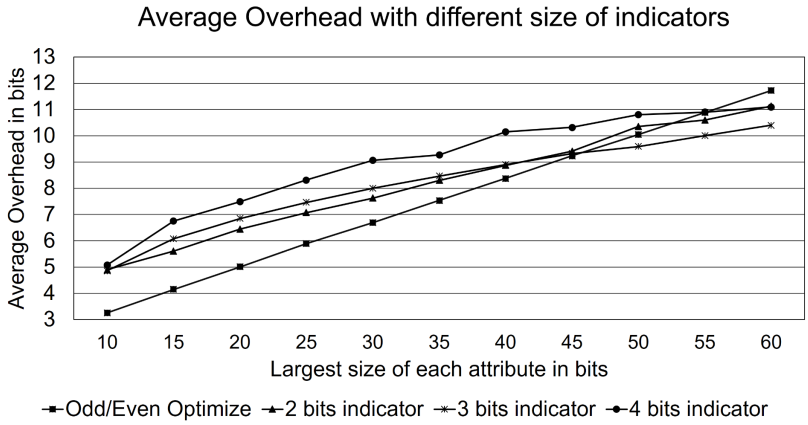


Figure 6. Average overhead in compressing data with two attributes

3.3. SMALL CODE LIBRARY ADD-ON

We can further improve Optimized two-dimensional Z-compression scheme by integrating a small code library. This library will enhance the compression performance without affecting the correctness. The code library is shown in Table II. For entries in the small library, one of the input delta value pairs should have '0' or '1'. We assign the output a smaller value which never appears in the result of Naive or Optimized Z-compression. For example, when we compress two delta values where both the values are 0, the output of encoding value will be 1-bit '1' instead of 2-bits '11' giving 50% improvement. Also, this small code library will not affect the correctness of the encoding and decoding procedures

given in Algorithm 1 because it will not generate Z-values that the small library has. When decoding, we can check the small library before checking the length of Z-value to get the correct decoding result.

Table 1. Initial small code library

Value 1	Value 2	Z value
0	0	1
0	1	11
1	0	10
0	-1	111
-1	0	110
0	V_2 >31 || V_2 <-31	10000⊕V
V_1 >31 || V_1 <-31	0	100000⊕V

3.4. OPTIMIZED N-DIMENSIONAL Z-COMPRESSION ALGORITHM

To improve Naive Z-compression for more than two-dimensional data, we proposed the optimized N-dimensional Z-compression that exploit the data correlation between close attributes. As the number of the sensing attributes are fixed based on the sensing hardware, we suppose both the encoder and decoder know how many attributes they are going to compress and decode.

The optimized N-dimensional Z-compression algorithm combines the procedures discussed in 3.A, 3.B, and 3.C using a predefined rule which is generated in Algorithm 3. When compressing multidimensional sensing data, we can either use Naive Z-encoding based compression on all the attributes or use Optimized Z-compression on the pairs of attributes and then merge the result. Testing all the combinations of the input attributes with the above two encoding methods is an NP-complete problem. Thus, we propose an approximate algorithm using two pointers and a local greedy approach to find the approximate combination result given in Algorithm 3. We use a two-dimensional array as

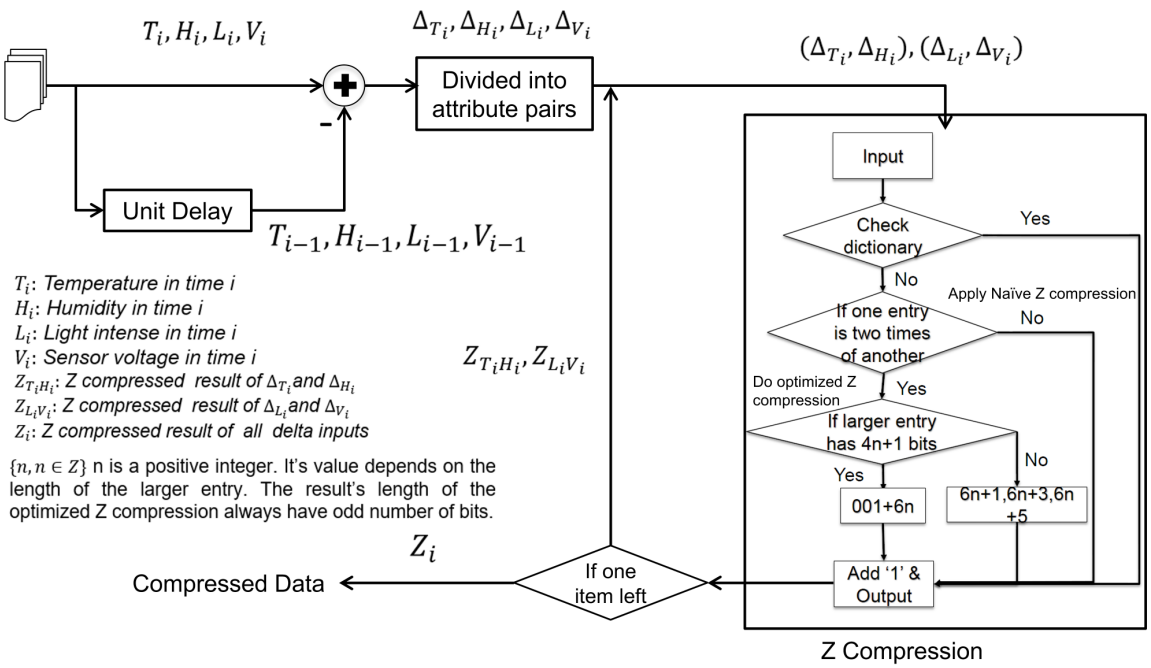


Figure 7. System model Of optimized Z Compression

GroupMember to represent the attribute IDs and their length. We use another array as Group to store the list of GroupMembers. The input of the algorithm is a list of GroupMembers which represent all the attributes. The output of the algorithm is a list of Groups which instructs the encoding and decoding procedure.

The encoder will use the Group information to help them encoding. If a Group contains more than three entries, the encoder will use the Z-order encoding to compress the attributes represented by the group. If a Group contains only two attributes, the encoder will use Algorithm 1 and Table 1 to compress them. At last, the sensor will further encode all the encoded values and output the result. For example, if there is a Group [{1,7},{2,5},{4,9}], the wireless sensor node will do a Z-order encoding on the attributes with field IDs 1, 2 and 4. If there is another Group [{3,3},{5,10}], the sensor will do the optimized two-dimensional Z-compression on the attributes with field ID 3 and 5. Then the node will

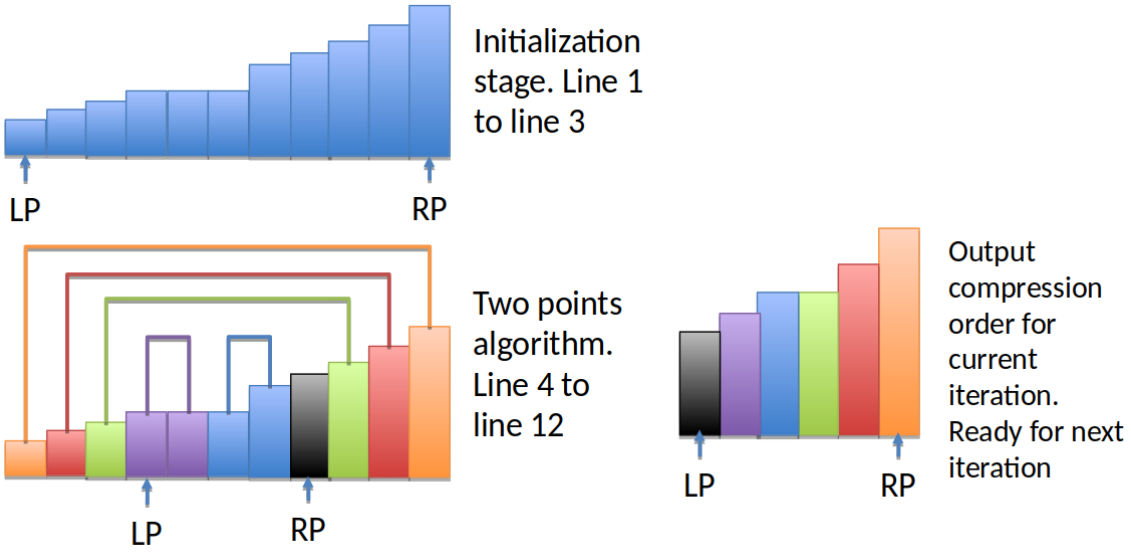


Figure 8. An example of group algorithm

compress the two encoded values using optimized Z-compression as there are only two groups. After transmitting the compressed value to the decoder, the decoder will decode the Z-value reversely based on the number of groups and the field IDs in each group.

To improve Naive Z-compression for more than two-dimensional data, we proposed the optimized N-dimensional Z-compression that exploit the data correlation between close attributes. As the number of the sensing attributes are fixed based on the sensing hardware, we suppose both the encoder and decoder know how many attributes they are going to compress and decode.

The optimized N-dimensional Z-compression algorithm combines the procedures discussed in 3.A, 3.B, and 3.C using a predefined rule which is generated in Algorithm 3. When compressing multidimensional sensing data, we can either use Naive Z-encoding based compression on all the attributes or use Optimized Z-compression on the pairs of attributes and then merge the result. Testing all the combinations of the input attributes with the above two encoding methods is an NP-complete problem. Thus, we propose an approximate algorithm using two pointers and a local greedy approach to find the

approximate combination result given in Algorithm 3. We use a two-dimensional array as GroupMember to represent the attribute IDs and their length. We use another array as Group to store the list of GroupMembers. The input of the algorithm is a list of GroupMembers which represent all the attributes. The output of the algorithm is a list of Groups which instructs the encoding and decoding procedure.

The encoder will use the Group information to help them encoding. If a Group contains more than three entries, the encoder will use the Z-order encoding to compress the attributes represented by the group. If a Group contains only two attributes, the encoder will use Algorithm 1 and Table 1 to compress them. At last, the sensor will further encode all the encoded values and output the result. For example, if there is a Group [{1,7},{2,5},{4,9}], the wireless sensor node will do a Z-order encoding on the attributes with field IDs 1, 2 and 4. If there is another Group [{3,3},{5,10}], the sensor will do the optimized two-dimensional Z-compression on the attributes with field ID 3 and 5. Then the node will compress the two encoded values using optimized Z-compression as there are only two groups. After transmitting the compressed value to the decoder, the decoder will decode the Z-value reversely based on the number of groups and the field IDs in each group.

For example, for compressing the Intel Berkeley Lab data, which includes the following four attributes: temperature, humidity, Light tense, and voltage of sensors, the system model of optimized Z-compression is shown in Figure 7.

First, calculate the delta values of these four attributes using Equation 1. Second, we do learning based on the first 100 packets. Find the compression rules using Algorithm 3. Third, group the attributes based on the compression rules. In this example, the best compression strategy is to group the temperature data with humidity data and group the light intensity data with the sensor voltage data, then group the compressed data of these two groups. Next, compress data in each group using the optimized Z-compression Algorithm 1 until there are no more groups and then output the compressed data. Figure 8 shows an example of using grouping algorithm to solve the optimized Z-compression.

Algorithm 3: Rule Generation Algorithm

```
input  : List of GroupMembers: GML
output  List of Groups: GL
:
Sort(GML); //sorting based on the length of attributes
LP← 0; //left pointer starting from the left end
RP ← GML.getSize(); //right pointer initialization
while (length(GML[LP])<length(GML[RP-1])/2&&LP<RP) do
    LP++,RP- -; //find pairs meeting Equation 2
//add groups of the pairs to the output list
while RP<GML.getSize() do
    GL.add(new Group{GML[RP++],GML[GML.getSize()-RP]}
bufGroup=new  Group{}//initial a new group
//add rest GroupMembers to the Group
for (i=LP;i<GML.getSize()-LP;i++) do
    bufGroup.add(GML[i]);
GL.add(bufGroup);
return GL;
```

Algorithm 4: Lossless data concatenating algorithm

```
input  : Q, N, b_max, b_min, sum, L
output  payload: out[]
:
Index=new byte[N], ind=1, prev=L[0][1];
Sort(L); //sorting is based on L[][0]
for (i=0;i<N;i++) do
    Index[L[i][1]]=ind;
    ind+=prev;
    prev=L[i][1];
prev=L[0][1];
for (i=0;i<N;i++) do
    buf=Q.poll();
    for (j=prev-L[i][1];j<L[i][1];j++) do
        out[Index[i] + j]=buf[j]
    prev=L[i][1];
out[0]=N;
return out;
```

3.5. LOSSLESS DATA CONCATENATING ALGORITHM

The main reason to concatenate the locally compressed packets is to save the energy and bandwidth usage further. In our experimental results, also shown in experimental analysis of radio performance in [35] and [36], we found that reducing the payload size of the packets at the leaf node will not give us much energy saving. Also, a leaf node is not the bottleneck in WSNs as there is not much traffic via them. However, we found by the experiment that the intermediate nodes are the bottleneck as they not only need to sense data but also need to route packets from the lower level nodes to the higher level nodes in the tree. The energy consumption rate and bandwidth occupation by the intermediate nodes are much more critical than the leaf nodes. Thus, to save energy and prolong the lifetime of WSNs, we only need to consider the energy consumption of the node with the most radio connections. To do that, we need to concatenate the upstream data to decrease the number of packets each intermediate node will transmit.

Next, we discuss how to concatenate the locally data packets. The compressed data are byte arrays of variable length. It has the node ID and timestamp as the primary key. It is not practical to decode and re-encode all the data at the intermediate nodes as it will increase the RAM and CPU load and thus, will cause delays. Here, we propose a data concatenating algorithm which will concatenate byte array input data efficiently. The idea is that in a packet transmitted by the intermediate node, the compressed data with a larger number of bytes is always in front of the compressed data with a smaller number of bytes. However, the compressed data with a smaller number of bytes will be filled with zero bytes to make them have the same number of bytes as their neighbor in front of them.

At the end, we set the first byte of the output packet as the number of bytes of the largest data. When decoding, we first read the length of the largest packet b_{max} at the first byte. Then we read the following $2b_{max}$ bytes to extract the first and the second data. If there are empty bytes in front of the second data, we need to update the b_{max} by subtracting the number of empty byte from b_{max}. Next, we use the updated b_{max} to extract the third data

Table 2. Fields of Experimental Dataset

data set name	number of fields	fields label
Intel Berkeley Lab data	6	epoch, node ID, Temperature, Humidity, Light, Voltage
Accelerator in moving car	5	epoch, node ID, X, Y, Z
ZebraNet data	5	epoch, node ID, Longitude, Latitude, Voltage
Vehicle trace data (V to V)	10	epoch, node ID, (Longitude, Latitude, Altitude, speed)× 2

and update the b_{max} again. Repeating this process until the last byte of the packet, we can extract all the data samples in the packet. Figure 9 shows how we concatenate small size packets into large size packets by sorting the size of the packets in the descending order.

$$PacketSizeLimit < (sum + b_{max} - b_{min}) \tag{9}$$

Note that we use a queue Q to store the upstream packets' payload, a two-dimensional array L to store the size of each payload and their index in Q, an index array $Index$ to help append the data in the queue to the output byte array. We also need to track the largest packet length b_{max}, the smallest packet length b_{min} and the total length of all the packets $sum = \sum_{i=1}^{n} b_i$ in the queue. Once the criteria in Equation 9 is satisfied, we use Algorithm 4 to concatenate the elements in the payload queue and create a larger size packet. The total number of data needs to concatenate is 'N' which exclude the last item in the queue. Then the intermediate node will transmit the large size packet to its downstream node.

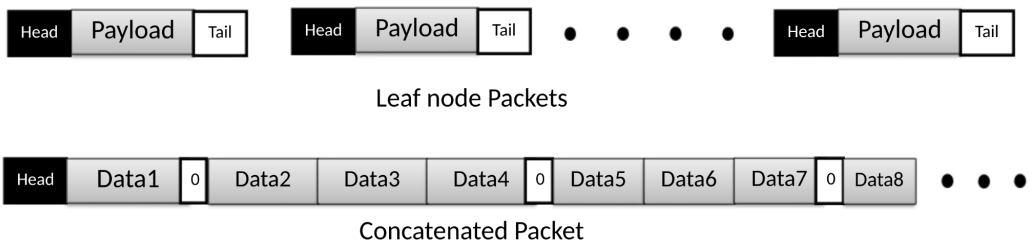


Figure 9. Concatenate packets based on packets' size

4. ENTROPY AND DATA DISTRIBUTION MODEL

Based on Shanon's entropy theory, the minimal number of bits needed to represent each value of the source data can be calculated using Equation 10 where p_i is the probability of the appearance of each symbol.

$$H(X) = -\sum_{i=0}^{N-1} p_i log_2 p_i \tag{10}$$

The entropy indicates how much the input sensing data can be compressed. The Equation 10 shows Shannon entropy determined by the probability of the data elements in the set. Since the distribution of the sensing data is not known before sensing and therefore, to evaluate the compression algorithms' performance, we propose a temporal and spatial data model that could simulate the delta (Δ) values of the real-world sensing data.

4.1. TEMPORAL AND SPATIAL DATA APPROXIMATION AND REGRESSION

The temporal property means, in a short sensing period, two consequent sensor data have similar values. The spatial property means the nearby sensor or the same sensor moving nearby, the sensor data will have the similar values. Below, we discuss the data approximation using regression, the probability mass function of the bits in the compressed sensor data and the possible compression strategies.

For data with temporal locality, we find that the sensing period can directly affect the statistical distribution of the delta values. Intuitively, if the sensing period is small, the delta values of the sensing data poses to be small because the data do not have much time to change. We describe the distribution of the delta values of the sensing data with the temporal locality mathematically in the following sections.

Assumptions and preconditions: When the sensor data have the temporal locality property periodically, we only consider data from one period only. In that period, the data have three stages; the ascending stage, the stable stage and the descending stage. We perform linear regression at each stage and use a straight line with a fixed slope β_1 as the linear model. Note that in the following equations, x_i refers to the time i and y_i refers to the observed sensing value at time i. We can find the β_0 and β_1 by minimizing the Pearson's correlation coefficient at each stage.

$$y_i = \beta_0^i + \beta_1 \times x_i$$

Regression and approximation: In each period, we approximate the data using the following three linear regression models; the ascending linear line model (11), the descending linear line model (12) and the static linear line model(13). After proper slicing of the dataset and regression, the average residual of the linear regression should fall in the 10% criteria which indicates a very small correlation. We assume that the residual is white noise with the mean value zero.

$$y_i = \beta_0^{asc} + \beta_1^{asc} \times x_i \text{ where } \beta_{asc} > 0; \tag{11}$$

$$y_i = \beta_0^{des} + \beta_1^{des} \times x_i \text{ where } \beta_{des} < 0; \tag{12}$$

$$y_i = \beta_0^{sta} \tag{13}$$

4.2. PROBABILITY DISTRIBUTION OF Δ OF THE SENSOR DATA

The Δ is the difference of two sensor data in the adjacent time period, $\Delta = Value_i - Value_{i-1}$ where $Value_i$ is the sensing value at time i. In sensing period, say T_s, the expected value changes can be calculated with Equations 11, 12 and 13. We calculate the average value changes E_Δ in the sensing period using the Equation 14. As each sensing is independent

of others, the distribution of each sensing period follows the Poisson distribution shown in Equation 15 where k indicates the steps of the data change in each sensing period. For example, consider a temperature sensor in the sensing period T_s where the temperature of this sensor is 0.1 degrees higher than the sensing data in the last period. Suppose the sensor has the sensing ranges from -40 degree to 123.8 degrees and the sensing result is in 14 bits binary format. In that sensing period, the total steps of the temperature changes can be given as $\frac{2^{13}\times 0.1}{123.8-(40)} \approx 10$ which results in a 5 bits delta value of the temperature. Note here that we need to consider the positive or negative delta values. For example, the positive 10 steps can be represented as 10100 which ends with 0 while the negative 10 steps can be represented as 10101 which ends at 1.

$$E_\Delta = \begin{cases} \beta_{asc} \times T_s \text{ When in ascending stage} \\ \beta_{des} \times T_s \text{ When in descending stage} \\ 0 \text{ When in static stage} \end{cases} \tag{14}$$

$$P_\Delta(k_\Delta) = \begin{cases} \frac{\beta_{asc}^{k_\Delta} e^{-\beta_{asc}}}{k_\Delta!} \text{ When in ascending stage} \\ \frac{|\beta_{des}|^{k_\Delta} e^{-|\beta_{des}|}}{k_\Delta!} \text{ When in descending stage} \\ 0^{k_\Delta} \text{When in static stage} \end{cases} \tag{15}$$

Considering the accuracy range of the sensors, based on the $3-\sigma$ rule, 99.7% of the sensing data should within the accuracy's upper-bound $3\sigma\prime$ and the accuracy's lower-bound $-3\sigma\prime$. The PMF distribution should follow the discrete approximation of the normal distribution with $\mu = 0, \sigma = \sigma\prime$ which is the binomial distribution Equation 16 with $n = 4 \times \sigma^2, p = 1/2$:

$$P_{bino}(k) = \binom{n}{k + n/2} p^{k+n/2} \times (1-p)^{n/2-k} \tag{16}$$

The noise of the sensor is independent from the sensing attributes so we can join these two PMFs 15 and 16 using the following equation:

$$P_{XY}(x, y) = P(X = x, Y = y) = P_X(x) \times P_Y(y) \tag{17}$$

Thus, we can get: $P_{k=k_\Delta+k_{bino}}(k_\Delta, k_{bino}) = P_\Delta(k_\Delta) \times P_{bino}(k_{bino})$

Next we want to find the PMF in terms of $k = k_\Delta + k_{bino}$ at each stage. For the convenience of the calculation, we assume the noise, within the sensor accuracy, is in the range of $[-3\sigma, 3\sigma]$. For stage 1, $k \in [-3\sigma, n + 3\sigma]$ where n is the positive sensing limitation. We get:

$$P_k(k) = \sum_{k_{bino}=-3\sigma}^{3\sigma} P_{bino}(k_bino) * P_\Delta(k - k_{bino})$$

For stage 2, $k \in [-n - 3\sigma, 3\sigma]$ where -n is the negative sensing limitation. We get:

$$P_k(k) = \sum_{k_{bino}=-3\sigma}^{3\sigma} P_{bino}(k_{bino}) * P_\Delta(k - k_{bino})$$

For stage 3, k is ranging from $[-3\sigma, 3\sigma]$. We get:

$$P_k = \binom{n}{k + 3\sigma + n/2} p^{k+3\sigma+n/2} \times (1 - p)^{n/2-(k+3\sigma)}$$

As the number of bits of the compressed data is only determined by the number of bits in the delta value $K = log_2(k)$, we only need to consider the PMF of K. Thus, we can represent each stage's PMF function P_K using each model's P_k as follows.

$$P_K(K) = \sum_{k=2^{K-2}}^{2^{K-1}-1} P_k(k) \tag{18}$$

Let's assume that the number of ascending sensing periods in stage 1 be np_1, the number of descending sensing periods in stage 2 be np_2, and the number of sensing periods in stage 3 be np_3. As discussed above, we assume that the expected delta value is zero over the long period. Thus, we have the following derivation:

$$np_2 = \frac{np_1 * \beta_{as}}{\beta_{de}} \tag{19}$$

So the distribution of the number of bits in the delta value becomes:

$$PMF(K) = \frac{np_1 * P_K(K)_{stage1}}{np1 + np2 + np3} + \frac{np_2 * P_K(K)_{stage2}}{np1 + np2 + np3} + \frac{np_3 * P_K(K)_{stage3}}{np1 + np2 + np3} \tag{20}$$

Consider Intel-Lab environmental data [37] which records the temperature, humidity, and light intensity; here we only consider the temperature and humidity as both of which have the temporal and spatial locality. From [38], in a single day, the indoor temperature rising period is about 6 hours when it increases by 1.6 degrees, descending temperature period is about 2 hours when it decreases by 1.6 degrees and the constant temperature period is about 16 hours. The ratio of these three time periods is the $3 : 1 : 8$. The three periods of the humidity has the value change of +36%, -36% and 0% whose ratio is the same as the ratio of the temperature. Thus, for distribution (Equation 15), stage 1 is for the rising period of the temperature and humidity where $\beta_{temp} = 1$ and $\beta_{humi} = 7$ based on the approximate sensing period of 2 minutes. Stage 2 is for the temperature and humidity descending period. Stage 3 is for the temperature and humidity holding period where $\beta_{temp} = 0$ and $\beta_{humi} = 0$. In Equation 19, we set $np_1 = 3, np_2 = 1, np_3 = 8$, thus, we can get $\beta_{de} = \frac{np_1 * \beta_{as}}{np_2}$. After studying the sensor's datasheet [39], we find the accuracy for the temperature is 0.5 degree, and the accuracy of the humidity is 3.5%.

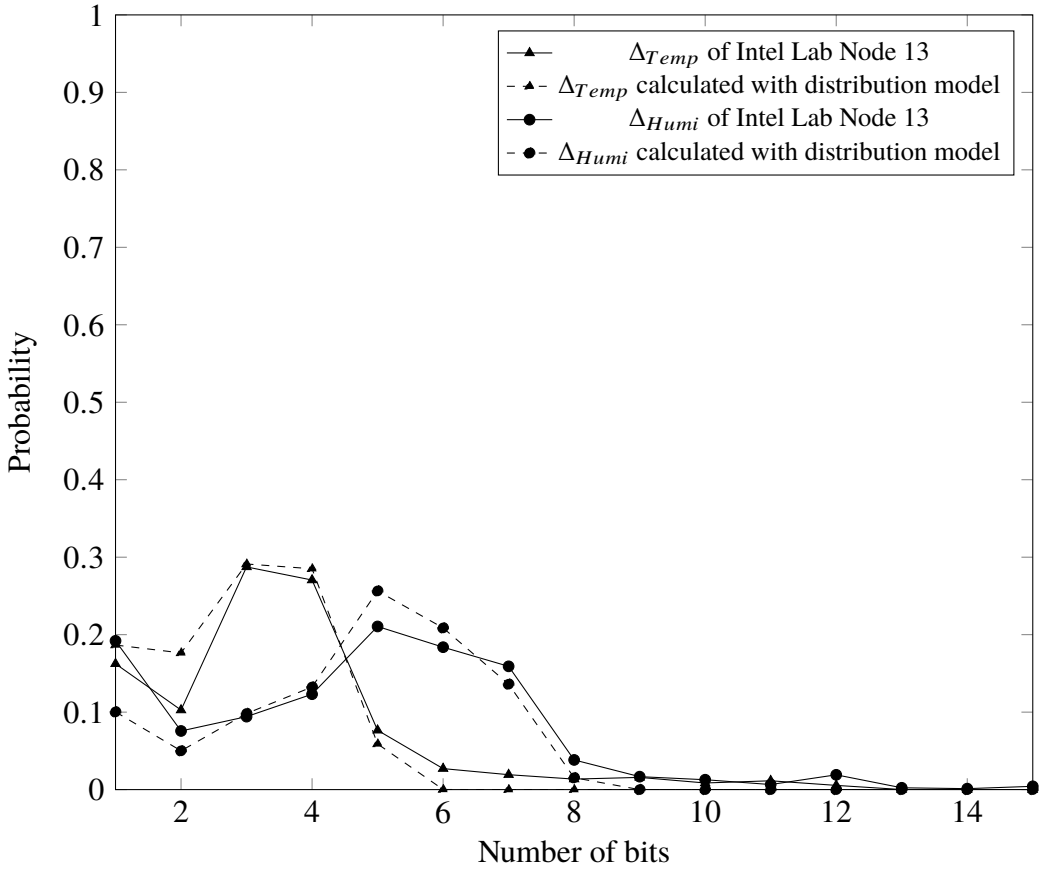


Figure 10. Calculated PMF of the number of bits in the Intel lab data

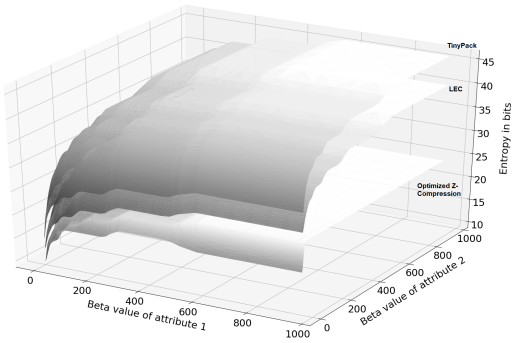
Consider the sensor resolution of the temperature as 0.01 and the sensor resolution for the humidity as 0.03%. As the accuracy means the sensing value should have high probability to fall in the accuracy range, we set $6\sigma = \frac{Accuracy}{resolution}$ which gives us $\sigma_{Temperature} = 3.1$ and $\sigma_{Humidity} = 7.3$. Then we can calculate $n = 4 \times \sigma^2$ by the distribute (Equation 16) postulate which is the discrete approximation of the normal distribution. Then after joining of 15 and 16 using 20, we get the approximate distribution of the number of bits of the indoor temperature and humidity as shown in Figure 10. Our distribution model represents the number of the bits of the sensing data correctly with less than 10% distortion. Using this data model, we can simulate different temporal and spatial sensing data by modifying the five arguments defined in this data model. We then evaluate the performance of different sensor compression algorithms on compressing the data generated using this data model.

4.3. PERFORMANCE EVALUATION USING TEMPORAL AND SPATIAL DATA MODEL

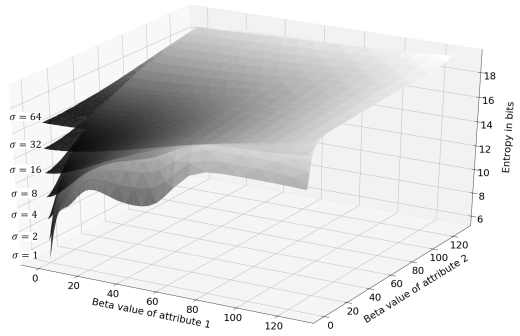
Recall the data model mentioned in Section 4.2 where five attributes which affect the distribution of the sensing data with the temporal and spatial property are given. The conjunction of the PMF of the sensing data model can be seen as the combination of three Poisson distribution with variance σ. The β values of stage 1 and stage 2 of the sensing period determine the center of the PMF of the data distribution. The stage period $n1, n2, n3$ determines the proportion of each Poisson stage in the whole sensing period. Suppose we have two independent randomly selected sensing attributes say attribute 1 and attribute 2 which need to be compressed with 300,000 samples. To study the effectiveness of optimized Z-Compression, TinyPack, and LEC algorithms on these two attributes' sensing data, we vary the β value, the σ value and the ratio of each stage period. We use the Equations 3,4, and 5 given in Section 4.2 to estimate the average number of bits in the compressed data. The result is shown in figures 11a,11b, 11c and 12.

In Figure 11a, we can notice that when compressing two independent attributes with the distribution model (mentioned in Section 4.2) that has $\sigma = 5$ and $1 : 1 : 0$ stages ratio, the optimized Z-Compression could always beat the TinyPack and LEC on the compression ratio independent of the β values of two attributes. We can also see that closer the two β values are, the better the compression ratio is. It is because a fewer number of extra bits will be added if two attributes are similar in the number of bits when using the optimal Z-Compression.

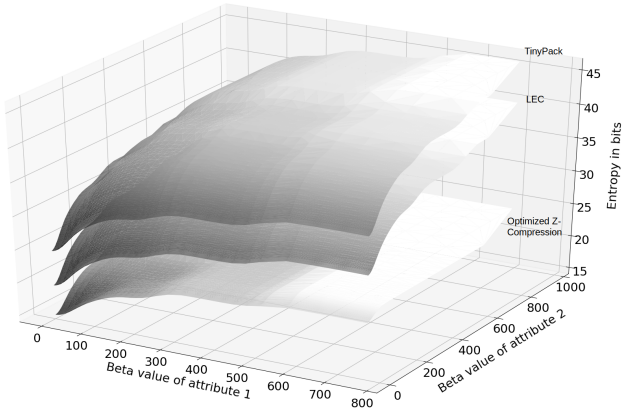
In Figure 11b, we can notice that when we increase the σ value of the sensing data distribution model, the average number of bits of the compressed data increases. The effect is more significant when the β value is small for the two compressing attributes. When the β values are more than twice of the σ value, the effect of the σ value on the number of bits of the compressed data is trivial.



(a) Average number of bits in the compressed data with different β values



(b) Average number of bits in the compressed data with different β and σ values

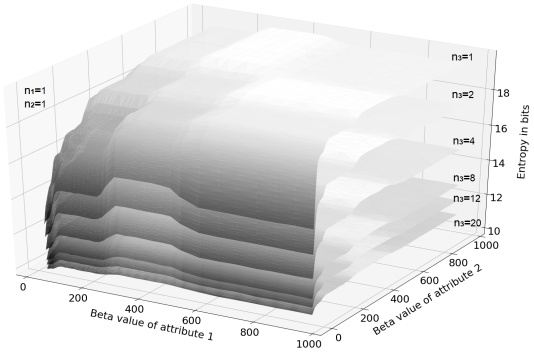


(c) Average number of bits in the compressed data with $\sigma = 64$

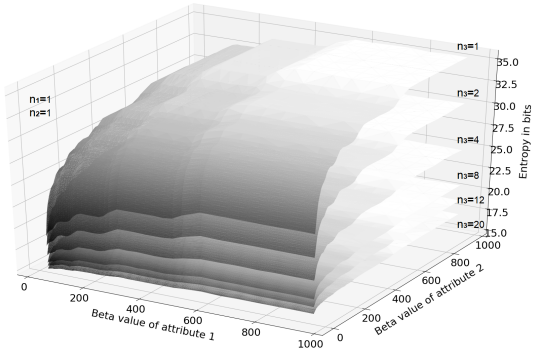
Figure 11. Average number of bits in the compressed data with different β and σ values for optimal Z-Compression, LEC, and TinyPack.

Figure 11c shows although larger σ value will introduce extra bits for the optimal Z-Compression, it is still better than the TinyPack and LEC for sensing data with *beta* values ranging from 1 to 1000. The reason is that TinyPack and LEC create more overhead in coding larger values.

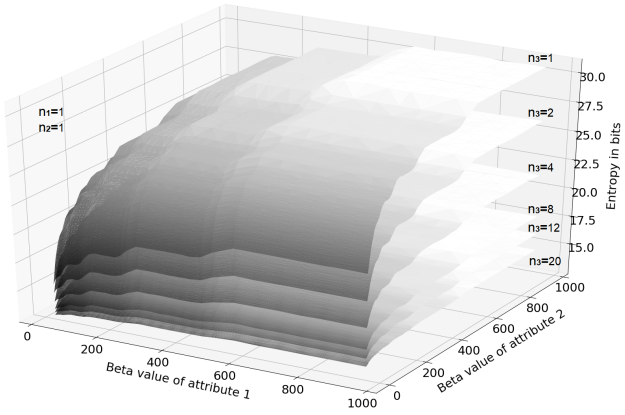
The stage ratio of the sensing data, mentioned in Section 4.2, will affect the average number of bits of the compressed value for Optimized Z-compression, TinyPack, and LEC. When the ratio of stage 3 in the sensing period is higher, suggesting the data has a high probability to be unchanged in the sensing period, fewer average bits are needed for the compressed value. Figure 12 shows for each ratio of stage 3 in the sensing period,



(a) Compressing using Optimized Z-compression



(b) Compressing using TinyPack



(c) Compressing using LEC

Figure 12. Average number of bits of the compressed values against different ratio of stage 3 (n_3) in the whole sensing period. Stage 1 and stage 2 has ratio $n_1 : n_2 = 1 : 1$, $\sigma = 5$

Optimized Z-compression performs better than LEC and TinyPack. Thus, the stages' ratios of the sensing period will not affect the rank of these three compression algorithms in the evaluation.

4.4. OBSERVATION

In this section, we proposed a data model with less than 10% data distortion to simulate the Δ values of the temporal and spatial sensing data. Next, we evaluated three different compression algorithms against this model which show that when the β value is small, meaning the data with a lower number of bits appear more frequently, Optimized Z-compression, LEC, and TinyPack work well. The reason is that LEC and TinyPack both use

the codebook which assumes the higher occurring probability of lower bit symbols like 0,1, and -1, so when the number of bits of the sensing data decreases, their compression ratio is close to each other (including of Z-compression). That is, LEC and TinyPack are designed for the data distribution with decreasing probability of occurrence when the number of bits of the data increases. However, when the β value increases, meaning the number of bits of the sensing attributes increases, the performance of LEC and TinyPack decrease significantly. It is because When the β value is small, the code of LEC and TinyPack fits the distribution model better, so the compression overhead is smaller. However, when the β or σ value increases, the distribution of the data diverges from the distribution model of LEC and TinyPack, thus the performance degrades. Whereas Optimized Z-compression does not depend on the data distribution only. It exploits the known information on how many attributes appear in compressing the sensing data and their average number of bits (better than estimating the bits using Shannon's entropy model or Huffman coding), therefore achieving better compression performance than the LEC and TinyPack.

5. Z-COMPRESSION IN HIGH STREAM RATE WSNS

In a wireless sensor network, there can be hundreds of wireless sensors nodes. As the number of sinks is limited, each data link to the sink is limited by the bandwidth of that sink. The bottleneck of the bandwidth is at the last hop of the data link which is the hop after the sink.

Both Naive Z-compression and Optimized Z-compression can be easily applied to different applications of WSNs. The best strategy to improve the throughput in a WSN that has high data streaming rate is to compress the data locally and concatenate the compressed data at the intermediate node.

To show, we assume a wireless sensor network organized as a tree structure where data will be transmitted from lower level to higher level in the tree. We assume that the root of the wireless sensor network is the base station. In such systems, the intermediate

nodes not only need to perform sensing but also need to do routing from the lower level to higher level nodes. As stated earlier, the energy consumption rate and bandwidth occupation by the intermediate nodes are much more critical than the leaf nodes. The lifetime of a wireless sensor network organized as a tree is the time elapsed until the first node in the network depletes its energy (like an intermediate node). In our experiments, we consider the intermediate node as the bottleneck for both energy and bandwidth consumption. Figure 13 shows a WSN organized as a tree structure with $R = 3$ where R stands for the number of children of each intermediate node. Every node will sense, compress and transmit local data at the same rate. The intermediate nodes will also forward the upstream data from their children to their parent node. In this case, the leaf nodes only need to handle the data generated by themselves while the intermediate nodes not only need to compress and transmit the data produced not only by themselves but also need to forward their R children's data to downstream nodes.

Figure 14 shows how small energy saving locally at a node can provide a considerable benefit to the intermediate nodes with more than one children in the tree structure. In the WSNs with a tree structure and R greater than 1, the packets the downstream nodes receive will increase exponentially when the number of hops increases. Also, with the help of the local compression algorithm, upstream nodes will generate fewer packets which will save the bandwidth at the intermediate nodes and mitigate the potential network congestion in WSNs.

6. Z-COMPRESSION IN LOW-POWER LISTENING WSNS

Low-power listening WSNs are widely used in real-world applications because they could extend the lifetime of the network by reducing the duty cycle of the sensors in the network. In low-power listening WSNs, nodes periodically fall asleep to save the energy used while listening to the channel. In each period, the sensor nodes will wake up for a short-time to listen to the channel. If the node receives a packet, it will wake up for a

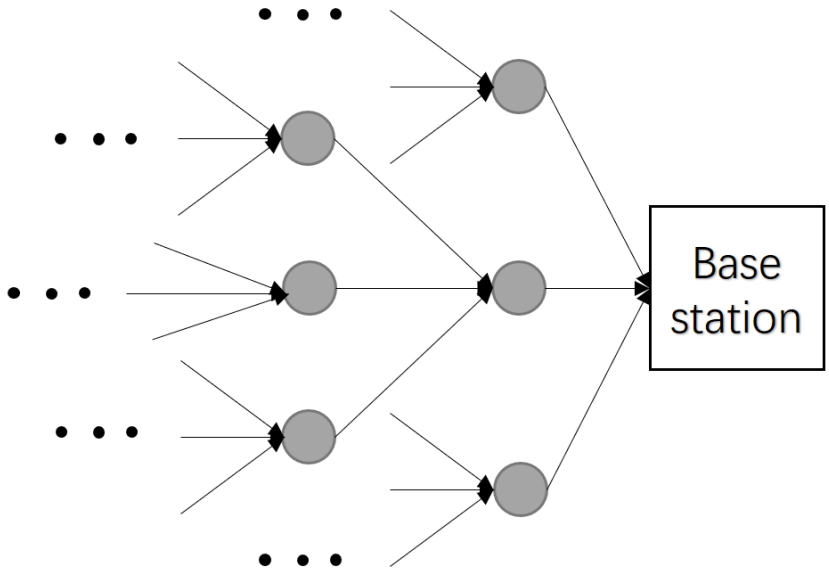


Figure 13. Tree structure with R=3

longer-time to receive additional packets and to compute and perform routing. However, if the node does not receive any packets, it goes back to sleep after the waiting time is over. In low-power listening WSNs, the only criteria to judge the energy consumption of the network is the duty cycle of all the sensor nodes in the WSNs. In this section, we are going to discuss how the Z-compression could reduce the duty cycles in the low-power listening WSNs as well as reduce the distortion (the difference between real data and sensing data) of the sensing data.

In a wireless sensor network with one sink node, we assume that the physical event delta value S_i has spatial correlation with the interested region S. The previous paper [40] modeled the physical phenomenon as joint Gaussian random variables (JGRVs) at each observation point i with zero mean and with σ_S as the variance. The observed sample at node i will be the sum of the physical event's value S_i plus the observation noise N_i which has zero mean and variance σ_N. The measured distortion can be calculated using the following equation:

$$D_E(M) = E[(S - \hat{S}(M))^2]$$

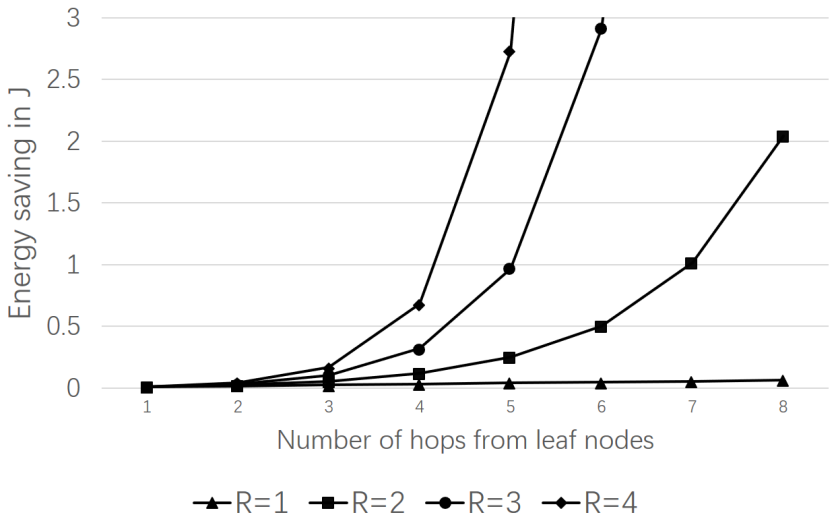


Figure 14. Energy savings at intermediate nodes with different number of hops and children when compressing ZebraNet data using Optimized Z-compression vs. no compression

Here S is the true value and S(M) is the reporting value. Based on [41], we define the distortion function 21 where $D_E(M)$ is the distortion of all the reporting message M.

$$D_E(M) = \sigma_S^2 + \frac{\sigma_S^4}{M(\sigma_S^2+\sigma_N^2)}(2\sum_{i=1}^{M}\rho_{(S,i)} - 1) + \frac{\sigma_S^6}{M(\sigma_S^2+\sigma_N^2)^2}\sum_{i=1}^{M}\sum_{j\neq i}^{M}\rho_{(S,i)} \tag{21}$$

From the Equation 21, we find that it is not necessary to collect all the data from the location of interest to keep the distortion low. If we choose several reporting nodes wisely from the area of interest, we can reduce the number of sensing nodes and still get highly reliable data.

In [41], the reporting nodes are selected using Lloyd's algorithm. The data from the reporting nodes send back to the sink node directly. However, in the low-power listening WSNs, the routing nodes need to wake up for a period doing the radio transmission. More the routing path we use, more the energy will be used as the routing node will not go to sleep

until it receives the acknowledgment from its successor. To reduce the usage of energy on the routing path, we can compress the sensing data using Z-compression and concatenate the reporting packets. Thus, we can reduce the duty cycles by reducing the wake-up time of the nodes on the routing path.

For example, in WSN in Figure 15, the base station will select the reporting sensors using the Lloyd's algorithm. After broadcasting the request to the areas of interest, the reporting sensors will report the sensing data as follows. First, the reporting node will sense and compress the data locally using Z-compression. Second, the reporting nodes will send the compressed data back to the broadcast center which was selected by the base station. Next, the sensing data will be concatenated in the broadcast center and then send back to the base station through the reliable route (which uses the same route the request message used).

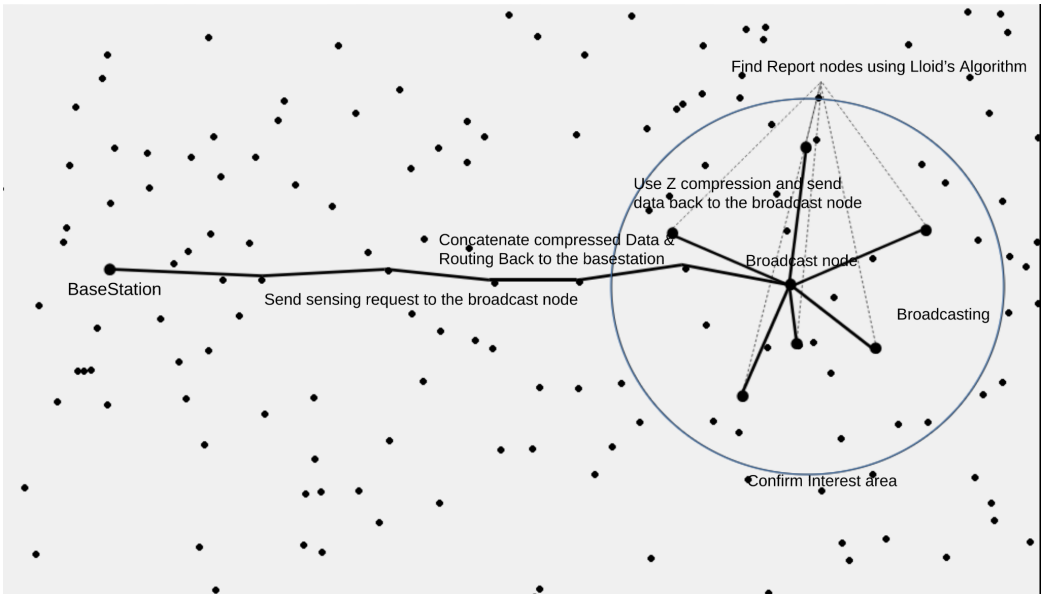


Figure 15. Data collection in low power listening WSN using Z-compression along with data concatenation

7. EXPERIMENTAL EVALUATIONS

We evaluated and compared compression algorithms using the following three types of experiments involving high-stream rate WSNs' data compression, local block data compression, and data compression in low-power listening WSNs.

7.1. Z-COMPRESSION IN HIGH STREAM RATE WSNS

The size of the data packets in wireless sensors is usually restricted to 128 bytes with 250kbps bandwidth, but the data streaming rate of each packet can range from seconds to hours. For example, the ZebraNet senses location data every few minutes. The Intel Berkeley Lab sensor network application collects temperature, humidity, and light lumen at different periods whereas the accelerometer on the vehicle collects 3-axis acceleration values, and the vehicle tracking application collects communicating vehicle's location, speed, and altitude data much more frequently.

7.1.1. Experimental Setup and Configurations. To demonstrate the effectiveness of our proposed Z-compression scheme in real-world situation, we tested it against different types of real-world multi-modal data sets such as GPS data [42], environmental data [37], accelerometer data [43], and vehicle tracking data [44]. The attributes in each dataset are shown in Table 4. Two common attributes which these data sets share are timestamp and node ID. These two attributes are used as the primary key of the local packet. We cannot compress the primary key because the intermediate nodes need to identify where the packet has come from as well as when the sampling starts. We assign the timestamp as fixed two bytes and the node ID as fixed one byte in the local packet, and we use a variable length packet for the compressed sensing values. The compression ratio CR in the performance matrix is defined as the compressed data length over the size of uncompressed data as shown in Equation 22. We use a base station to accumulate the number of bytes of all n compressed packets. $\sum_1^n L_{compressed}$ is the compressed data size and $\sum_1^n L_{original}$ is the original data size.

$$CR = \frac{\sum_1^n L_{original}}{\sum_1^n L_{compressed}} \tag{22}$$

To find the energy cost of the intermediate nodes, we use PowerTOSSIM-Z to simulate the energy consumption in WSN. The tool will calculate the CPU cycles and radio usage at each node. It then uses the predefined power model to generate the power consumption at each node in the experiment. Data is inserted into the leaf nodes using the python script.

We notice that when we increase the sampling rate close to a certain interval of two consecutive sensing samples, the packets drop start happening at some intermediate nodes, and at the sink node. Here, we define the sampling rate as the total number of sensing samples per second. To find the effectiveness of the compression algorithms on the sampling rate, and the maximum sampling rate a WSN can have, we define Equation 23 which outputs the approximate maximum sampling rate of the WSN. Here $T_{ap} = 30.31\%$ is the maximum experimentally normalized throughput of IEEE 802.15.4 radio in application layer [45], $V_{ch} = 250kbps$ is the channel speed of the radio [46], S_{data} is the uncompressed size of an original sensing sample and CR is the compression ratio of respective compression algorithm.

$$sampleRate_{max} \approx \frac{CR \times T_{ap} \times V_{ch}}{S_{data}} \tag{23}$$

7.1.2. Compression Performance Comparison. This experiment evaluates the average compression ratio in compressing 5000 data items from each of the four different datasets listed before. The leaf nodes will do the compression locally. Then the compressed packets are concatenated at the intermediate nodes using the Algorithm 4. The evaluation results are shown in Figure 16. The compression ratios in Y-axis is calculated using the Equation 22. Note that a higher compression ratio means better compression performance.

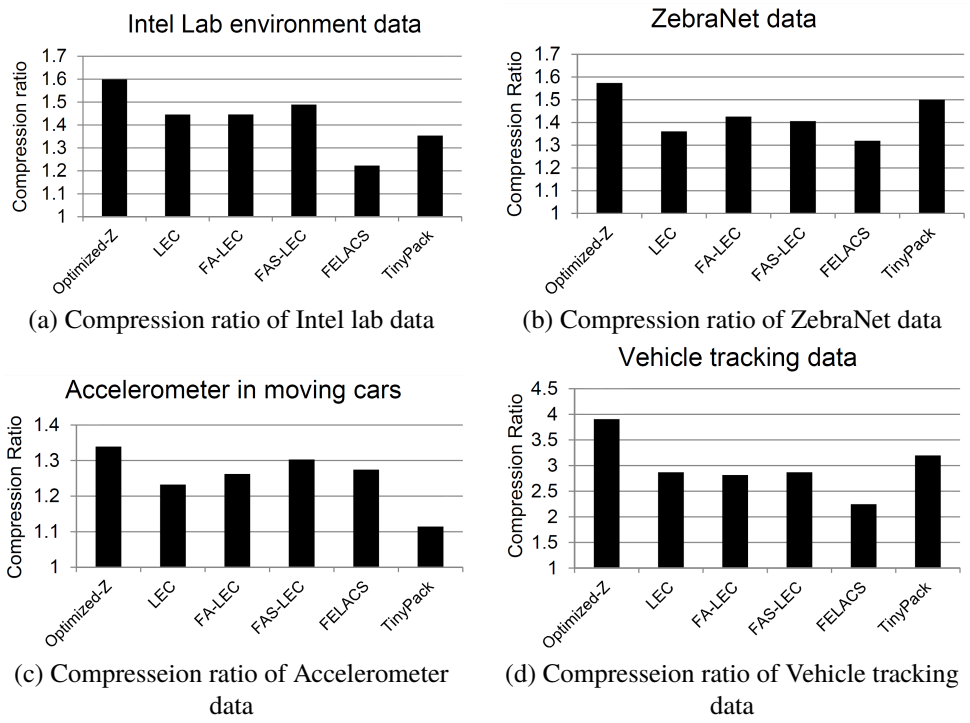
On compressing vehicle trace dataset, Z-compression has more than 30% improvement over other compression algorithms. On compressing other three datasets, Z-compression has between 5% to 30% improvement over other compression algorithms. Z-compression has better compression ratio improvement when using the vehicle trace dataset because this dataset has more unchanged samples than others. Z-compression also has an all-is-well function that can reduce the multiple zero delta values into a single Z-value of zero. Also, the performance of Z-compression is stable whereas we can see that other compression algorithms, for example, TinyPack and LEC, have large performance variation when the input data distribution changes. LEC performs better for the dataset with smaller delta values. However, TinyPack has a better compression ratio over the datasets which have more smaller delta values.

Table 3. Maximum approximate sampling rate using different compression algorithms with data concatenating

data set name	Z-comp	LEC	FA-LEC	FAS-LEC	FELACS	TinyPack	No concat
Intel Lab environment data	1376	1244	1245	1282	1053	1165	131
Accelerator in moving car	1409	1297	1328	1371	1341	1172	136
ZebraNet data	1656	1431	1500	1479	1389	1578	136
Vehicle trace data (V to V)	1947	1430	1403	1430	1119	1594	117

Adaptive-LEC improved the compression performance by using the real-time adaption. However, when adapting multi-modal sensing data, there can be more than two frequency center exists like what Figure 10 shows. That limits the performance of Adaptive-LEC. FELACS has the worst performance because it is not good at compressing multi-modal sensor data. It has the same drawback as Naive Z-compression. That is, when compressing skewed data, both Naive Z-compression and FELACS need to add '0' at the front of the smaller delta values to make their length equal to the length of the largest delta value. For example, in the Intel Berkeley lab environment dataset, the length of the delta values of the humidity is always larger than the length of the delta values of the light intensity. For the

multi-modal sensing data, the distribution of the length of each attribute can be different. That means the dataset is prone to be skewed like Figure 10. That is the reason FELACS can not perform well on compressing real-world multi-modal datasets.



(a) Compression ratio of Intel lab data

(b) Compression ratio of ZebraNet data

(c) Compresseion ratio of Accelerometer data

(d) Compresseion ratio of Vehicle tracking data

Figure 16. Compression ratio of real-time datasets

The result shows that the TinyPack achieves a higher compression ratio in compressing dataset with many "zero" delta values like in ZebraNet data in Figure 16.b and Vehicle tracking data in Figure 16.d. However, in compressing ZebraNet dataset, Optimized Z-compression algorithm achieves a higher compression ratio than Naive Z-compression and has about 15% improvement over the next best TinyPack algorithm. On compressing Intel Berkeley dataset, LEC achieves a higher compression ratio than Naive Z-compression. The reason is that the light lumen values in the IntelLab dataset do not change as frequently as the temperature and humidity values do, so the dataset is skewed. Thus, the delta values of temperature and humidity are much higher than delta values of the light lumen. However,

here our Optimized Z-compression groups temperature and light lumen together and then compressed with the humidity data. Thus, the result in Figure 16.a shows that optimized Z-compression gains about 10% improvement over the next best LEC.

On compressing Vehicle tracking data in Figure 16.d, we use the small code library and odd/even Optimized Z-compression. We notice that the attributes of the vehicle tracking dataset are evenly distributed with the average number of bits of the delta value close to 1. So for the eight attributes of the dataset, we generate a rule to compress pairs of attributes first; then compress pairs of encoded values recursively till only one Z-value is obtained. From Table 1, we can compress two zero delta values, which is represented as '1', into one bit Z-value of '1'. If all eight attributes of vehicle tracking data are zero, the final encoded result will be '11' (The first '1' is the placeholder). Therefore, Optimized Z-compression gets about 50% improvement over next best TinyPack.

Accelerometer dataset has an evenly distributed delta values. Therefore, Figure 16.c shows that Naive Z-compression achieves good performance on the compression ratio with an improvement of about 18.4% over LEC and 30% over TinyPack. However, as we discussed in the last section, the Optimized Z-compression uses the same compression rule as Naive Z-compression so the performance of Optimized Z-compression in Figure 16.c is same as of Naive Z-compression.

7.1.3. Energy Usage Comparison. These experiments are performed with TOSSIM simulator using PowerTOSSIM-Z [28]. We inserted data at the leaf nodes using the python script again. For some datasets such as Intel Lab and vehicle trace dataset, we inserted about 20,000 samples each time. After compression and concatenating, the number of packets forwarded to the sink is much smaller than the original 20,000 samples. The result is shown in Figure 17. As the compression ratio of Z-compression is better than the other compression algorithms for all the four datasets, the Z-compression reduces more packets than all others and thus, saving more energy and also it reduces the bandwidth usage in the network.

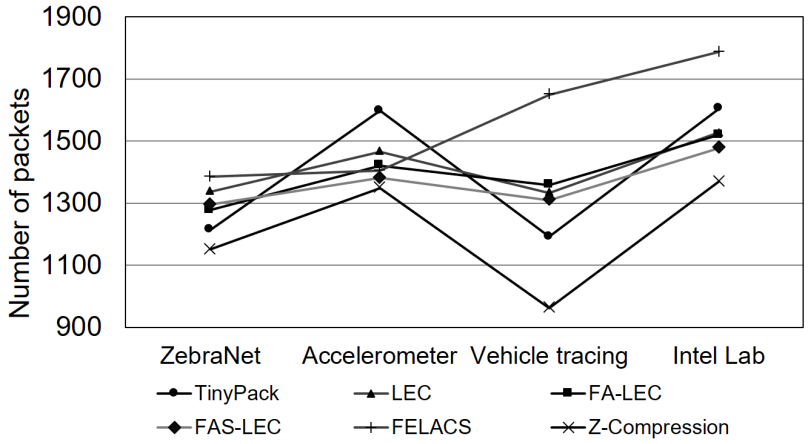
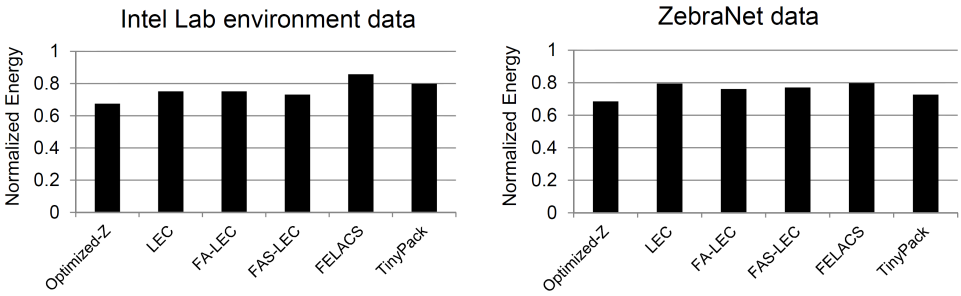


Figure 17. Total packets after compression and concatenating for 20,000 data samples

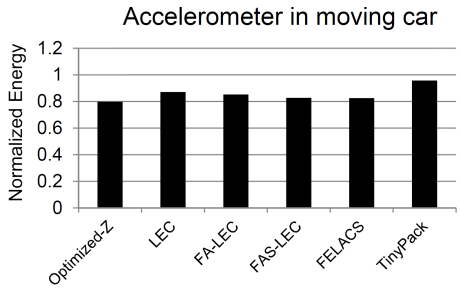
Note that as we are comparing different datasets and each dataset has a different number of samples in the energy comparison results, thus we use the normalized energy instead the real energy cost to show the effectiveness of each compression algorithm. The normalized energy is the ratio of the energy consumed by compressing or concatenating data and transmitting the fused packets over the energy consumed by only transmitting the fused packets. The experimental results are shown in Figure 18; the result shows that Z-compression provides the best energy saving for the WSN. It is because with the better compression ratio achieved with Z-compression, the intermediate nodes can concatenate much more leaf node data into a larger packet that reduces the radio usage which saves battery. The fact that the intermediate nodes do not perform the compression, and the overhead due to concatenating leaf node's payload is negligible, thus, we do not show the CPU energy usage in the result.

7.1.4. Approximate Maximum Sampling Rate. As we discussed in Section 5, the maximum sampling rate is the rate at which the maximum throughput of the sink node can be supported without dropping packets. It mainly depends on the compression ratio of the leaf nodes. The compression time will only determine the minimum sample interval of the leaf nodes in the WSN. The maximum sampling rate will not be affected by the compression

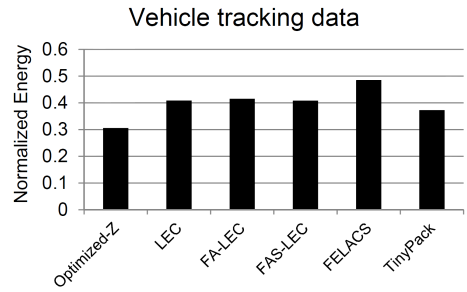


(a) Energy consumption of Intel lab data

(b) Energy consumption of ZebraNet data



(c) Energy consumption of Accelerometer data



(d) Energy consumption of Vehicle tracking data

Figure 18. Energy consumption of real-time datasets

time as we can increase the number of leaf nodes. Also, the time complexity of all these compressing algorithms is O(n). It takes about 30-60 milliseconds for compressing each sample including the sampling time.

Table 3 shows the approximate maximum sampling rate for compressing and concatenation of different data sets versus direct forwarding. The optimal Z-compression has the best performance according to the results.

7.2. LOCAL BLOCK DATA COMPRESSION

In wireless sensor networks, energy is the most critical factor for the lifetime of the network. Most of the power consumed by wireless sensor nodes is due to data transmissions using radio communication. The time radio is on mainly depends on the number of packets to be transmitted, which in turn depends on the packet size. Thus, the compression algorithm which achieves a better compression ratio will usually have a better energy saving due to the

reduced radio transmission time as it will send fewer packets. However, the microcontroller also consumes energy in compressing data as well. Thus, it is essential that we not only consider the compression ratio as the performance metric but also compare the energy savings of compression methods.

For the experiment here, different from the last which uses real-time datasets, the system model of the evaluation here is an only one-hop delay-tolerant wireless sensor network. We initialize two blocks of data as byte arrays. Each block is 528 bytes containing a sequence of continuous timestamp data. Each data item is composed of timestamp and sensing values. To exploit the temporal locality property of sensing attributes, We regrouped data by sensor category and types before compressing. For example, when compressing ZebraNet data, we group based on timestamp, the longitude, the latitude and the boolean values separately based on the sensor types. For the case without compression, local energy consumption is the energy required to transmit the whole block of data. For the node doing compression, the energy consumption is the sum of the energy needed to transmit the compressed data and the energy used in compressing. We define the energy saving as the energy consumed when compressing and transmitting data over the energy consumed in transmitting uncompressed data immediately.

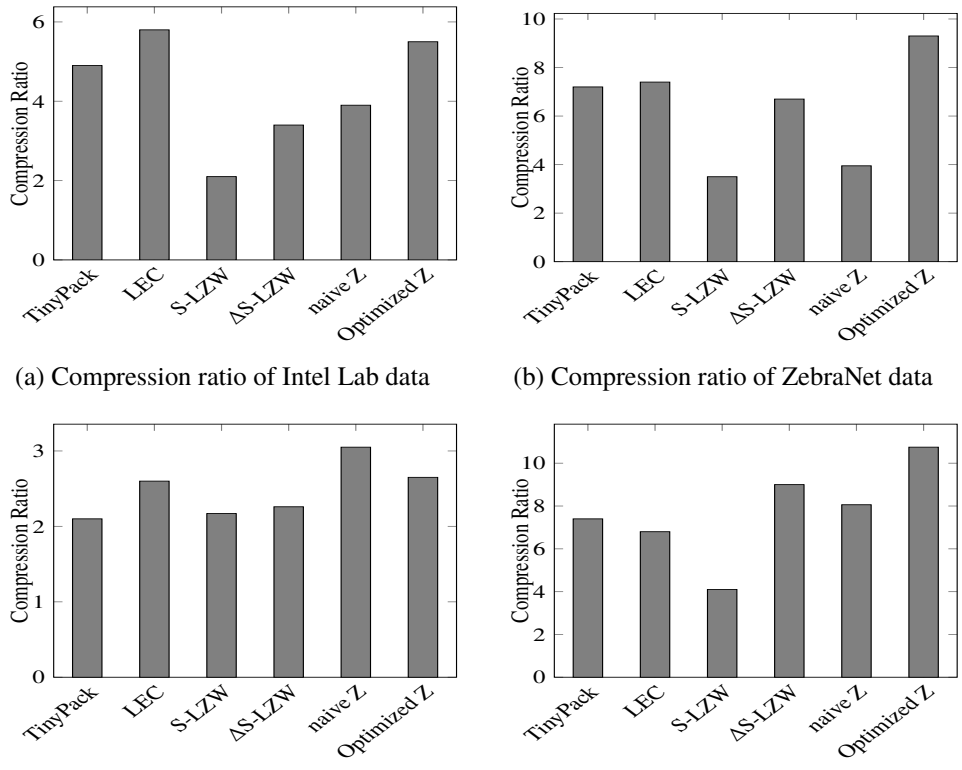
The compression algorithms we evaluated in this part are S-LZW (as it works traditionally on data blocks), Delta-S-LZW, LEC, TinyPack, Naive Z-compression and Three-round Z-compression. The size of the output packet is 114 bytes which are suitable for the TelosB mote. For S-LZW and Delta-S-LZW, we use the dictionary with 1024 entries. We propose here Delta-S-LZW that compresses the delta values rather than the actual values. It exploits the temporal locality and gives us a higher compression ratio as shown in Figure 19 and better energy saving as shown in Figure 20 than S-LZW. For TinyPack and LEC, Huffman code is generated respectively based on Figure 1. For Naive Z-compression and Three-round Z-compression, to reduce the compression overhead, we split the output packet into smaller blocks of the same size. The number of blocks under

a packet is predefined. In this experiment, we set this to be 6. After generating all the blocks of a packet using Naive Z-compression, we then merge those blocks to get the output packet. Note that each block needs extra one byte to indicate how many data items are in the block. Next, Three-round optimized Z-compression is proposed to improve compression performance on the datasets with many delta values as '0'. In this case, we compress a sequence of continuous '0' delta values together using LEC to encode the number of '0's. Then we apply Naive Z-compression on the newer dataset generated in the first round. We call this optimization Three-round Z-compression algorithm because it uses three rounds to get the result. Note that the first round that handles a sequence of '0' bits will add extra one bit to the delta values which are not 0s.

We regrouped sensor data by category and types before compressing. For example, for ZebraNet data, we group timestamp, the longitude, the latitude and the boolean values separately based on sensor types. For each group, temporal locality property is exploited. To reduce the compression overhead when using Z-compression, we set the sub-group size first and then compress data into groups with the size less than the sub-group size. After generating all the sub-groups, we apply Naive Z-compression on those sub-groups to get the final output. We also refer to this as Two-round Naive Z-compression to indicate that it uses two rounds of compression.

Compressing large blocks of local data is different from compressing smaller data packets. First, delays are not considered in this case. Second, the compressing produces data packets which have size limitations. Third, the time stamp is included. Fourth, the format of input data is the byte array.

Figure 19 shows the local compression ratio of four different kinds of datasets with five different compression algorithms. And, Figure 20 shows the normalized energy consumption for different compression algorithms using four different datasets. Naive Z-compression beats all the other compression algorithms on compression ratio and energy saving in compressing Intel lab and Accelerometer datasets shown in Figure 19.a, 19.c and



(a) Compression ratio of Intel Lab data

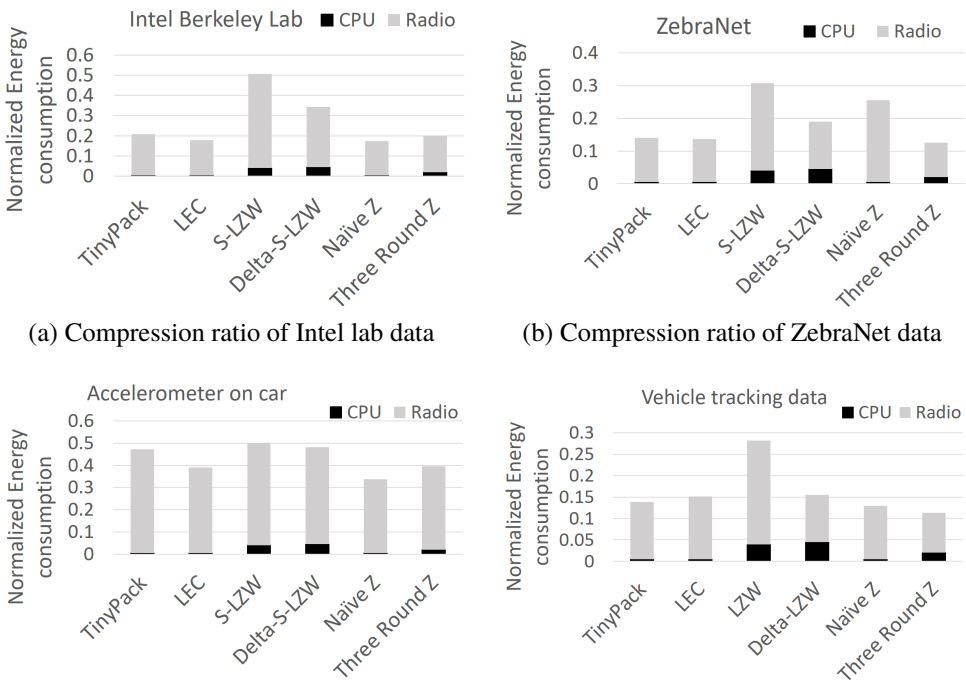
(b) Compression ratio of ZebraNet data

(c) Compresseion ratio of Accelerometer data

(d) Compresseion ratio of Vehicle tracking data

Figure 19. Compression ratio of different local datasets

Figure 20.a, 20.c for the reason that Naive Z-compression has less overhead in compressing evenly distributed dataset. Three-round Z-compression beats all the other compression algorithms on compression ratio and energy saving for compressing ZebraNet and Vehicle tracking datasets showing in Figure 19.b, 19.d and Figure 20.b, 20.d for the reason that it will decrease overhead by encoding sequence of '0' delta values into a single LEC code. Three-round Z-compression has better performance in compressing Vehicle tracking dataset than compressing ZebraNet dataset because Vehicle tracking dataset contains many '0' delta values than ZebraNet. The more '0' delta values a dataset contains, the more better Three-round Z-compression performs.



(a) Compression ratio of Intel lab data

(b) Compression ratio of ZebraNet data

(c) Compresseion ratio of Accelerometer data

(d) Compresseion ratio of Vehicle tracking data

Figure 20. Energy consumption in compressing different datasets

Datasets such as Accelerometer and Intel lab which contain fewer '0' bits items are not suitable for Three-round Z-compression algorithm, which is also validated by Figure 20.a and 20.c. As explained above, for these two datasets, Naive Z-compression is the best. Thus, our proposed Z-compression schemes also perform well for not so real-time time case. Although Delta-S-LZW always achieves better compression ratio, it is still not as good as other delta compression algorithms. It shows that delta compression has an advantage over local sensor data compression.

7.3. EXPERIMENTS AND EVALUATIONS OF OPTIMAL Z-COMPRESSION IN LPL WSNS

In this experiment, we use TOSSIM to simulate the wireless sensor networks. We use PowerTOSSIM-Z to measure the energy consumption of each node. The sensors are deployed in a rectangle area with length $\times$ width equal to 3600×2000. We assume that the

area is homogeneous and the sensors hold the same radio range in the area. The network is not sparse, and the sensors are uniformly deployed in the rectangle area. By configuring the range of the radios of the sensors, we can make sure that each sensor node has an average of six to ten direct neighbors. The sensor deployment rule is shown in Table 4. We did two experiments with a different number of nodes. In each experiment, we compare the energy consumption at each node using direct sensing and reporting the energy consumption using the compression and concatenation.

Table 4. Sensors deployment parameters

type	Experiment 1	Experiment 2
Number of nodes	990	2500
Area to deploy	3600×2000	3600×2000
Radio range	300	200
Energy model	micaz	micaz
Sleep period	2 second	2 second
Duty cycle	10%	10%
Area of interest	radius = 1000	radius = 600
# of Reporting nodes	10	10
# of Sensing attributes	10	10

TOSSIM does not perform real sensing. To simulate the data collection and compression, we use the data injection function in TOSSIM to insert data at each node. The data injection works as follows. First, choose the node to inject the data. Second,inject the artificial data at the set time. Third, continue the rest of the code. However, this data injection will increase the duty cycle of the sensor's which makes the simulation results incorrectly. So we hard-coded the sensing data in the header file and refer the header file from the NesC code of the sensors based on the node ID. Also, to ensure the routing accuracy, each sensor has also hard-coded its neighbors' ID in their source code. The 990 sensor nodes' result is shown in figures 21 and 22. And, the 2500 sensor nodes' result is shown in figures 23 and 24.

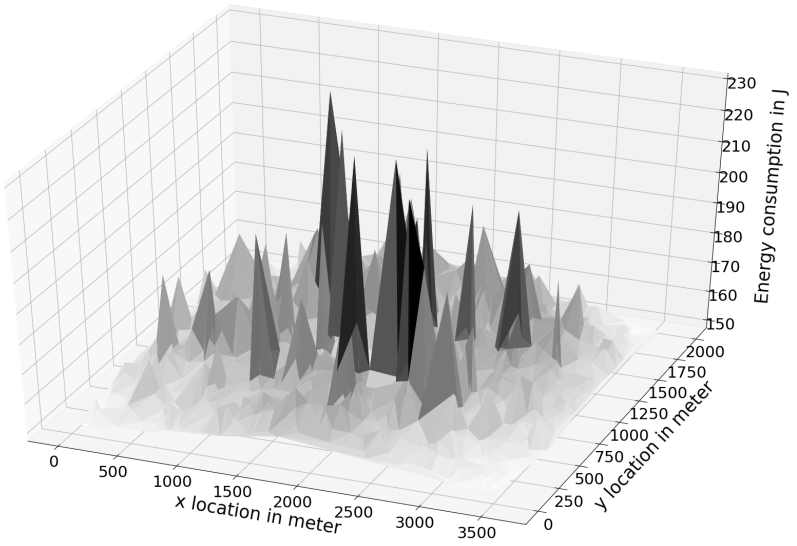


Figure 21. Energy usage of 990 node WSN in 10,000 seconds with 1250 sensing requests with no compression

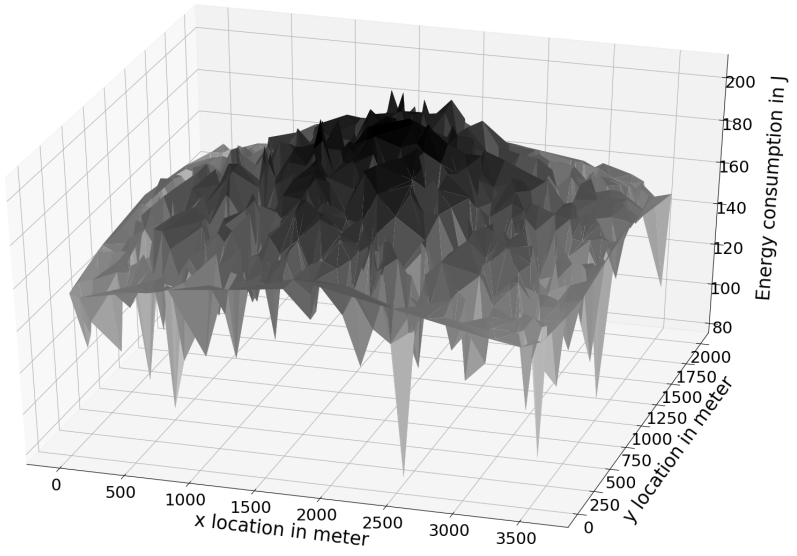


Figure 22. Energy usage of 990 nodes WSN in 10,000 seconds with 1250 sensing requests with optimal Z-compression along with data concatenation

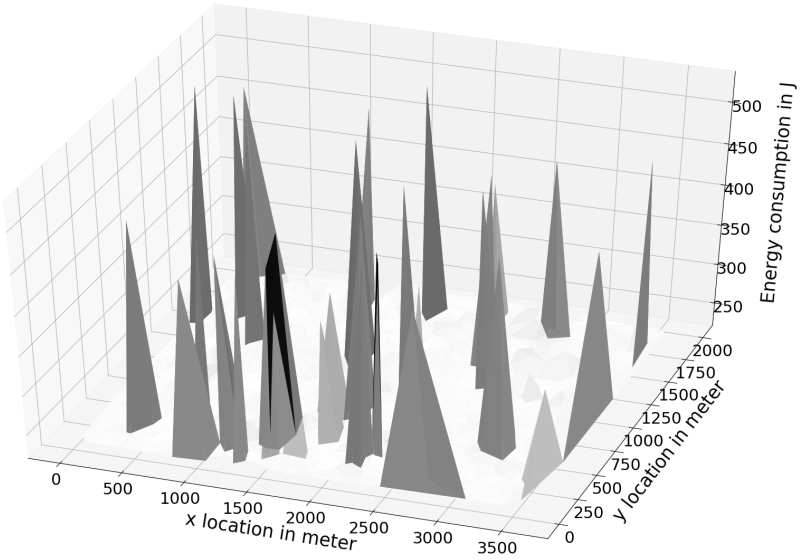


Figure 23. Energy usage of 2500 nodes WSN in 10,000 seconds with 1250 sensing requests with no compression

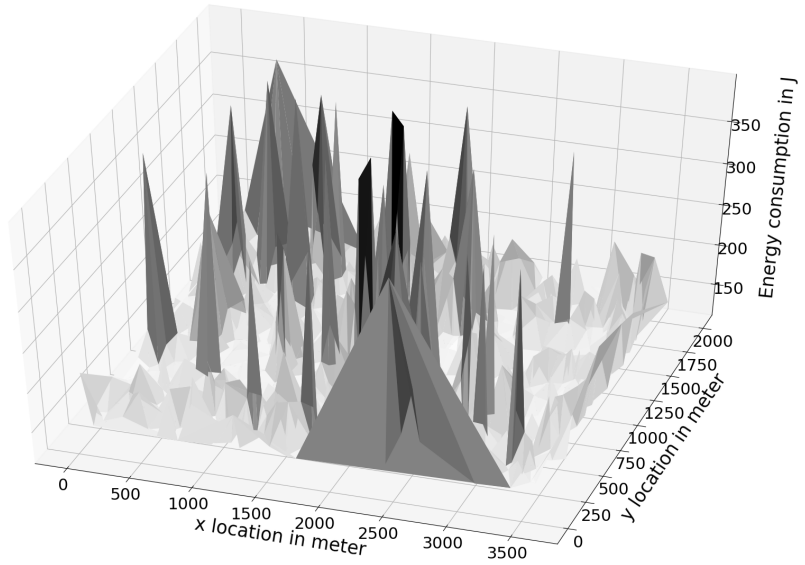


Figure 24. Energy usage of 2500 nodes WSN in 10,000 seconds with 1250 sensing requests with optimal Z-compression along with data concatenation

The simulation results using 990 nodes show that the optimal Z-compression along with the data concatenation reduces the average energy consumption by about 9%. It also balances the load of the network significantly by reducing the load on the intermediate nodes. Figure 22 is smoother than Figure 21 which directly transmits the reporting message back to the base station without compression. The improvement reaches about 24% at the nodes with peak loads. It significantly increases the lifetime of the whole network.

In the result with 2500 nodes in Figure 23 and 24, when using the optimal Z-compression, the average energy saving reaches 26%. The nodes with peak load have about 24% energy saving. We notice that the average energy saving for the 2500 nodes is more significant than that for the 990 nodes. The reason is that for both of these WSNs, the number of reporting nodes is the same. However, the number of hops in the path from the reporting nodes to the sink is more for the 2500 nodes than that of 990 nodes. Thus, Optimal Z-compression saves much more energy as it reduces the duty cycle of the sensors on the routing path of the data which reduces the number of reporting nodes.

8. CONCLUSION AND FUTURE WORK

In this paper, we proposed, based on Z-order, a multi-model Z-compression scheme for sensor data in wireless sensor networks to conserve bandwidth as well as energy. It compresses multi-dimensional data by exploiting the temporal and spatial locality. It reduces the packets size and allows the intermediate nodes to transmit less number of packets and thus, save energy, and being able to reduce the packet drops when streaming rate is high. We have performed several ToSSIM and TinyOS based experiments using four real-world sensor datasets. It performs much better when compared with well-known compression schemes like LEC, Adaptive-LEC and TinyPack using the compression ratio, energy usage and sampling rate as performance metrics. The Z-compression algorithm compresses multi-model sensing data locally in a real-time fashion, and thus, it can work with different MAC protocols to achieve further efficiency in WSNs. In the high-stream

rate WSNs, Z-compression improves the throughput thus increase the maximum stream rate of the network. In the low-power wireless sensor networks using asynchronous MAC protocol such as low-power listening (LPL), Z-compression could reduce the duty cycle of the nodes in the path from where the reporting data routes back. The experimental results also demonstrated that Z-compression could not only save the energy and bandwidth but also balances the load in the WSNs which could prolong the lifetime of the sensor nodes.

In future, we plan to implement the Z-compression in a sensor cloud [1], which provides on demand sensing as a service to users satisfying the QoS. With the help of lossless Z-compression, we can handle the maximum sensing request rate from many different clients without additional delays, and also maintain the QoS requests of users. Similarly, in IoT-based applications such as smart-city, Z-compression can adpat according to the device type and can handle the network heterogeneity as well to meet the application demands.

REFERENCES

[1] Sanjay Madria, Vimal Kumar, and Rashmi Dalvi. Sensor cloud: A cloud of virtual sensors. *Software, IEEE*, 31(2):70–77, 2014.

[2] John Burgess, John Zahorjan, Ratul Mahajan, et al. CRAWDAD dataset umass/diesel (v. 2008-09-14), September 2008.

[3] Enrico Perla, Art Ó Catháin, Ricardo Simon Carbajo, Meriel Huggard, and Ciarán Mc Goldrick. Powertossim z: realistic energy modelling for wireless sensor network environments. In *Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pages 35–42. ACM, 2008.

[4] Injong Rhee, Ajit Warrier, Mahesh Aia, Jeongki Min, and Mihail L Sichitiu. Z-mac: a hybrid mac for wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 16(3):511–524, 2008.

[5] Luca Anchora, Antonio Capone, Vincenzo Mighali, Luigi Patrono, and Francesco Simone. A novel mac scheduler to minimize the energy consumption in a wireless sensor network. *Ad Hoc Networks*, 16:88–104, 2014.

[6] Jun Long, Mianxiong Dong, Kaoru Ota, and Anfeng Liu. A green tdma scheduling algorithm for prolonging lifetime in wireless sensor networks. *IEEE Systems Journal*, 11(2):868–877, 2017.

[7] Michael Buettner, Gary V Yee, Eric Anderson, and Richard Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320. ACM, 2006.

[8] Yanjun Sun, Omer Gurewitz, and David B Johnson. Ri-mac: a receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 1–14. ACM, 2008.

[9] Jun Bum Lim, Beakcheol Jang, and Mihail L Sichitiu. Mcas-mac: A multichannel asynchronous scheduled mac protocol for wireless sensor networks. *Computer Communications*, 56:98–107, 2015.

[10] Tung-Wei Kuo, Kate Ching-Ju Lin, and Ming-Jer Tsai. On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: Np-completeness and approximation algorithms. *IEEE Transactions on Computers*, 65(10):3109–3121, 2016.

[11] Chen-Xu Liu, Yun Liu, Zhen-Jiang Zhang, and Zi-Yao Cheng. High energy-efficient and privacy-preserving secure data aggregation for wireless sensor networks. *International Journal of Communication Systems*, 26(3):380–394, 2013.

[12] Dylan McDonald, Stewart Sanchez, Sanjay Madria, and Fikret Ercal. A survey of methods for finding outliers in wireless sensor networks. *Journal of network and systems management*, 23(1):163–182, 2015.

[13] Nguyen Quoc Viet Hung, Hoyoung Jeung, and Karl Aberer. An evaluation of model-based approaches to sensor data compression. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2434–2447, 2013.

[14] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM SIGMOD Record*, 30(2):151–162, 2001.

[15] Chiranjeeb Buragohain, Nisheeth Shrivastava, and Subhash Suri. Space efficient streaming algorithms for the maximum error histogram. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 1026–1035. IEEE, 2007.

[16] Hazem Elmeleegy, Ahmed K Elmagarmid, Emmanuel Cecchet, Walid G Aref, and Willy Zwaenepoel. Online piece-wise linear approximation of numerical streams with precision guarantees. *Proceedings of the VLDB Endowment*, 2(1):145–156, 2009.

[17] G Kumar, K Baskaran, R Elijah Blessing, and M Lydia. A comprehensive review on the impact of compressed sensing in wireless sensor networks. *International Journal on Smart Sensing & Intelligent Systems*, 9(2), 2016.

[18] Francesco Marcelloni and Massimo Vecchio. An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks. *The Computer Journal*, 52(8):969–987, 2009.

[19] Massimo Vecchio, Raffaele Giaffreda, and Francesco Marcelloni. Adaptive lossless entropy compressors for tiny iot devices. *IEEE Transactions on Wireless Communications*, 13(2):1088–1100, 2014.

[20] Tommy Szalapski and Sanjay Madria. On compressing data in wireless sensor networks for energy efficiency and real time delivery. *Distributed and Parallel Databases*, 31(2):151–182, 2013.

[21] Jonathan Gana Kolo, S Anandan Shanmugam, David Wee Gin Lim, and Li-Minn Ang. Fast and efficient lossless adaptive compression scheme for wireless sensor networks. *Computers & Electrical Engineering*, 41:275–287, 2015.

[22] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[23] David A Huffman et al. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[24] Jeffrey Scott Vitter. Design and analysis of dynamic huffman codes. *Journal of the ACM (JACM)*, 34(4):825–845, 1987.

[25] Guy M Morton. *A computer oriented geodetic data base and a new technique in file sequencing*. International Business Machines Company New York, 1966.

[26] Bret Hull, Kyle Jamieson, and Hari Balakrishnan. Mitigating congestion in wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 134–147. ACM, 2004.

[27] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137. ACM, 2003.

[28] Enrico Perla, Art Ó Catháin, Ricardo Simon Carbajo, Meriel Huggard, and Ciarán Mc Goldrick. Powertossim z: realistic energy modelling for wireless sensor network environments. In *Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pages 35–42. ACM, 2008.

[29] Azad Ali, Abdelmajid Khelil, Piotr Szczytowski, and Neeraj Suri. An adaptive and composite spatio-temporal data compression approach for wireless sensor networks. In *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pages 67–76. ACM, 2011.

[30] Christopher M Sadler and Margaret Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 265–278, 2006.

[31] Sorabh Gandhi, Suman Nath, Subhash Suri, and Jie Liu. Gamps: Compressing multi sensor data by grouping and amplitude scaling. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 771–784. ACM, 2009.

[32] Henry Ponti Medeiros, Marcos Costa Maciel, Richard Demo Souza, and Marcelo Eduardo Pellenz. Lightweight data compression in wireless sensor networks using huffman coding. *International Journal of Distributed Sensor Networks*, 2014.

[33] Terry A Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, 1984.

[34] Tommy Szalapski and Sanjay Madria. Energy efficient distributed grouping and scaling for real-time data compression in sensor networks. In *2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)*, pages 1–9. IEEE, 2014.

[35] Dimitrios Lymberopoulos, Nissanka B Priyantha, and Feng Zhao. Towards energy efficient design of multi-radio platforms for wireless sensor networks. In *Information Processing in Sensor Networks. IPSN'08. International Conference on*, 2008.

[36] Bhaskar Krishnamachari, Deborah Estrin, and Stephen Wicker. The impact of data aggregation in wireless sensor networks. In *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*, pages 575–578. IEEE, 2002.

[37] S Madden. Intel berkeley research lab data, 2003.

[38] HM Künzel, A Holm, D Zirkelbach, and AN Karagiozis. Simulation of indoor temperature and humidity conditions including hygrothermal interactions with the building envelope. *Solar Energy*, 78(4):554–561, 2005.

[39] Memsic Crossbow. Telosb v2 data sheet. Downloaded from www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf, 2008.

[40] Mehmet C Vuran, Özgür B Akan, and Ian F Akyildiz. Spatio-temporal correlation: theory and applications for wireless sensor networks. *Computer Networks*, 45(3):245–259, 2004.

[41] Mehmet C Vuran and Ian F Akyildiz. Spatial correlation-based collaborative medium access control in wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 14(2):316–329, 2006.

[42] Yong Wang, Pei Zhang, Ting Liu, Chris Sadler, and Margaret Martonosi. CRAWDAD dataset princeton/zebranet (v. 2007-02-14). Downloaded from http://crawdad.org/princeton/zebranet/20070214, February 2007.

[43] Mohit Jain, Ajeet Pal Singh, Soshant Bali, and Sanjit Kaul. CRAWDAD dataset jiit/accelerometer (v. 2012-11-03), November 2012.

[44] Richard M. Fujimoto, Randall Guensler, Michael P. Hunter, Hao Wu, Mahesh Palekar, Jaesup Lee, and Joonho Ko. CRAWDAD dataset gatech/vehicular (v. 2006-03-15). Downloaded from http://crawdad.org/gatech/vehicular/20060315, March 2006.

[45] Nelson I Dopico, Carlos Gil-Soriano, Iñigo Arrazola, and Santiago Zazo. Analysis of ieee 802.15. 4 throughput in beaconless mode on micaz under tinyos 2. In *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*, pages 1–5. IEEE, 2010.

[46] TelosB Datasheet. Crossbow Inc. Downloaded from http://www.memsic.com/userfiles/files/Datasheets/WSN, 2013.

III. EFFICIENT GEOSPATIAL DATA COLLECTION IN IOT NETWORKS FOR MOBILE EDGE COMPUTING

Xiaofei Cao and Sanjay Madria

ABSTRACT

The Mobile Edge Computing (MEC) paradigm changes the role of edge devices from data producers and service requesters to data consumers and processors. MEC mitigates the bandwidth constraint between the edge server and the cloud by directly processing the large data created by the sheer volume of IoT devices in the edge locally. An efficient data-gathering scheme is crucial for providing quality of service (QoS) within MEC. In this paper, we proposed an efficient data collection scheme that only gathers the necessary data from IoT devices like wireless sensors along a trajectory for local services based on geospatial constraints. We only use a vector of the minimal distance of hops (DV-Hop) to the anchor nodes selected by the fog server, instead of using GPS data. The proposed scheme includes a lossy compression algorithm that could compress each routing message, thus reducing the response time. In this paper, the experiments are conducted to evaluate the performance of our data collection using the encoded trajectory routing scheme compared with others using a TOSSIM simulator, and also using the powerTOSSIM-Z with real sensor motes. Our scheme performs better in terms of latency, reliability, coverage, and energy usage compared to other state-of-the-art schemes.

1. INTRODUCTION

The Internet of Things (IoT) facilitate fast access, process, and utilization of the big data created by the 'things' surrounding many applications such as disaster management or battlefield monitoring. The number of IoT devices and the data rate and size produced by

this large number of things are increasing faster than wireless bandwidth in IoT networks. Further, the applications which rely on real-time data delivery cannot accept delays caused by the routing and the bandwidth bottleneck between the edge of the IoT network and the cloud. For example, in a battlefield, soldiers patrolling a border area need to receive real-time sensing data from RF scanners about IEDs from the locations of interest, so any delay in receiving such data may endanger their lives.

In recent years, researchers have turned their focus on edge computing [1] and fog computing [2]. By collecting, caching, and exploiting sensing data locally, and interacting with local wireless sensor and actuator networks (WSAN) directly, the edge/fog provides services with higher reliability. However, data collection at the edge with reduced latency is still a challenge in mobile edge networks (MEN). For some location-aware applications, location anonymity is crucial for safety of users. For example, an adversary could easily infer users' locations like home address and working place and predict their mobility based on the history of using the location-aware services [3]. Thus, the challenge is to collect data by preserving the location anonymity in MEN.

In MEN, sensors and other IoT devices in WSAN contribute to the most volume of sensing data. Broadcasting is widely used as the fastest way to disseminate real-time data requests to the whole network. The counter-based broadcasting scheme [4] and their adaptive versions [5][6][7] are proposed to minimize the redundant rebroadcasting to save energy and mitigate the broadcast storm effects [4]. However, even the state-of-the-art adaptions of the counter-based broadcasting cannot reduce more than 60% rebroadcasting.

Some trajectory-based routing protocols, which route packets through wireless sensor nodes that reside more or less on the designed trajectory, have the potential to fetch the data from specific areas with the minimum overhead of redundant forwarding. However, most of the trajectory routing protocols like [8] and [9] require all the sensor nodes to have the GPS to decode the encoded routing trajectory, which is not practical for low-cost WSAN. Although the cubic Bezier curve used in [8] provides a good compression ratio for

the position of interest (POI), it still cannot be adapted in WSAN without GPS modules. A virtual coordinate system is an option for IoT WSAN without GPS. It can use local connectivity information such as the number of neighbors of each node and the perimeter nodes' locations as in [10]. It can also use the anchor nodes and the vector of minimum hop distance (DV-Hop) to the anchor nodes to estimate the distance between nodes. However, the state-of-the-art DV-Hop based location estimation [11] requires lots of memory resources and computational power which is not suitable for low power wireless sensor networks.

To address the shortcomings of the existing works, we propose a spatial data collection scheme that has both low latency and less overhead of redundant broadcasting. Instead of using the exact nodes' location information from GPS as in [8][9][12], we only use a vector of the minimal distance of hops (DV-Hops) to all the anchor nodes selected by the secure fog server. The area of position of interest (POI) can be represented as a list of hop constraints to the anchor nodes. Our routing message only contains two basic geometric shapes: hyperbola and arc. These shapes can be represented with two simple mathematical equations. The sensor nodes could avoid the complex geometric computing, which makes it suitable for WSAN that have low-power and low-computing resources.

In addition, the proposed scheme provides location anonymity by avoiding using and transmission of the GPS location information. To decode the POI of the client, the adversary has to have the encoded message as well as the location of the anchor nodes, which are stored in the secure fog server. We also address the broadcast storm issue by integrating the counter-based broadcasting mechanism.

Our performance evaluation shows that the proposed scheme compresses the data about 8 times and reduces more than half of the latency in data requesting and collection process compared to directly broadcasting the list of node identifications which reside within the POI. The reliability of the proposed scheme also beats state-of-the-art geospatial routing

protocols like ring routing and nested routing [13][14], which route messages in a circular trajectory. The energy consumption of data requests within our scheme is also reduced compared to the state-of-the-art counter-based broadcasting schemes.

2. RELATED WORKS

2.1. COUNTER-BASED BROADCASTING

Broadcasting is the fastest way to flood a message into the whole WSAN. However, limited bandwidth causes delay in broadcasting a sequence of messages into the network. After a node receives a given packet, the counter-based broadcasting scheme [4][5][6][7] requires the node to wait for a short period of time to listen to its neighbors and count how many times the given packet has been rebroadcast. If the broadcast count of the given packet reaches the predefined threshold, it will drop the packet. Thus, only few of the nodes in the network will rebroadcast the given packet which saves bandwidth and thus, alleviates the congestion.

2.2. RING ROUTING AND NESTED ROUTING

Ring routing and nested routing [13][14] are proposed to solve the problem of routing packets to a mobile sink. The idea behind ring routing and nested routing is to store the current mobile sink's location in a ring or nested ring structure. The data source needs to query the nodes in the ring/nested ring structure to fetch the updated location of the mobile sink before routing the packets. Then the data source routes the packets to the mobile sink using the updated sink location. Ring routing and nested routing achieve good delay and energy performance because searching the ring structure is easier than searching the whole network.

2.3. TRAJECTORY-BASED ROUTING AND VIRTUAL COORDINATES

Trajectory based routing [8][9] is a paradigm that only the nodes near the given routing trajectory will forward the packets. It includes trajectory generating and encoding and the routing decision rules for each sensor node. It has the following challenges: First, the trajectory encoding algorithm should able to compress the trajectory as the encoded message will be included in the routing packets. Second, each compressed message should be able to route through the trajectory to the sink reliably. Third, the overhead of routing caused by redundant rebroadcast should be minimized. However, for WSAN, the additional challenge is to route through a trajectory without using any GPS-based location information.

In order to route through a trajectory with virtual coordinates using DV-Hop rather than GPS, the virtual coordinates should be able to reflect the sensors' real location precisely. Intuitively, increasing the number of anchor nodes will improve the precision of the virtual coordinates. This has also been proven by DV-Hop based localization algorithms such as in [15], [16], and [17]. The challenge, however, is to reduce the computation and memory usage which are limited in sensor nodes. The naive combination of greedily checking the distance to the routing trajectory and the use of the virtual coordinate system with many anchor nodes requires computational resources, and is also error prone due to the use of estimated location. According to the DV-Hop based localization algorithms, in the worst case, the error rate can be as large as 45% of the range of the radio [11], which could lead to a routing failure.

3. PROPOSED DATA COLLECTION SCHEME

The proposed data collection scheme enables the fog server to directly collect only the necessary data for the edge clients from nearby IoT networks by sending a data request message. The data request packets should be able to reach from mobile edge devices like cellphones, computers, and routers that are near the client's position of interest (POI) to the

target IoT devices (usually the wireless sensor motes) with minimum overhead and latency. In heterogeneous IoT networks, different wireless devices with different radio standards cannot directly communicate with each other. We first introduce the system overview where we use a cellphone as a gateway of a wireless sensor network. Then, we present a novel trajectory encoding algorithm that could not only compresses the location information to reduce the system response time and latency, but also preserve the location anonymity which is essential for a location-based edge-computing application.

3.1. SYSTEM OVERVIEW

The location-based applications usually need real-time response and current and consistent data. The conventional approaches that collect all the sensor data periodically and process the clients' requests in the cloud data center not only have poor QoS due to high latency and data distortion but also suffer from bandwidth bottleneck due to the huge number of IoT devices at the edge. The proposed data collection scheme is based on edge-computing paradigm where the edge devices consume data with geospatial constraints specified by the client. It uses mobile edge devices as the gateway that coordinate with wireless sensor networks and downstream fog servers. The data collection tasks are offloaded to the local edge nodes near the POI and thus reduces the latency. The data processing tasks are offloaded to the local fog server.

The system overview is shown in Figure 1. The cloud data centers only collect, process, and store the time-insensitive data. The fog servers collect, process, and generate the compressed geospatial constraints of user requests which will be discussed in the next section. Then, the data collection task is offloaded to the mobile edge devices near the POI. With the help of the serial port listening and writing app [18], the edge device becomes a gateway of IoT sensor network by connecting a sensor mote to its serial port.

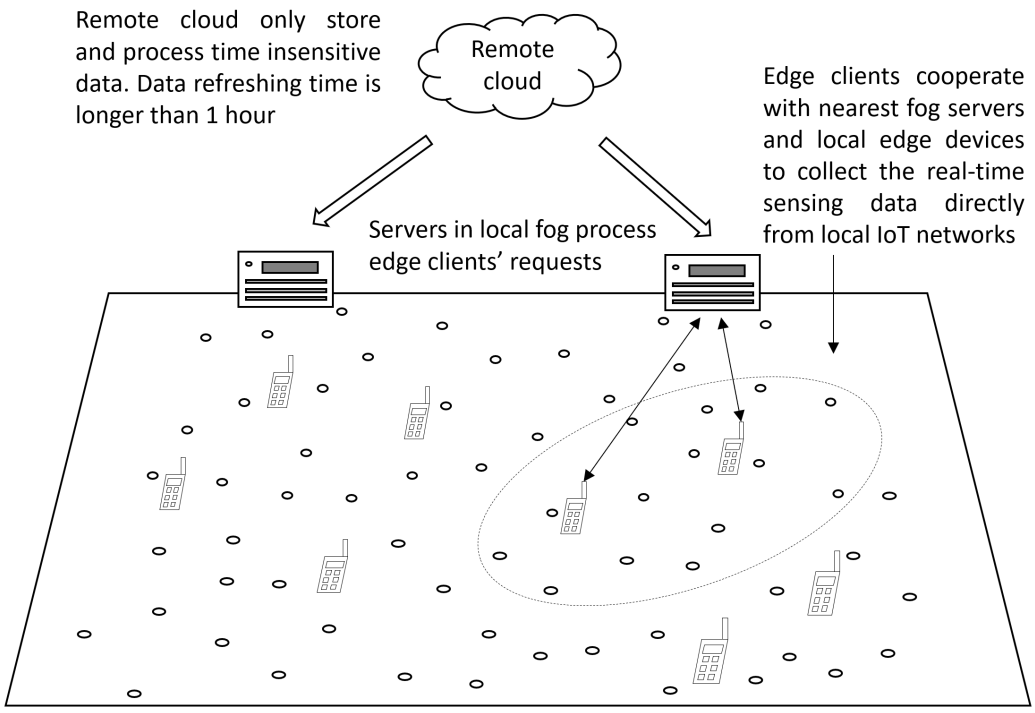


Figure 1. Data collection in local edge network

3.2. ASSUMPTIONS

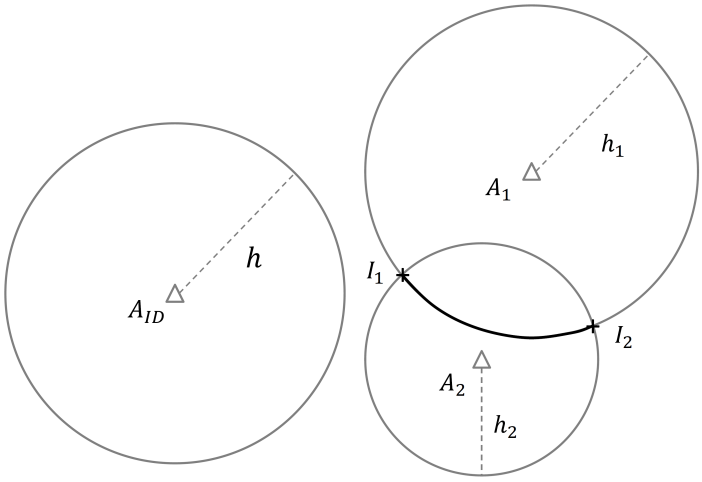
In the IoT network, many applications prefer to have the geospatial information tagged with the sensing data. For example, firefighters want to know the fire status based on the temperature and infrared sensors' data tagged with the location information. As discussed in the Introduction, in location-based IoT networks, the wireless sensor nodes are the most populated 'things' and generate most of the data. Due to their low cost and energy limitation, most of the sensor motes do not contain GPS module themselves. To enable these sensors provide geospatial information without GPS, we use a vector of the minimal distance of hops to the anchor nodes (DV-Hop) as the virtual coordinates. The anchor nodes, which are selected by the fog server, know their locations. They have a one-byte ID number, which is enough for a local fog network. We assume that mobile edge devices try to collect data from a WSAN directly. We also assume that the fog server, which encodes the geospatial request, has the location information of the local anchor nodes and

has enough computing resources. In the WSAN, near each sensor node, there are at least three randomly deployed anchor nodes which will flood their identification to the others in the area within a limited number of hops (H_{max}) from them. While flooding, all the sensor nodes will create and update a vector of minimal distance of hop (DV-Hop) to their nearby (within H_{max} hops) anchor nodes. The anchor nodes also have the DV-Hop of their nearby (within H_{max} hops) anchor nodes and will eventually transmit this information to the fog server. After the network has been initialized, if any nodes move, they need to update their DV-Hop by querying all their new neighbors' DV-Hop. For the new DV-Hop of the moving nodes, the new DV-Hop is set to be the minimal hop count of all the new neighbors' DV-Hops to the respective anchor nodes plus one.

The trajectory, which includes the routing path and POI, can have the shape of any type of continuous curve. It may have different widths in different segments, and can overlap with itself. However, we assume that the trajectory is unidirectional, so the overlapping trajectory is seen as one curve and the intersection trajectory is seen as a branch. To avoid looping, we assume that every node in the trajectory only re-broadcast the same routing packet once. A routing packet should be fewer than 127 bytes (the limit of IEEE 802.15.4 packet size).

3.3. DV-HOP BASED GEOSPATIAL ENCODING ALGORITHM

The idea of our encoding algorithm is to use a set of geometry shapes to represent position of interest (POI) and the trajectory from a gateway to the POI. With the assumption as in 3.1, the trajectory and POI drawn by the clients can be seen as a set of discrete pixels in a 2d Euclidean space. Each pixel has two parameters: the x- and y-coordinates from the predefined origin point. The unit of the coordinate, μ, is chosen based on the application requirement. So each 1 μ by 1 μ area in this WSAN is a pixel that can be represented with a tuple (x_{coor}, y_{coor}). We call this set of pixel the Trajectory Area Set (TAS).



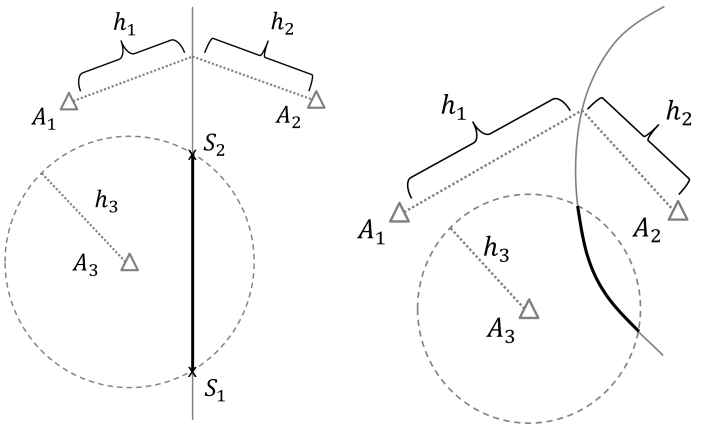
(a) A circular trajectory (b) An arc trajectory

Figure 2. Example of a circular, and an arc trajectory represented with hop constraints

As the fog server knows the location and the DV-Hop of each anchor node, we can calculate the average hop distance d_{avg} in μ unit between each pair of anchor nodes. Then, we can use hop constraints to represent the following simple shapes. For example, as shown in Figure 2-a, a circular area with width d_{avg} can be represented with a center anchor node ID, A_{ID}, and the circle radius in number of hops h. With two anchor nodes, we can represent the arc by defining each anchor node's (A_1, A_2) hop distance (h_1, h_2) to its circle and finding the two intersection points (I_1, I_2) between these two circles as shown in Figure 2-b.

Another example in Figure 3-a shows the shortest path from node S_1 to node S_2 is a straight line. Using anchor nodes A_1 and A_2, we can define a hyperbola $h_1 - h_2 = 0$ which is a line passing through S_1 and S_2. Then, we need a third anchor node A_3 with hop h_3 to cover both S_1 and S_2. Finally, we obtain a segment of line that starts and ends from S_1 and S_2. Here, the anchor nodes A_1, A_2 determine the line's direction, and A_3's location and hop constraints determine the line's starting and ending points. The line is a special case

of constraints $h_1 - h_2 = 2 \times a, (a \in Integer)$ that represent an arc of a hyperbola shown in Figure 3-b. With the anchor node A_3 and its hop count h_3, we can obtain a segment of the hyperbola.



(a) A segment of line trajectory (b) A segment of hyperbola trajectory

Figure 3. Example of a segment of line, and a hyperbola trajectory represented with hop constraints

Intuitively, the line can be used to connect two nodes with the shortest path length. The arc can be used to connect two lines with different directions. Any trajectory can be seen as the assembly of these two shapes. As a straight line is a special case of a hyperbola, we use hyperbola to approximate the line. If there is no anchor pair that could create this line, we can still use a segment of hyperbola to approximate it.

The computation complexity is another challenge in encoding a geospatial area using shapes. For a WSAN with anchor nodes N_{Anchor} and hops limitation H_{max} for each anchor node, the shapes (hyperbola and arc) which constitute the trajectory are chosen from $N_{shapes} = N_{Anchor}^2 H_{max}^2 + N_{Anchor}^3 H_{max}^2$ different possible shape constraints. Testing of all the combinations of the shapes has $O(N_{shapes}!)$ time complexity, which is not practical.

Thus, we propose a greedy algorithm which considers both the number of newly covered pixels and the effective coverage ratio (ECR). ECR, which is defined in Equation 1, is the ratio of the overlapping area of TAS and a shape over the area of the shape.

$$ECR = \frac{Area_{covered}}{Area_{shape}} \tag{1}$$

The trajectory encoding algorithm is divided into the following two steps:

3.3.1. Find All Possible Shapes from N_{shapes} and Their Area $Area_{shape}$. We define the possible shapes as the shapes that could constitute a portion of the trajectory. First of all, we calculate the minimal number of hops h_{start} from each anchor node to the nearest pixel in the TAS, and the minimal number of hops h_{end} required by each anchor node to reach the farthest pixels in the TAS.

Thus, to find all possible arcs, we use two layers nested for loops to iterate through the anchor nodes. Within each layer, we iterate respective anchor node's hops from h_{start} till h_{end}. Within the inner anchor loop, we calculate the average one-hop length (d_{avg}) for each pair of anchor nodes. So each possible arc can be represented as follows:

$$C_{arc} = [A_1, A_2, h_1, h_2, d_{avg}] \tag{2}$$

As shown in Figure 2-b, the arc is from point I_1 to point I_2 with width d_{avg} which is the average one-hop distance from inner circle with hop count $h_1 - 1$ to the outer circle with hop count h_1. We use $Area_o$ to represent the overlapping area between the outer circle of A_1 and the control circle of A_2 and use $Area_i$ to represent the overlapping area between the inner circle of A_1 and the control circle of A_2. Thus, the area of the arc $Area_{arc}$ meets $Area_{arc} = Area_o - Area_i$.

There are four cases exits when calculating the overlapping area of two circles. When two circles are separated, the overlapping area is zero. When circle A_2 is inside circle of A_1, the overlapping area is the area of A_2. When circle of A_1 is inside circle of A_2, the

overlapping area is the area of circle A_1. For the last case that two circles intersect with each other, the overlapping area can be calculated using Equation 3. Algorithm 1 shows the procedure for calculating the area of a segment of the arc trajectory.

Algorithm 1: Get the area of the arc

Result: Area of arc: $Area_{arc}$

Input : $arcCircle$, $controlCircle$, d_{avg}

1 $innerCircle = new\ Circle(arcCircle, d_{avg})$;

2 $Area_{outerCircle}$ = CALL Algorithm(2):
$circlesIntersectArea(arcCircle, controlCircle)$;

3 $Area_{innerCircle}$ = CALL Algorithm(2):
$circlesIntersectArea(innerCircle, controlCircle)$;

4 return $Area_{arc} = Area_{outerCircle} - Area_{innerCircle}$;

A hyperbola can be defined with two anchor nodes which are its foci. A control circle is defined with one anchor node which is its center. To find all possible hyperbola segments, we use three layers nested for loop to iterate through the anchor nodes. For each shape, we iterate hops from h_{start} to h_{end}. We calculate both the average one-hop distance between the foci of the hyperbola as d_{focus} and the average one-hop distance between all three anchor nodes as d_{avg}. So each possible hyperbola can be represented as shown in Equation 4.

$$\begin{aligned} Area = {} & r_2^2 \times acos((d^2 + r_2^2 - r_1^2)/(2 \times d \times r_2)) \\ & + r_1^2 \times acos((d^2 + r_1^2 - r_2^2)/(2 \times d \times r_1)) \\ & - 0.5 \times sqrt((-d + r_2 + r_1) \times (d + r_2 - r_1) \\ & \times (d - r_2 + r_1) \times (d + r_1 + r_2)) \end{aligned} \tag{3}$$

$$C_{hyper} = [A_1, A_2, A_3, a, h_3, d_{focus}, d_{avg}] \tag{4}$$

Algorithm 2: Get the area of two intersecting circles

Result: Intersect area of two circles

Input : $circle_1, circle_2$

```
1  d = getCenterDistance(circle_1, circle_2);
2  if d >= circle_1.radius + circle_2.radius then
3  |   return 0; //case (1)
4  else if circle_1.radius >= circle_2.radius + d then
5  |   return π × (circle_2.radius)^2; //case (2)
6  else if circle_2.radius >= circle_1.radius + d then
7  |   return π × (circle_1.radius)^2; //case (3)
8  else
9  |   return run Equation (3) ; //case (4)
10 end
```

For example, in Figure (4), assume that the x-coordinate of the anchor nodes is X_{ID} where "ID" is the anchor node's identification number and the y-coordinate of the anchor nodes is Y_{ID}. Assume that the Euclidean distance between A_1 and A_2 is $2 \times c$. We define the overlapping area of the hyperbola of A_1 and A_2 and control circle of A_3 is $Area_{hyper} = Area_{innerHyper} - Area_{hyperbola}$. The inner hyperbola meets $Dis_{A1} - Dis_{A2} = 2 \times a \times d_{focus}$. Here $Dis_{A_{ID}}$ is the distance between any point in the hyperbola to the focus with given 'ID'; 'a' is a positive integer less than H_{max}.

Algorithm 3 and 4 show the procedure of calculating the area of a segment of the intersection of a hyperbola and a circle. First, the hyperbola is rotated and shifted to standard format, which obeys Equation (5) by multiplying the hyperbola with the transformation matrix (6) and subtracting a shift vector shown in Equation 7. Then, multiplying the circle

with center A_3 with the same transformation matrix 6, and shift using vector 7, which gives us the new center shown in Equation (8). With the transformed circle center, we can get the circular function as in Equation (9).

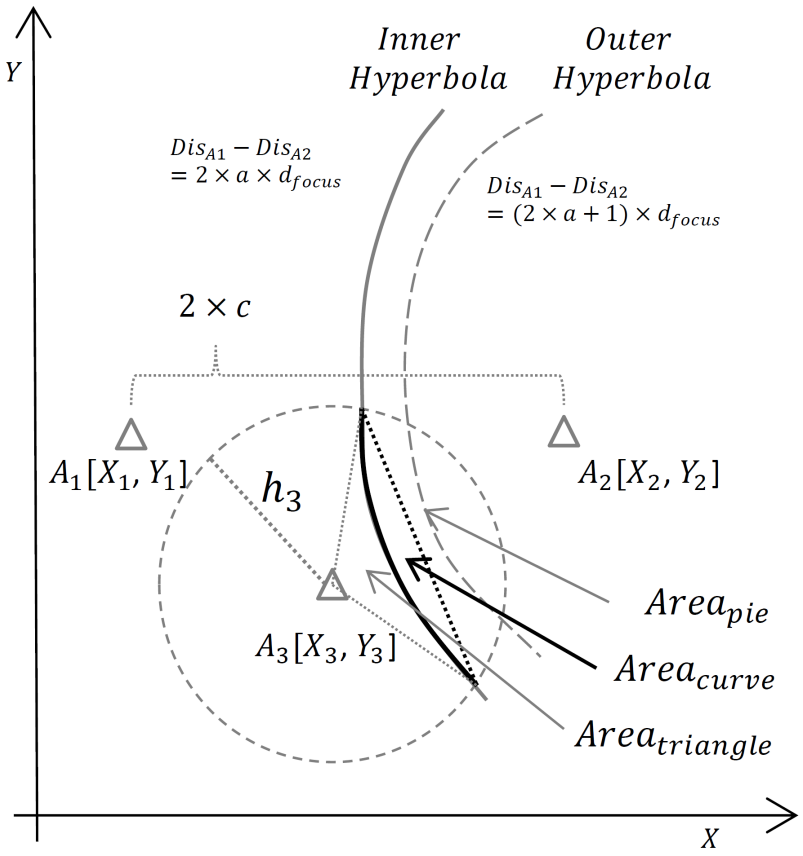


Figure 4. A hyperbola segment's area estimation example

$$\frac{x^2}{a^2} - \frac{y^2}{(c-a)^2} = 1 \tag{5}$$

$$T = \begin{bmatrix} \frac{c\times(X_2-X_1)}{2\times((X_2-X_1)^2+(Y_2-Y_1)^2)} & \frac{c\times(Y_1-Y_2)}{2\times((X_2-X_1)^2+(Y_2-Y_1)^2)} \\ \frac{c\times(Y_2-Y_1)}{2\times((X_2-X_1)^2+(Y_2-Y_1)^2)} & \frac{c\times(X_2-X_1)}{2\times((X_2-X_1)^2+(Y_2-Y_1)^2)} \end{bmatrix} \tag{6}$$

$$V = \begin{bmatrix} (X_1 + X_2)/2 \\ (Y_1 + Y_2)/2 \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} X_{cycle} \\ Y_{cycle} \end{bmatrix} = T \cdot (\begin{bmatrix} X_3 \\ Y_3 \end{bmatrix} + V) \tag{8}$$

$$(x - X_{cycle})^2 + (y - Y_{cycle})^2 = (h_3 \times d_{avg})^2 \tag{9}$$

Algorithm 3: Get the area of hyperbola segment

Result: Area of hyperbola segment: $Area_{hyper}$

Input : $hyperbola, controlCircle, d_{avg}$

1 $innerHyper = new\ Hyperbola(hyperbola, d_{avg})$;

2 $Area_{hyperbola}$ = CALL Algorithm(4):
$hyperIntersectCircle(hyperbola, controlCircle)$;

3 $Area_{innerHyper}$ = CALL Algorithm(4):
$hyperIntersectCircle(innerHyper, controlCircle)$;

4 **return** $Area_{hyper} = Area_{innerHyper} - Area_{hyperbola}$;

The Equation 5 and 9 can be merged into a quartic-equation. Then, we can approximate the intersection points of the hyperbola and the circle by solving the quartic-equation using Newton's method [19], and sort the points list, $List_p$, based on the y-coordinate values. After finding the intersection points, the overlapping area of a hyperbola and the control circle can be calculated as follows: $Area_{hyper} = Area_{pie} + Area_{triangle} - Area_{curve}$.

As shown in Figure 4, the $Area_{curve}$ is the area between the line connected the two intersection points and the hyperbola curve between the two intersection points. We can calculate this area by solving the definite integral of the hyperbola function minus the line function. This area is the overlapping area of the pie area and the triangle area. So it need to be deducted from the sum of the pie area and triangle area to get the final intersection area.

Algorithm 4: Get the area of hyperbola intersect circle

Result: Area of hyperbola intersect circle

Input : $hyperbola\ h, circle\ c$

$standardize(\&h, \&c)$; //Eq:(5)-(9);

$coefficients = quarticCoefficient(h, c)$;

$roots = getRealRootsQuarticEq(coefficients)$;

$List_p = findIntersectionPoint(roots, h, c)$;

for $i \leftarrow 0$ **to** $size(List_p)$ **by** 2 **do**

  $\phi = intersectAngle(List_p[i], List_p[i+1])$;

  if $\sin\phi < 0$ **then**

    $Area_{pie} = \arccos(\cos\phi) * r^2/2$;

  else

    $Area_{pie} = (2 * pi - \arccos(\cos\phi)) * r^2/2$;

  end

  $Area_{triangle} = \sin\phi \times r^2/2$;

  $Area_{curve} = FindCurveArea(h, List_p[i], List_p[i+1])$;

  $Area_{hyper} += Area_{pie} + Area_{triangle} - Area_{curve}$;

end

return $Area_{hyper}$;

3.3.2. Compute the Effective Coverage Ratio (ECR) and Elect the Best Shapes. To calculate the $Area_{covered}$, a brute force method is used by testing hop constraints pixel by pixel. Different from the first step which uses a lot of condition branches, the second step has few branches. Thus, we are able to accelerate the computation with GPU computing. We designed a NVIDIA CUDA kernel that calculates the ECR as follows:

For the arc, if the distance from any pixel in the TAS to A_1 (Dis_{A_1}) obeys $Dis_{A_1} = h_1 \times d_{avg}$ and the distance from any pixel in the TAS to $A_2(Dis_{A_2})$ obeys $Dis_{A_2} \leq h_2 \times d_{avg}$, then that pixel is covered by the arc shape. For the hyperbola, if the distance from any pixel in the TAS to A_1 (Dis_{A_1}) and to A_2 (Dis_{A_2}) obeys $Dis_{A_1} - DisA_2 = 2 \times a \times d_{focus}$ and the distance from any pixel in the TAS to $A_3(Dis_{A_3})$ obeys $Dis_{A_3} \leq h_3 \times d_{avg}$, then that pixel is covered by the hyperbola shape [note that here all the notations are the same as in Equations (2) and (4) and Figure 4].

We consider both the total number of pixels in TAS covered by the element shapes, and the Effective Coverage Ratio (ECR). In each iteration of our greedy algorithm for finding the best shape, we always elect the shape which provides the maximum value of the Greedy Factor (GF) which is the number of pixels multiplied by cubic ECR as shown in Equation (10):

$$GF = Area_{covered} \times ECR^3 \tag{10}$$

After each iteration, we first eliminate the pixels of TAS, which also exists in the best shape in 3.5.1. Then, we calculate the newly covered pixels for each possible shape with the updated TAS. But instead of calculating a new ECR, the ECR is reused to speed up the calculation. The shape with the maximum GF are chosen as the best shape. The pixels covered by the best shape are deducted from the TAS.

Last, we repeat this procedure until the size of updated TAS is less than $1 - Th$ of the original size, where Th is the predefined coverage threshold and will be discussed in Figure 11. If the trajectory encoding message is exceed the packet size limitation, which is shown in Table 1, we will divide the POI into two and create two separate trajectories with two different gateway nodes, which is shown in Figure 1. Then the algorithm will encode them separately and send two request packets in sequence. The second packet will refer to the ID of the first packet and able to rebroadcast in the trajectory the first packet indicates.

3.4. PAYLOAD DATA STRUCTURE OF ROUTING PACKET

The message structure of TinyOS has a 11-byte header that includes the sender's address. We also exploit the type and the group data in the header. The payload structure, defined in Table 1, is used for our implementation of DV-TE-R and adapted DV-TE-BR which will be discussed in Section 4.

3.5. ROUTING DECISION FOR WIRELESS SENSORS

For the WSAN where sensor nodes never sleep, we propose the counter-based routing decision to mitigate the broadcast storm effect [4]. In the proposed counter-based routing decision, each wireless sensor uses two kinds of criteria to decide if it should forward the routing packet or not. First is to check if it meets any constraint of the encoded trajectory ($flag_C$). Second is to check if the counter used to count the number of the nearby redundant re-broadcasting for the same packet reaches the threshold ($flag_T$).

As we have discussed in 3.3, the arc constraint has a size of four bytes: A_1, A_2, h_1, h_2. For any sensor node, we set its $flag_C$ to be true if its DV-Hop entry of A_1 and A_2 obeys Equation (11). The hyperbola constraint has a size of five bytes: A_1, A_2, A_3, a, h_3. For the same sensor node, we set its $flag_C$ to be true if its DV-Hop entry of A_1, A_2, A_3 satisfies Equation (12).

$$\begin{cases} DV_{Hop}[A_1] == h_1 \;||\; DV_{Hop}[A_1] == h_1 - 1 \\ DV_{Hop}[A_2] \leq h_2 \end{cases} \tag{11}$$

$$\begin{cases} floor(DV_{Hop}[A_1] - DV_{Hop}[A_2])/2 == a \\ DV_{Hop}[A_3] \leq h_3 \end{cases} \tag{12}$$

The counter-based routing decision is first proposed in [4]. Once a sensor node broadcasts a packet to its neighbors, the neighbors will be listening to the channel for a random amount of time before it actually forwards the packet. During the listening period,

the sensors will count the number of times the same packet has been forwarded. If it exceeds the counter threshold, it will set the $flag_T$ to be false so only the sensors with $flag_C == true$ and $flag_T == true$ will forward the routing packets. Thus, only a few of the nodes in the network will rebroadcast the given packet which saves bandwidth and thus, alleviates the congestion.

Table 1. Payload data structure

Descriptions	Starting Bytes	Length in Bytes
Message ID	0	4
Parent Node address	4	2
Hop counts	6	1
1st group size	7	1
relaxation parameter	8	1
constraints for relax	9	5
routing constraints	14	100

Another case is for low-power listening WSAN where sensors hibernate for most of their lifetime. The sensors will rebroadcast immediately if they find they satisfy Equation (11) and Equation (12) ($flag_C == true$) and initialize a counter with value 0. Then, they will stop broadcasting when there is a timeout or when their rebroadcasting neighbors' number reaches the counter threshold ($flag_T == true$). Note, the rebroadcaster will update their packets based on how many neighbors has broadcast. In this way, only the counter number of neighbors will rebroadcast the packets which saves energy and bandwidth. Thus, only a few of the nodes in the network will rebroadcast the given packet which saves bandwidth and thus, alleviates the congestion. Further, the broadcast could reduce the routing delay which mitigates the long latency of LPL.

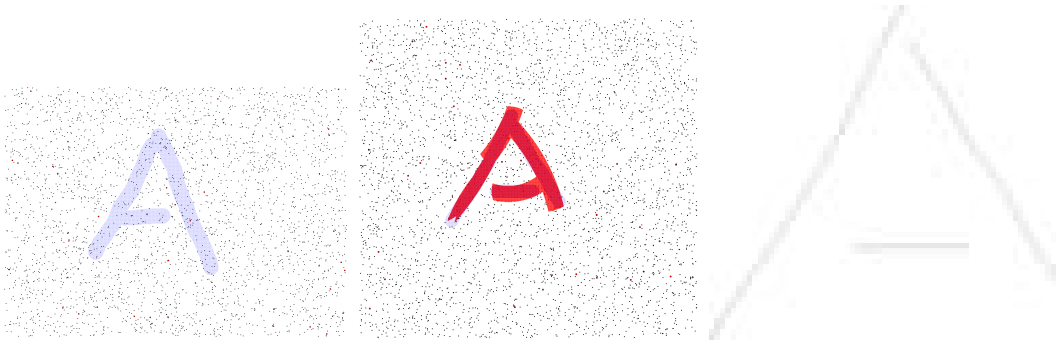
3.6. SAMPLE ROUTING RESULT AND ANALYSIS

The proposed geospatial area encoding algorithm works for any shape and trajectory. Here we test the encoding algorithm on some sample trajectories shown in Figure 5-(a),(d),(g). Figure 5-(a) is a trajectory of hand written 'a'. Figure 5-(d) is an outline trajectory of Breuer park in Rolla, Missouri, USA. Figure 5-(g) is a trajectory of automatically generated nested rings. For each of the routing trajectories, assume anchor node ID is one byte long and the number of hops is one byte long and the length of the encoded trajectory is 23, 54, and 89 bytes as shown in Table (2). The red shapes, which represent the encoded trajectory, shown in Figure 5-(b),(e),(h) are the cascaded arc and hyperbola shapes. We also compress the minimum area of the hand drawing trajectory using JPEG format, which is a lossy compression algorithm for images. The resolution of the output of JPEG is set to be 64×64.

For time complexity analysis, assume that there are 'n' anchor nodes in local edge network, each anchor node floods at most 'r' hops, and the TAS has 'm' entries. Then the time complexity of finding the best arc shape is $O(mn^2r^2)$, the time complexity of finding the best hyperbola is $O(mn^3r^3)$.

Table 2. Encoding result for sample trajectory

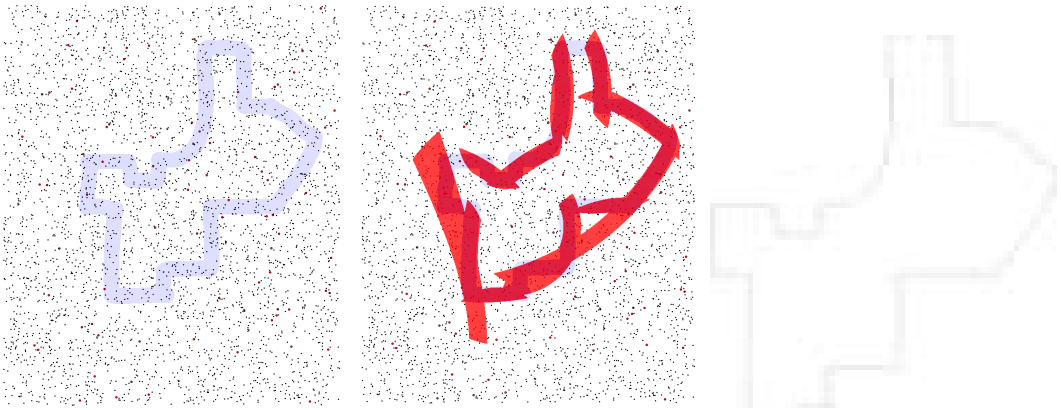
Trajectory type	A	outline	Circles
Number of arcs	3	12	17
Number of hyperbolas	2	1	4
Message length(byte)	23	54	89
Compressed size with JPEG(byte)	867	887	934



(a) Hand drawing 'A'

(b) Encoding trajectory 'A'

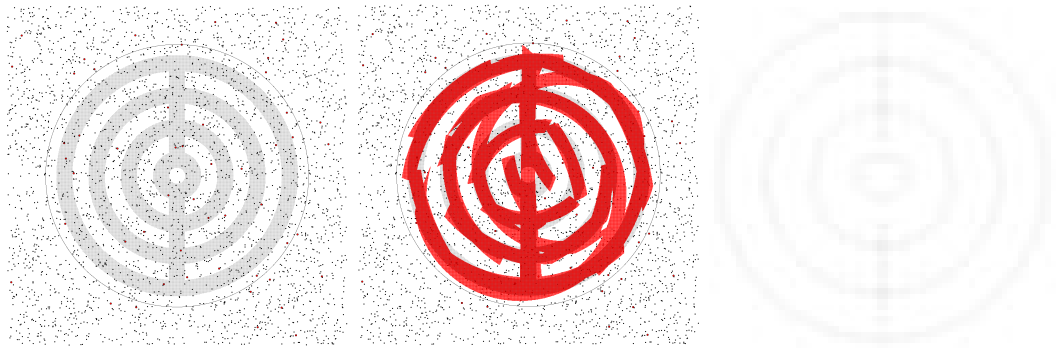
(c) JPEG compressed hand drawing A



(d) Hand drawing boundary of Breuer Park, Rolla, MO

(e) Encoding trajectory of Breuer Park, Rolla, MO

(f) JPEG compressed park boundary



(g) Auto drawing 'rings'

(h) Encoding trajectory 'rings'

(i) JPEG compressed nested rings

Figure 5. Example of a circular, and an arc trajectory represented with hop constraints and the compressed trajectory using JPEG

4. ADAPTED DV-HOP BASED DATA COLLECTION SCHEME FOR LOW-POWER WSAN

To ensure high QoS in WSAN, we adapt the proposed DV-Hop based trajectory encoding and routing protocol (DV-TE-R). For low-power WSAN, which can be deployed in the harsh environment, the density of the network topology can be heterogeneous. Somewhere in the region, the sensors may be sparsely deployed, or the routing path could be obstructed by some 'holes' shown in Figure 6, where we plan to forward a packet from node S_i to S_o through an arc with center A and radius h hops with 1 hop width. Although the DV-hop of both the nodes S_i and S_o is h, these two nodes are not directly connected because of an obstacle between them. If using local broadcasting, for example in Figure 6, to fix the routing path, we at least need to flood 4 hops which is a huge overhead. We propose a bridge on the edge adaption (DV-TE-BR) that could connect a broken routing path with minimum overhead.

After a forwarder node has forwarded the routing packet and has not overheard any rebroadcast from its neighboring nodes nor acknowledgement from the sinks, it will start iterating its valid constraints, relax those by one hop, note all the changes and rebroadcast the packet again. If it receives the rebroadcast from its neighbors, it will stop iterating and go to sleep immediately. If a node receives the relaxed-constraint packet, it will tighten the constraint by one, and repeat the previous procedure until recover the original constraint. Note that both h_1 and h_2 are relaxed for arc constraint, and both a and h_3 are relaxed for hyperbola constraint.

For example, like Figure 6, the nodes B_1, B_2, B_3, B_4 have an increased DV-hop entry of A from $h + 1$ to $h + 2$. Thus, the sensor node S_i needs to relax the hop constraints by one. Then, the node B_1 needs to relax the hop constraint by one more. For B_2, it will hold the constraints as of B_1. The constraint is tighten by one for B_3, and so does B_4. Finally, the route is fixed after B_4 forwards the packet to S_o.

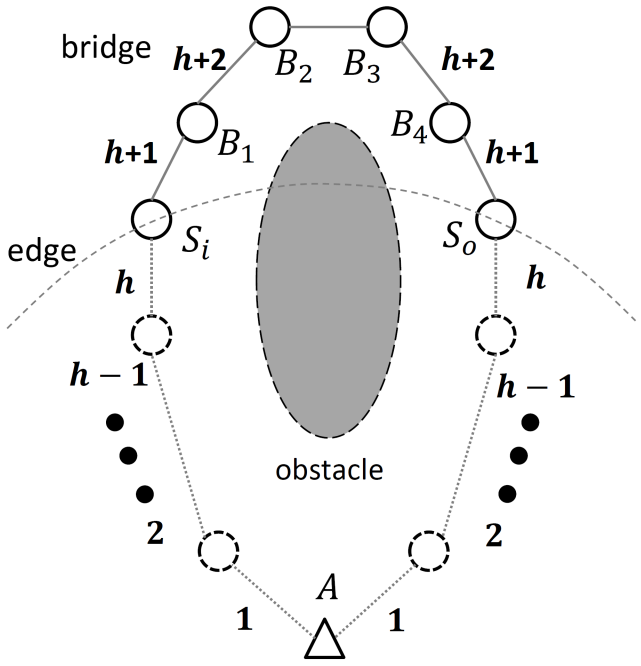


Figure 6. Example of the bridge on the edge adaption

5. EXPERIMENTS AND RESULTS

We set-up our experiments using the parameters in Table 3. The trajectory and POI encoding is executed using a desktop, which acts as a fog server, with a Xeon E5-1620 v2 and a Nvidia RTX 2070 GPU. We assume that the POI is located within a WSAN distributed in a 1800m by 1700m area. The density of the WSAN is less than 0.6%. To save energy, all of the wireless sensors work under low power listening mode (LPL), where each sensor node only wakes up for a few milliseconds to listen to the channel. The WSAN is simulated using TOSSIM. We also use powerTOSSIM-Z to estimate the energy consumption of the activated sensor nodes. There are 30 local edge nodes randomly deployed in the WSAN field. The local edge nodes act as the gateway that will broadcast the encoded data requests' packets and collect the data from the WSAN. The performance metric used includes the compression ratio, reliability, average delay in data reporting, and energy consumption.

Table 3. Parameters for the experiments

Area of deployment	1800×1700 m
Number of sensor nodes	5000
Communication range	100 m
Number of edge devices	30
Broadcasting hop limitation	30 hops
LPL sleeping time	600ms
LPL wake time	10ms
Energy model	MicaZ
Coverage threshold	90%
Number of anchor nodes	(20 - 100)
Anchor cover range	20 hops

To measure the compression ratio of the proposed trajectory encoding algorithm, we use real-world taxi-trajectory data [20]. Each line of the trajectory data contains the trajectory of a taxi trip, in the city of Porto in Portugal, represented as a list of 8 byte GPS data (latitude and longitude). The 2 GB data-set contains trajectories of different shapes and lengths. We scale the taxi trajectory to the center of the sensing field. The width of the trajectory is set as two thirds of the wireless radio range. In Figure 7, the X-axis is the number of GPS data points in the trajectory, and the Y-axis is the size of the set of node IDs in the trajectory. The figure shows that trajectory with more GPS data will cover more sensor nodes. In Figure 9, where the X-axis is the number of GPS data in a trajectory and the Y-axis is the compression ratio, 500 trajectories were sampled, the number of GPS data points ranging from [2, 20],[21, 40],[41, 60],[61, 80], and [81, 105] and the number of anchor nodes in the set [20, 40, 60, 80, 100] which is (0.4%, 0.8%, 1.2%, 1.6%, 2%) of the total number of nodes in the network. The compression ratio (CR) is defined as

the uncompressed-data size over the compressed data size using the proposed encoding algorithm. Here the uncompressed data size is the size of a list of sensor IDs that reside in the trajectory. The results show that when the number of anchor nodes increase, the CR also increases. It is more likely to find the enough closest 'shapes' when the number of anchor nodes is large, which costs fewer iterations to cover the area. We notice that when the number of GPS data points increases, the CR is not increased when encoding with 20 or 40 anchor nodes. A possible reason is that when there are fewer anchor nodes, redundant shapes may be selected in the late iterations, which increases the size of the encoded message. For example, in Figure 5(e)(h), there are more shapes that overlap with other shapes than that in the case of Figure 5(b).

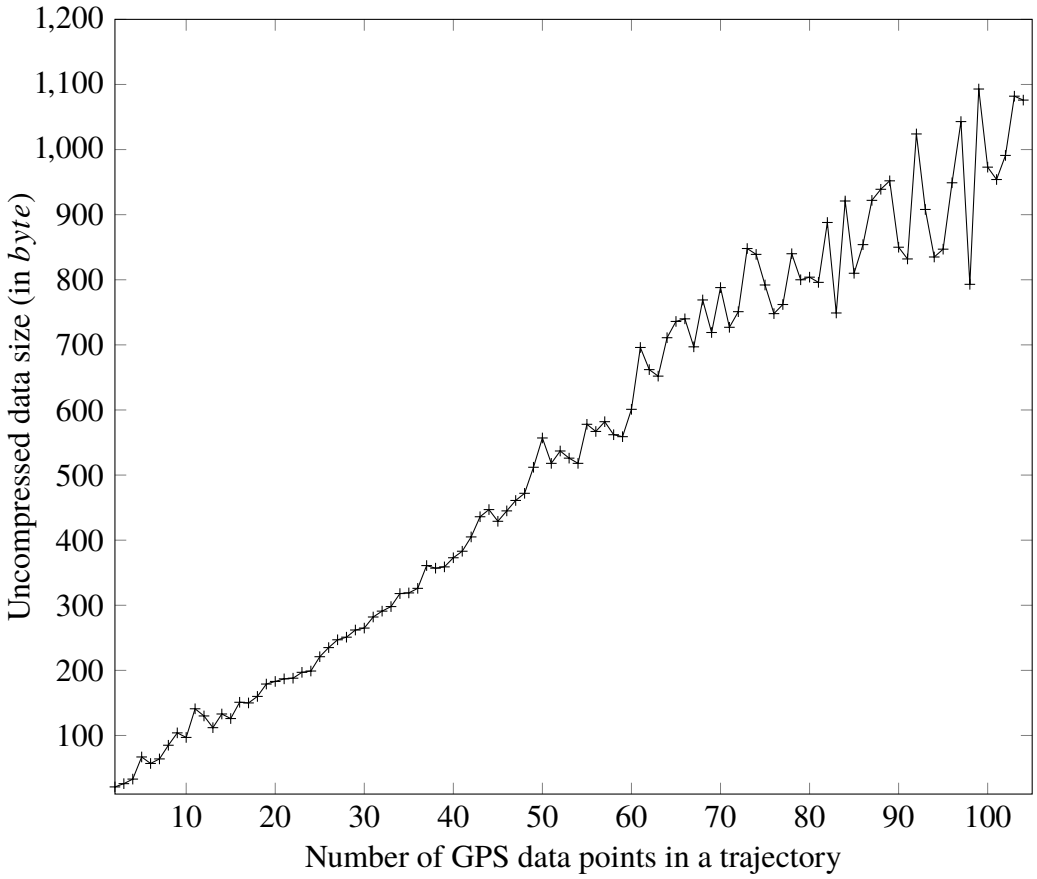


Figure 7. Property of taxi trajectory dataset

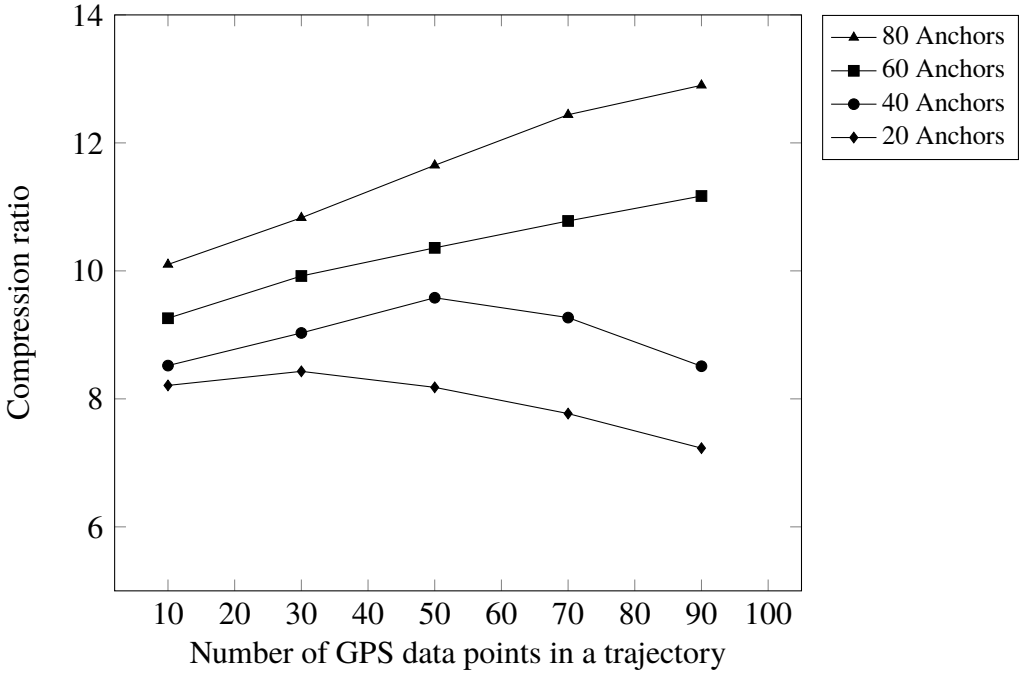


Figure 8. The compression ratio of the proposed DV-Hop based trajectory encoding algorithm for different trajectory sizes with different number of anchor nodes

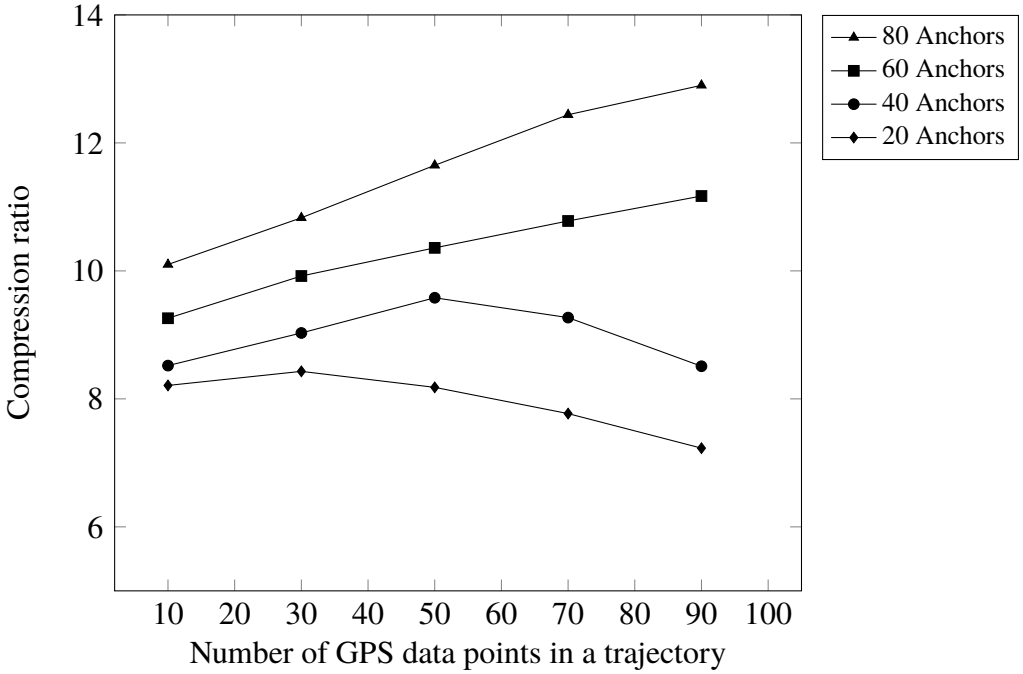


Figure 9. The compression ratio of the proposed DV-Hop based trajectory encoding algorithm for different trajectory sizes with different number of anchor nodes

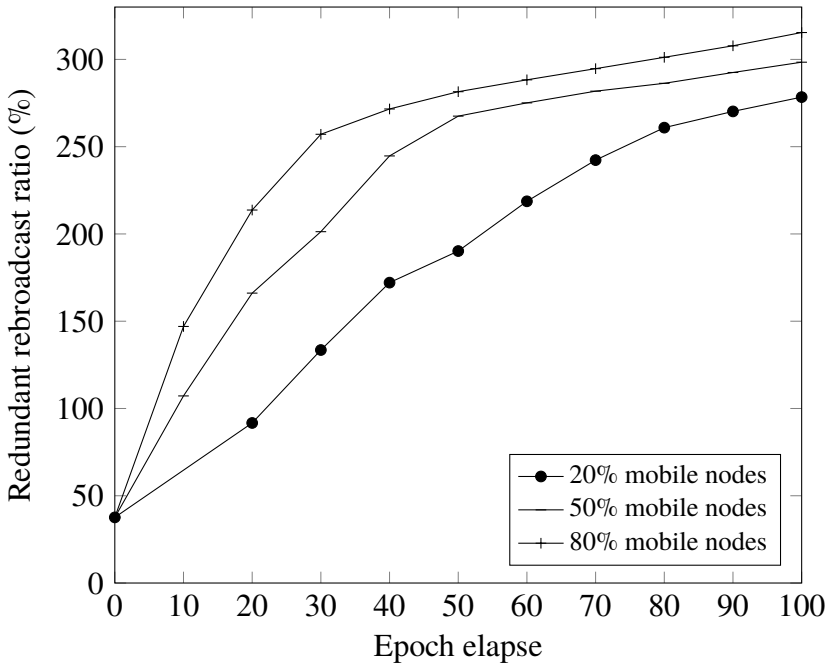


Figure 10. Average redundant rebroadcast ratio

Since the proposed trajectory encoding algorithm uses basic shapes (arc and hyperbola segments) to approximate the trajectory, there is a possibility that the encoding will fail when no suitable shape is found. Here, we define an encoding failure as when the encoding algorithm cannot find any combination of shapes that will cover a certain threshold ratio of TAS, which is defined as the predefined coverage ratio. For example, when the threshold is set to 85%, the hops constraints represented shapes must overlap with more than 85% of the trajectory area. The lower the threshold, the higher is the encoding successful rate. However a threshold lower than 85% is not recommended, as the routing reliability (the rate of successfully routing the data request message to the POI) will be affected due to the uncovered gaps between the routing paths. In this experiment, we find the relationship between the number of anchor nodes, success rate (the percentage of encoding that do not fail), and coverage threshold. The result in Figure 11 shows that as the number of anchor nodes increasing, the successful rate of encoding will increase. However, increasing the coverage threshold will decrease the successful rate of encoding.

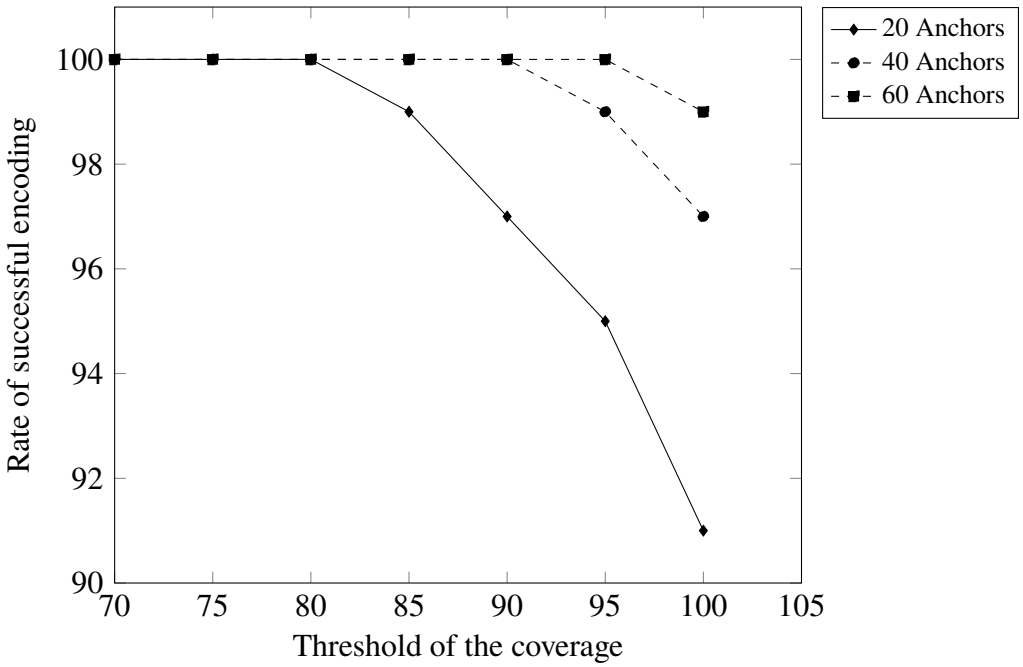


Figure 11. Experiment of successful encoding rate with different number of Anchor nodes and coverage threshold

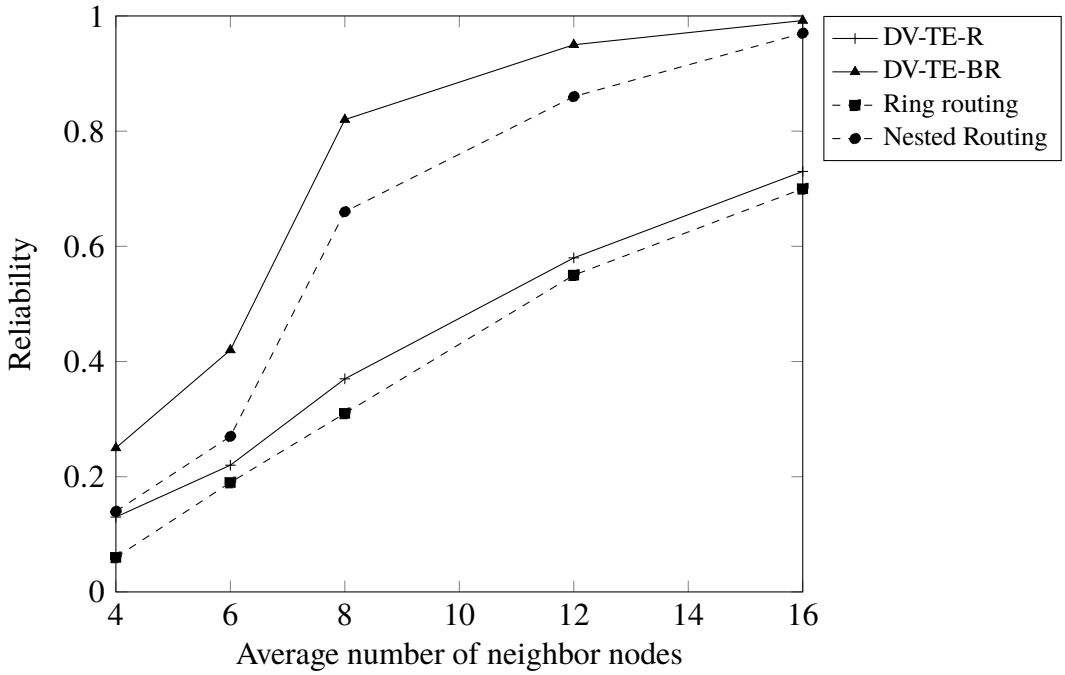


Figure 12. The reliability of DV-TE-R, DV-TE-BR, Ring routing, and Nested routing with different number of neighbors

We compare the reliability of the proposed encoding and routing protocol with the ring routing [13] and the nested routing [14], which both enable data routing to a moving sink (mobile edge device that can move in some applications) by relaying the data to a circular area where the nodes know the updated location of the mobile edge device. The area of a nested ring is in the middle of the sensing field, as shown in Figure 5(g). In this experiment, we use 50 anchor nodes for the proposed algorithm. We define the reliability of ring routing as the success rate of generating a ring structure.

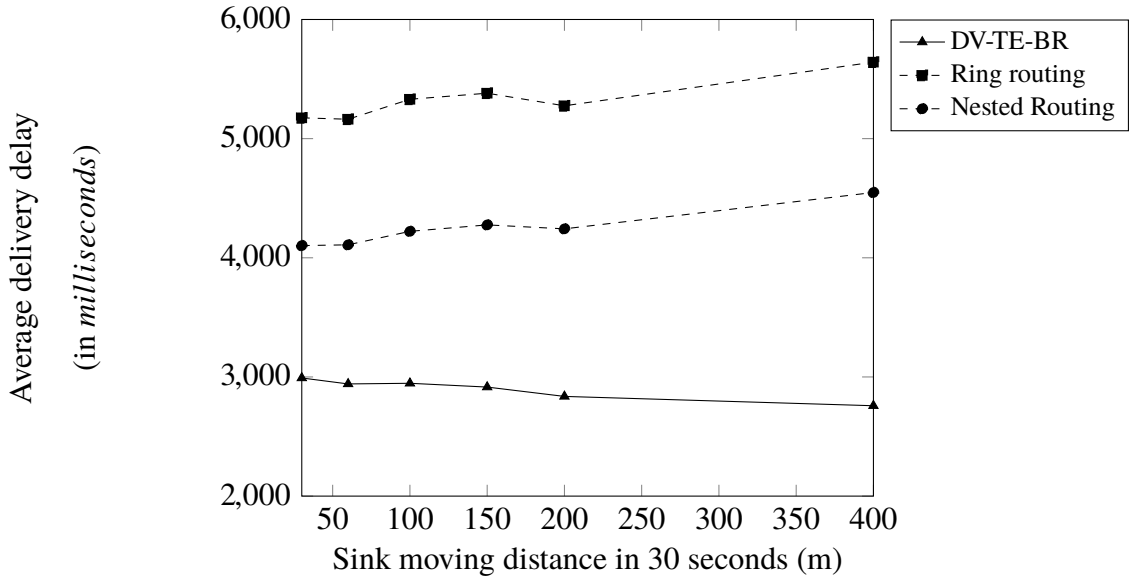


Figure 13. The average delay of data reporting comparing with state-of-art schemes

The reliability of nested routing is the success rate of generating any one of the ring structures within its nested ring structure. The reliability of the proposed DV-Hop based trajectory encoding and routing (DV-TE-R) and its bridge on the edge (DV-TE-BR) adaption is the success rate of generating the encoded message of the trajectory and routing the data request packets to the nodes within POI. The result is shown in Figure 12. When the average number of neighbors of each node is smaller than 8, which is 0.16% of the total number of nodes, both protocols have low reliability. The nested routing protocol has

better reliability performance than ring routing because it has redundant rings. The bridge on the edge adaption improves the reliability of DV-TE-R by relaxing the hop constraints. When the average number of neighboring nodes is greater than 16, which is 0.32% of the total number of nodes, the reliability of DV-TE-BR is greater than 99%. It achieves the best reliability performance compared to ring routing and nested ring routing. In the rest of the experiments, by default, we use DV-TE-BR.

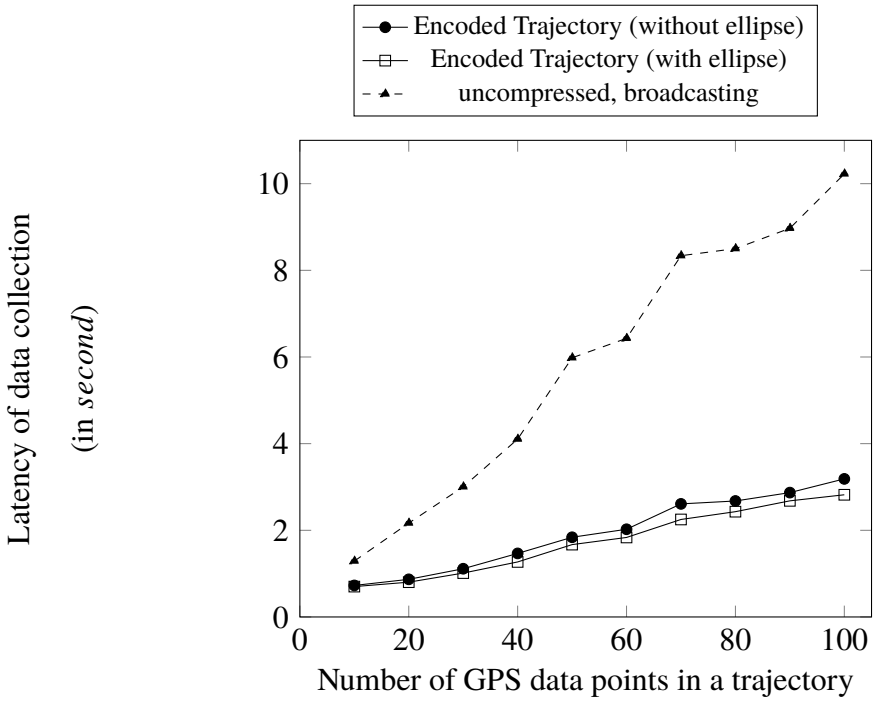
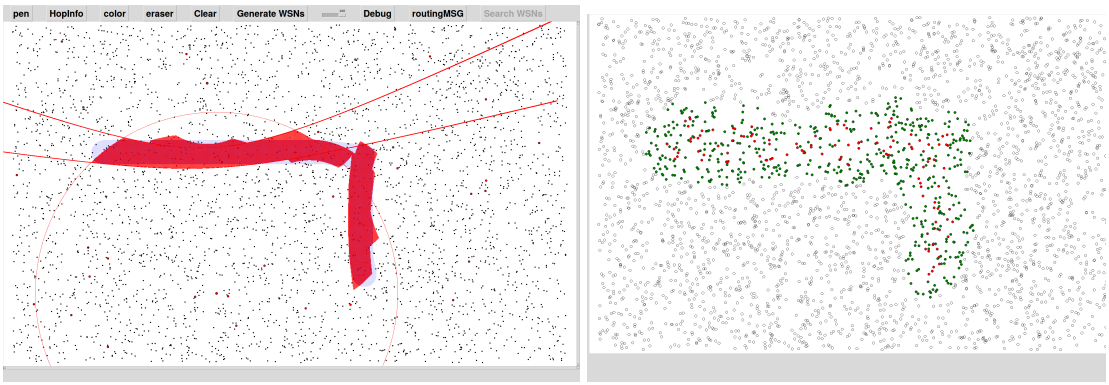


Figure 14. Average delay from starting broadcast request till receiving all the data from the POI

In the following average data reporting delay experiment, we assume that mobile edge devices will move randomly with different speeds in the local IoT network with the configuration, as shown in Table 3. The average delay in data reporting is the time when the moving mobile edge devices receive the data minus the time when the source reports the data. The proposed DV-TE-BR fixes the current routing path by letting the mobile edge

devices update their locations periodically. Thus, it cannot always guarantee the shortest reporting path as the ring routing and nested routing methods do. Although the ring routing and nested routing provide the current location of the mobile sink, fetching this information from the ring or nested ring for the source node causes the delay overhead of one round trip to the closest ring. Thus, the delays of ring routing and nested ring routing are still higher than the delay of DV-TE-BR. In addition, the counter-based routing strategy of DV-TE-BR reduces the waiting delay for the low-power listening WSN because the first awaked node could start routing, while ring routing and nested routing have to wait for specific routing nodes within its routing table. Nested routing has a better delay performance than ring routing because its average shortest distance from the source to the rings is shorter than the ring routing. The delay performance of data reporting is shown in Figure 13.



(a) Encoded trajectory (b) Visualized routing result

Figure 15. Sample routing example

In the following latency and energy consumption experiments, we compare the proposed scheme with the state-of-the-art counter-based broadcasting algorithm. The experimental set-up, which uses the taxi trajectory data, is shown in Table 3. We use TOSSIM to simulate the routing of data request messages and data reporting packets and visualize the results using Python. Figure 15(a) shows a sample trajectory being encoded using hops

constraints represented shapes, and Figure 15(b) shows a sample output of the visualized routing result where the red dots are the sensors that forwarded a message and the green dots are sensors that received a message.

The latency in collecting the requested data is an important factor in meeting the quality of service. Broadcasting is the fastest way to flood the data request into the whole network. The proposed data collection scheme also broadcasts the data request to the POI. However, as opposed to flooding approaches [6], only the nodes in a hop constraints defined trajectory can rebroadcast. Thus, the energy consumption and bandwidth usage are minimized.

Figure 14 compares the latency of the local edge devices receiving all the sensing data of the POI from the local WSAN, which are working under low power listening (LPL) mode with a 660 ms sleeping and waking period. We assume that all local edge devices will broadcast the data request messages. The sensor nodes that receive the data request packets will be awake and send the data back to the nearest edge devices if they are at the POI. The experimental result shows that the proposed data collection scheme could reduce latency because broadcasting a compressed message requires fewer packets than an uncompressed message which includes IDs of all the sensor nodes residing within the POI.

The energy consumption experiment is simulated with powerTOSSIM-Z, which is an energy simulation tool for wireless sensors. It uses the micaZ energy model and can measure the energy consumption at the packet level. The result in Figure 16 shows that the proposed data collection scheme consumes less energy than the broadcasting approach. We next compare the average number of rebroadcasting nodes of the proposed DV-hop based trajectory encoding and routing scheme (DV-TE-BR) verses the state-of-the-art counter-based broadcasting [6] for each single data request packet. The result shows that the proposed DV-TE-BR scheme reduces the number of redundant rebroadcasting packets (142 vs. 2491) by 94% and thus, saves bandwidth usage in the WSAN.

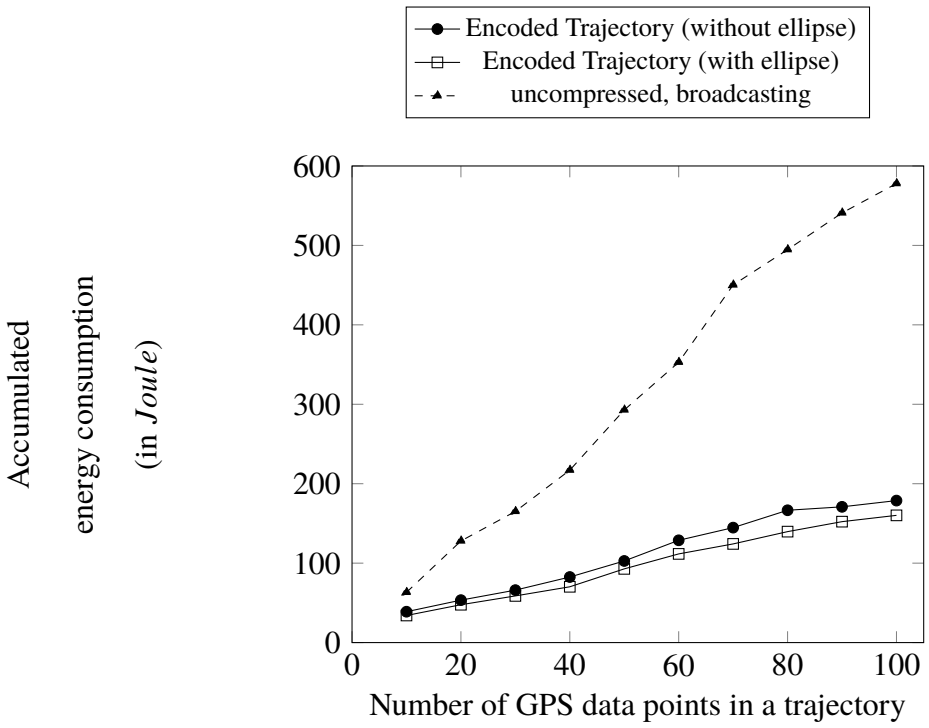


Figure 16. Accumulated energy consumption in fetching the data from the POI

6. CONCLUSION AND FUTURE WORK

The proposed trajectory encoding and data collection algorithm for IoT applications has improved energy efficiency, reduced latency, and achieved reliable performance when fetching data from the POI in the local fog network without using GPS. In addition, with the use of virtual coordinates, location anonymity is achieved for the source, sink, and intermediate nodes in the routing path, as only the secure server in the local fog knows the anchor nodes' locations. In the future, we plan to integrate the Z-compression algorithm [21] to further compress the trajectory message and sensing data of the IoT devices. The link of the current implementation provided as a web service is here http://www.routing-demos.com:8080/.

REFERENCES

[1] Weisong Shi and Schahram Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

[2] Shanhe Yi, Zijiang Hao, Zhengrui Qin, and Qun Li. Fog computing: Platform and applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*. IEEE, 2015.

[3] Yingzi Wang, Nicholas Jing Yuan, Defu Lian, Linli Xu, Xing Xie, Enhong Chen, and Yong Rui. Regularity and conformity: Location prediction using heterogeneous mobility data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1275–1284. ACM, 2015.

[4] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless networks*, 8(2-3):153–167, 2002.

[5] Chien Chen, Chin-Kai Hsu, and Hsien-Kang Wang. A distance-aware counter-based broadcast scheme for wireless ad hoc networks. In *Military Communications Conference, MILCOM*. IEEE, 2005.

[6] Ji-Young Jung and Dong-Yoon Seo. Counter-based broadcast scheme considering reachability, network density, and energy efficiency for wireless sensor networks. *Sensors*, 18(1):120, 2018.

[7] Kok-Poh Ng, Charalampos Tsimenidis, and Wai Lok Woo. C-sync: Counter-based synchronization for duty-cycled wireless sensor networks. *Ad Hoc Networks*, 61:51–64, 2017.

[8] Murat Yuksel, Ritesh Pradhan, and Shivkumar Kalyanaraman. An implementation framework for trajectory-based routing in ad hoc networks. *Ad Hoc Networks*, 4(1):125–137, 2006.

[9] Houda Labiod, Nedal Ababneh, and Miguel García de la Fuente. An efficient scalable trajectory based forwarding scheme for vanets. In *Advanced Information Networking and Applications (AINA), 24th IEEE International Conference on*, pages 600–606. IEEE, 2010.

[10] Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 96–108. ACM, 2003.

[11] Shrawan Kumar and DK Lobiyal. Novel dv-hop localization algorithm for wireless sensor networks. *Telecommunication Systems*, 2017.

[12] Badri Nath and Dragoş Niculescu. Routing on a curve. *ACM SIGCOMM Computer Communication Review*, 33(1):155–160, 2003.

[13] Can Tunca, Sinan Isik, Mehmet Yunus Donmez, and Cem Ersoy. Ring routing: An energy-efficient routing protocol for wireless sensor networks with a mobile sink. *IEEE Transactions on Mobile Computing*, 14(9):1947–1960, 2015.

[14] Ramin Yarinezhad. Reducing delay and prolonging the lifetime of wireless sensor network using efficient routing protocol based on mobile sink and virtual infrastructure. *Ad Hoc Networks*, 84, 2019.

[15] Hongyang Chen, Kaoru Sezaki, and Ping Deng. An improved dv-hop localization algorithm with reduced node location error for wireless sensor networks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 91(8):2232–2236, 2008.

[16] Kai Chen, Zhong-hua Wang, Mei Lin, and Min Yu. An improved dv-hop localization algorithm for wireless sensor networks. 2010.

[17] Shrawan Kumar and DK Lobiyal. An advanced dv-hop localization algorithm for wireless sensor networks. *Wireless personal communications*, 71(2):1365–1385, 2013.

[18] Xiaofei Cao. Serial port reader and writer for the android gateway. https://github.com/cxfcdcpu/gateway.

[19] Carl T Kelley. *Solving nonlinear equations with Newton's method*, volume 1. Siam, 2003.

[20] Taxi trajectory data. www.kaggle.com/crailtap/taxi-trajectory.

[21] Xiaofei Cao, Sanjay Madria, and Takahiro Hara. Multi-model z-compression for high speed data streaming and low-power wireless sensor networks. *Distributed and Parallel Databases*, 2019.

IV. EFFICIENT DATA COLLECTION IN IOT NETWORKS USING TRAJECTORY ENCODED WITH GEOMETRIC SHAPES

Xiaofei Cao and Sanjay Madria

ABSTRACT

The Mobile Edge Computing (MEC) paradigm changes the role of edge devices from data producers and requesters to data consumers and processors. MEC mitigates the bandwidth limitation between the edge server and the remote cloud by directly processing the large amount of data locally generated by the network of the internet of things (IoT) at the edge. An efficient data-gathering scheme is crucial for providing quality of service (QoS) within MEC. To reduce redundant data transmission, this paper proposes a data collection scheme that only gathers the necessary data from IoT devices (like wireless sensors) along a trajectory. Instead of using and transmitting location information (which may leak the location anonymity), a virtual coordinate system called "distance vector of hops to anchors" (DV-Hop) is used. The proposed trajectory encoding algorithm uses ellipse and hyperbola constraints to encode the position of interest (POI) and the trajectory route to the POI. Sensors make routing decisions only based on the geometric constraints and the DV-Hop information, both of which are stored in their memory. Also, the proposed scheme can work in heterogeneous networks (with different radio ranges) where each sensor can calculate the average one-hop distance within the POI dynamically. The proposed DV-Hop updating algorithm enables the users to collect data in an IoT network with mobile nodes. The experiments show that in heterogeneous IoT networks, the proposed data collection scheme outperforms two other state-of-the-art topology-based routing protocols, called ring

routing, and nested ring. The results also show that the proposed scheme has better latency, reliability, coverage, energy usage, and provide location privacy compared to state-of-the-art schemes.

1. INTRODUCTION

The Internet of Things (IoT) facilitates fast access, process, and utilization of the big data created by the 'things' surrounding many applications such as disaster management, battlefield monitoring, and moving object tracking. However, for some IoT devices like wireless sensors, the limited energy, and high recharging cost require them to save energy as much as possible during their duty cycles. Bandwidth limitation is another challenge for IoT networks. In recent years, the number of IoT devices, the data rate, and the enormous data produced by these large numbers of things are increasing faster than the growth of wireless bandwidth. Also, for heterogeneous IoT networks with mobile nodes, there is a need of maintaining the correct and stable data collection routes without leaking the users' locations information. Last but not the least, different applications should be able to share the same IoT network efficiently. Therefore, the users of different applications can fetch data from sensors regardless of their types, precise locations, and identities.

In recent years, researchers have turned their focus on edge computing [1] and fog computing [2] to support IoT networks. The edge/fog networks interacting with local wireless sensor networks (WSNs) provide services with higher reliability in collecting, caching, and exploiting sensing data locally. Also, the latency of data collection in mobile edge networks (MEN) is reduced because edge nodes can collect and process data faster than the remote cloud. However, with the participation of third party mobile edge devices, the location anonymity problem draws some researchers' attention again. For example, consider a celebrity athlete who may try to get pollution levels along his/her running trajectory from crowd-sourcing and/or existing environmental sensor networks. An adversary could easily infer the user's locations like the hiking trail and home address, and predict the mobility

based on the history of using the location-aware services [3]. Therefore, as such this individual user would like to keep the location anonymity against the potential security risks.

In MEN, sensors and other IoT devices contribute to the most volume of sensing data. In most of the WSN applications, sensors report their sensing data periodically. However, periodically sensing has a long delay which is not desirable for applications requiring real-time low-distorted data. In recent works like [4], data sensing only happens when the sensor or mobile user receives a data request packet. Therefore, unnecessary data sensing and reporting outside of the position of interest are reduced. Another way to reduce the data delay is broadcasting, which is widely used as the fastest way to disseminate real-time data to the whole network. The drawback of broadcasting is its high energy consumption due to massive rebroadcasting. Therefore, the counter-based broadcasting scheme [5] and their adaptive versions [6][7][8] are proposed to minimize the redundant rebroadcasting to save energy and mitigate the broadcast storm effects [5]. However, even the state-of-the-art adaptions of counter-based broadcasting cannot reduce more than 60% rebroadcasting. Also, the broadcast can not control the data flow precisely which is not acceptable for some military applications like tracking enemy objects, while not being detected by the targets.

Some trajectory-based routing protocols, which route packets through wireless sensor nodes that reside more or less on the designed trajectory, have the potential to fetch the data from specific areas with the minimum overhead of redundant forwarding. However, most of the trajectory routing protocols like [9] and [10] require all the sensor nodes to have the GPS to decode the encoded routing trajectory, which is not practical for low-cost WSNs. Although the cubic Bezier curve used in [9] provides a good compression ratio for the position of interest (POI), it still cannot be adapted in WSNs without GPS modules. A virtual coordinate system is an option for IoT consisting of WSNs without GPS. It can use local connectivity information such as the number of neighbors of each node and the perimeter nodes' locations as in [11]. It can also use the anchor nodes and the vector of

minimum hop distance (DV-Hop) to the anchor nodes to estimate the distance between nodes. However, the state-of-the-art DV-Hop based location estimation [12] requires lots of memory resources and computational power, therefore, is not suitable for low power wireless sensor networks.

To address the shortcomings of the existing works, this paper proposes a spatial data collection scheme that has both low latency and less overhead of redundant broadcasting. Instead of using the exact nodes' location information from GPS as in [9][10][13], The proposed algorithm uses a vector of the minimal distance of hops (DV-Hops) to all the anchor nodes selected by the secure fog server as a dictionary or virtual coordinate. The area of the position of interest (POI) can be represented as a list of hop constraints to the anchor nodes. The routing message only contains two basic geometric shapes; hyperbola and ellipse segments in the proposed scheme. Each shape is encoded with simple hop constraints (e.g., size of the ellipse and the hyperbola and the start and end of the segment). The sensor nodes could avoid complex geometric computing, which makes it suitable for WSNs that have low-power and low-computing resources. In addition, the proposed scheme provides location anonymity by avoiding using and transmission of the GPS location information. To decode the POI of the client, the adversary has to have the encoded message as well as the location of the anchor nodes, which are stored in the secure fog server. The broadcast storm issue is addressed by integrating the counter-based broadcasting mechanism. The performance evaluation shows that the proposed scheme reduces the redundant rebroadcast in a small real-world WSN. In simulation experiments, it compresses the data about eight times and reduces more than half of the latency in the data requesting and collection process compared to directly broadcasting the list of node identifications that reside within the POI.

The reliability of the proposed scheme also beats the state-of-the-art geospatial routing protocols like ring routing and nested routing [14][15], which route messages in a circular trajectory. The energy consumption of data requests within our scheme is also reduced compared to the state-of-the-art counter-based broadcasting schemes [7][8].

The paper is organized as follows: Section 2 discusses related works about routing, broadcasting, and virtual coordinate. Section 3 first describes the system overview and assumptions. Then it elaborates the method of encoding trajectory with hyperbola and ellipse. Last, it shows some examples of encoding and routing with the encoded trajectory message. Section 4 gives an adapted data collection protocol for low-power WSN in where sensors sleep periodically. Section 7 explains the experiment setup and demonstrates the performance improvement with detailed elaborations. Section 8 concludes the paper with future work ideas.

2. RELATED WORKS

2.1. COUNTER-BASED BROADCASTING

Broadcasting is the fastest way to flood a message to cover the whole WSN. However, limited bandwidth causes a delay in broadcasting a sequence of messages into the network. After a node receives a given packet, the counter-based broadcasting schemes [5][6][7][8] require a node to wait for a short period to listen to its neighbors and count how many times the given packet has been rebroadcast. If the broadcast count of the given packet reaches the predefined threshold, it will drop the packet. Thus, only a few of the nodes in the network will rebroadcast the given packet which saves bandwidth and thus, alleviates the congestion.

2.2. GRID-BASED ROUTING

Hierarchical grid-based routing is an energy-efficient method for routing of data packets [16]. With the mobile sink and predefined virtual grid, packets could bypass the congestion area of the grid and route to the mobile sink by fetching the updated mobile sink's location from the cell-center. The grid-based routing protocol can be classified into two categories, the query-based protocol (i.e. PANEL [17], GMCAR [18], etc.) and the

event-based protocol EAGER [19]. For the query-based routing protocol, sensor nodes only sending the sensing data on request, while event-based protocol sensors will report events based on the pre-configuration of the event definition. Comparing to the event-based routing protocol where each sensor report sensing data periodically, the query-based routing protocol greatly reduces the overhead of unnecessary sensing and routing, which saves both energy and bandwidth. However, efficiently disseminating the data request packets in a WSN without GPS is a challenge for the query-based routing protocol. Some works like [16] switch alternately grid-head states to overcome the energy and bandwidth overhead of flooding control packets. The work [20] uses the location information of the cell-header and their neighbors to forward the query towards the target cell and flood message only in the target cell. Though these works reduce the broadcast overhead, the grid-based routing still needs the GPS information and extra energy to maintain the grid topology.

2.3. RING ROUTING AND NESTED ROUTING

Ring routing and nested routing [14][15] are proposed to solve the problem of routing packets to a mobile sink. The idea behind ring routing and nested routing is to store the current mobile sink's location in a ring or nested ring structure. The data source needs to query the nodes in the ring/nested ring structure to fetch the updated location of the mobile sink before routing the packets. Then the data source routes the packets to the mobile sink using the updated sink location. Ring routing and nested routing achieve good performance because searching the ring structure is easier than searching the network.

2.4. TRAJECTORY-BASED ROUTING AND VIRTUAL COORDINATE

Trajectory based routing [9][10] is a paradigm that only the nodes near the given routing trajectory will forward the packets. It includes trajectory generating and encoding and the routing decision rules for each sensor node. It has the following challenges: First,

the trajectory encoding algorithm should able to compress the trajectory as the encoded message will be included in the routing packets. Second, each compressed message should be able to route through the trajectory to the sink reliably. Third, the overhead of routing caused by redundant rebroadcast should be minimized. However, for a WSN, the additional challenge is to route through a trajectory without using any GPS-based location information.

To route through a trajectory with virtual coordinates using DV-Hop rather than GPS, the virtual coordinates should be able to reflect the sensors' real location precisely. Intuitively, increasing the number of anchor nodes will improve the precision of the virtual coordinates. This has also been proven by DV-Hop based localization algorithms such as in [21], [22], and [23]. Some previous works like [24] and [23] use the non-dominated sorting (NSGA-II) algorithm to improve positioning accuracy for complex network topologies with many anchor nodes. Their results demonstrate DV-Hop based localization algorithm could achieve good localization accuracy. The challenge in this work, however, is to reduce computation and memory usage which are limited in sensor nodes. The naive combination of greedily checking the distance to the routing trajectory and the use of the virtual coordinate system with many anchor nodes requires computational resources and is also error-prone due to the use of the estimated location. According to the DV-Hop based localization algorithms, in the worst case, the error rate can be as large as 45% of the range of the radio [12],[24] and [23]. which could lead to routing failure.

2.5. LOCATION PRIVACY IN WSN

WSNs are vulnerable to be attacked by an eavesdropper. To oppose the eavesdropper and protect the location information of the source, sink, and gateways, different approaches are proposed. To protect the source's location privacy, the paper [25] proposed a dynamic clustering algorithm and dynamic shortest path scheme that change the network topology dynamically. Thus, the adversary won't be able to locate the routing path and can't find the source by backtracking. However, the dynamic clustering algorithm consumes excessive

energy which is a drawback for many real-world applications. A more energy-efficient approach [26] proposed a hybrid source location privacy protection scheme that combined phantom source node strategy and ring routing. The phantom source node could mislead the adversary to the wrong location and the ring routing saves energy while has a good routing performance.

To protect the sink nodes' location privacy, the work [27] proposed a routing scheme that preserves the sink's location anonymity and guarantees to route within a delay threshold. The data packets route to the sink's location through several ray routes. Only one ray route is the true that goes through the sink location. The sink collects the data from random nearby intermediate nodes which store the data from the sender. However, the proposed ray routing requires the sensor nodes to maintain a table of their nearby neighbor nodes and their location information.

3. PROPOSED DATA COLLECTION SCHEME

The proposed data collection scheme enables the fog server to directly collect only the necessary data for the edge clients from nearby IoT networks by sending a data request message. The data request packet from mobile edge devices like cellphones that are near the client's position of interest (POI) should be able to reach the targeted IoT devices (usually the wireless sensor motes) with minimum overhead and latency. In heterogeneous IoT networks, different wireless devices with different radio standards cannot directly communicate with each other.

3.1. SYSTEM OVERVIEW

The proposed data collection scheme is based on an edge-computing paradigm where the edge devices consume data within the geometric constraints specified by the client. It uses mobile edge devices as the gateway that coordinate with the wireless sensor network

and downstream fog servers. The data collection tasks are offloaded to the local edge nodes near the POI, and thus reduces the latency. The data processing tasks are offloaded to the local fog server.

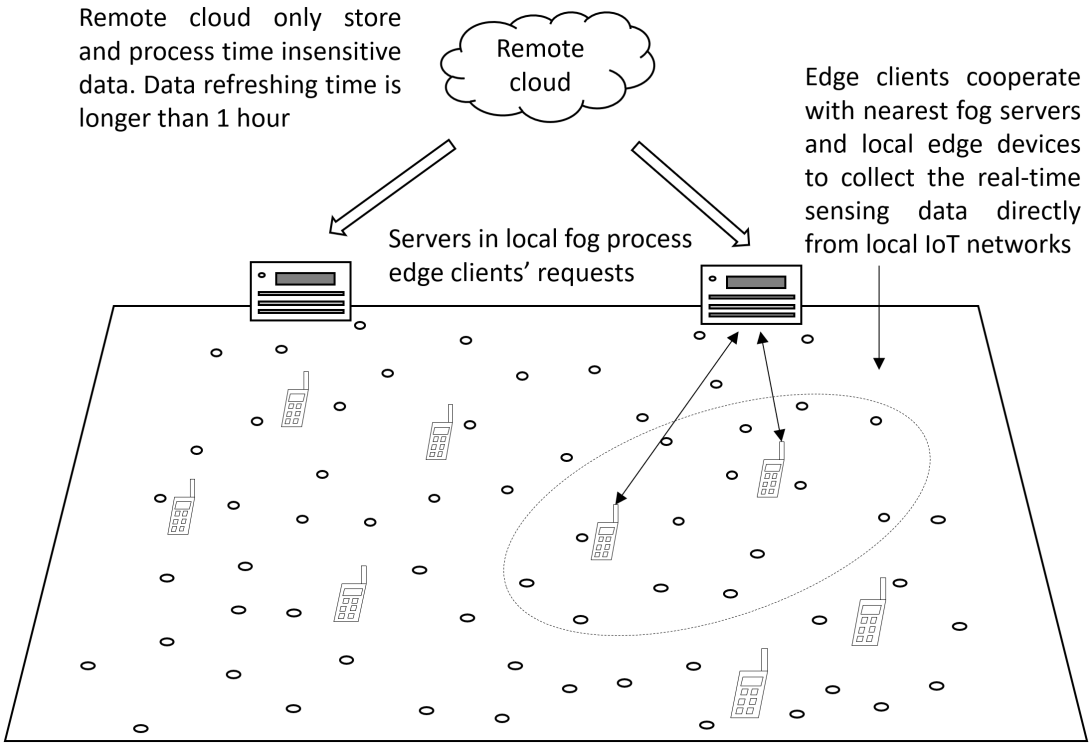


Figure 1. Data collection in local edge network

The system overview is shown in Figure 1. The cloud data centers only collect, process, and store the time-insensitive data. The fog servers collect, process, and generate the compressed geospatial constraints of user requests which will be discussed in the next section. Then, the data collection task is offloaded to the mobile edge devices near the POI. With the help of the serial port listening and writing app [28], the edge device becomes a gateway to the IoT sensor network by connecting a sensor mote to its serial port.

3.2. ASSUMPTIONS

In IoT networks, many applications prefer to have the geospatial information tagged with the sensing data. For example, firefighters want to know the chemical leak status based on the temperature and infrared sensors' data tagged with the location information. Due to their low cost and energy limitation, most of the sensor motes do not contain GPS modules themselves. To enable these sensors to provide geospatial information without GPS, the proposed algorithm uses a vector of the minimal distance of hops to the anchor nodes (DV-Hop) as the virtual coordinates.

The proposed data collection scheme has the following assumptions:

First, the anchor nodes, which are selected by the fog server, know their locations. They have a one-byte long ID number, which is enough for a local fog network. Also, the mobile edge devices, which different from anchor nodes, collect data from a WSN directly. Nodes in the WSN (maybe mobile also) may have different radio ranges and different sensing preferences.

Second, the fog server, which encodes the geospatial data request, has the location information of the local anchor nodes and has enough computing resources. In the WSN, near each sensor node, there are at least three randomly deployed anchor nodes that will flood their identification to the others in the area within a limited number of hops (H_{max}) from them. While flooding, all the sensor nodes will create and update a vector of the minimal distance of hop (DV-Hop) to their nearby (within H_{max} hops) anchor nodes.

Third, the anchor nodes also have the DV-Hop of their nearby (within H_{max} hops) anchor nodes and will eventually transmit this information to the fog server. After the network has been initialized, if any nodes move, they need to update their DV-Hop by querying all their new neighbors' DV-Hop. For the new DV-Hop of the moving nodes, the new DV-Hop entry is set to be one plus the minimum hop count entry of all the new neighbors' DV-Hops.

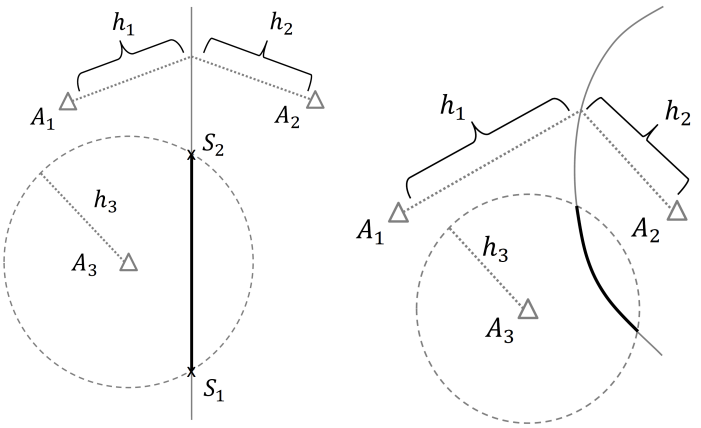
Fourth, this work assumes the user only wants to collect the necessary data where the definition of necessary data can vary based on the applications. For example, in the battlefield monitoring application, we have mentioned in the introduction, unnecessary radio broadcasting not only has the risk of being eavesdropped on but also exposes to the tactical intent. For an environmental monitoring application, we may want to monitor different locations with different frequencies. For this paper, in short, the sensor data reside in the trajectory that the user requests as the necessary data. We haven't restricted how the users should define their personalized trajectory.

Fifth, the trajectory, which includes the routing path and POI, can have the shape of any type of continuous curve. It may have different widths in different segments and can overlap with itself. The trajectory is unidirectional. So the overlapping trajectory is seen as one curve and the intersecting trajectory is seen as a branch. To avoid looping, it is assumed that every node in the trajectory re-broadcast the same routing packet only once. A routing packet size should be fewer than 127 bytes (the limit of IEEE 802.15.4 packet size).

3.3. DV-HOP BASED GEOSPATIAL ENCODING ALGORITHM

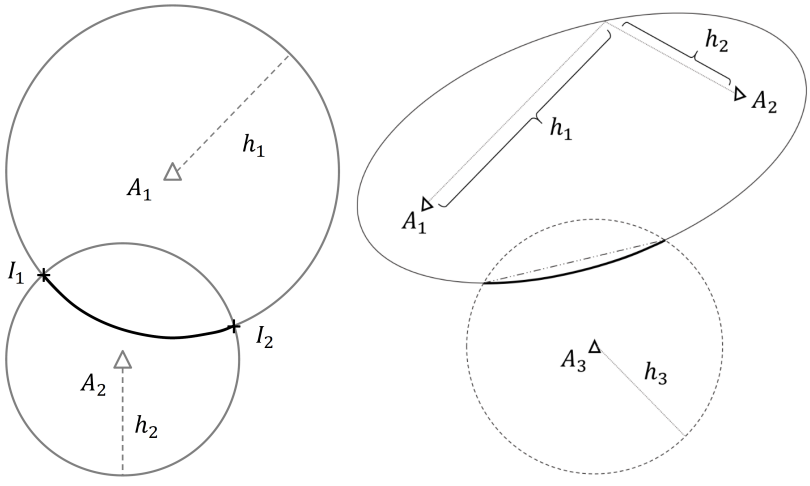
The topology of the IoT network is dynamically changing, as the network is heterogeneous and some nodes are mobile. So, the fixed routing table, which proactive routing protocol uses, is not suitable for routing and collecting data in IoT networks. Also, the reactive or cluster-based routing protocols are not efficient due to the overhead of updating the routing table or maintaining the cluster topology. They also need to be tuned for each specific application. Thus, it is hard to mix the use of different applications. The proposed trajectory encoding algorithm uses a similar idea in a software-defined network (SDN), which is to decouple the data plane (network layer) and the control plane of the IoT network. In the proposed work, the routing controllers are the fog servers shown in Figure 3.1. The routing rules are encapsulated in each routing packet. Here, the routing rules are

encoded by the controller application which resides in the control plane. Each sensor in a data plane is also seen as a mini router that decides the routing action (i.e., re-broadcast or drop packets) based on the matching rules in the routing packets.



(a) A segment of line trajectory (b) A segment of hyperbola trajectory

Figure 2. Example of a segment of line, and a hyperbola trajectory represented with hop constraints



(a) A segment of circle (b) A segment of ellipse

Figure 3. Example of a segment of circle, and a ellipse segment

To route without using the GPS data, the proposed algorithm uses DV-Hop as the virtual coordinate and store the DV-Hop table in each IoT device in the network. The controlling information in the routing packets is the geometric shape constraints discussed in the following paragraphs.

The idea of the proposed encoding algorithm is to use a set of geometric shapes to represent the position of interest (POI) and the trajectory from a gateway to the POI. With the assumption as in 3.1, the trajectory and POI drawn by the clients can be seen as a set of discrete pixels in a 2d Euclidean space. Each pixel has two parameters: the x- and y-coordinates from the predefined origin point. The unit of the coordinate, μ, is chosen based on the application requirement. So each 1 μ by 1 μ area in this WSN is a pixel that can be represented by a tuple (x_{coor}, y_{coor}) which is called the Trajectory Area Set (TAS).

Intuitively, a line can be used to connect two nodes with the shortest path length and an arc can be used to connect two lines with different directions. Then, any trajectory can be seen as the assembly of these two shapes or their more generalized form: the Hyperbola segment and an arc segment.

For example, in Figure 2-(a), the shortest path from node S_1 to node S_2 is a straight line. Using anchor nodes A_1 and A_2, a hyperbola can be defined as: $h_1 - h_2 = 0$ which is a line passing through S_1 and S_2. Then, a third anchor node A_3 with hop h_3 is required to cover both S_1 and S_2. Finally, it gives a segment of the line that starts and ends from S_1 and S_2. Here, the anchor nodes A_1, A_2 determine the line's direction, and A_3's location and hop constraints determine the line's starting and ending points. As a straight line is a special case of a hyperbola, a hyperbola can be used to approximate the line. The left arc of a hyperbola shown in Figure 2-(b) can be represented with constraints $h_1 - h_2 = 2 \times a, (a \in Integer)$. Then, with the anchor node A_3 and its hop count h_3, it generates a segment of the hyperbola. When $a = 0$, the segment is a straight line.

Another example in Figure 3 shows how to use the arc of a circle to connect two lines with different directions [29]. As a circle is a special case of an ellipse with overlapped foci, the ellipse segment could replace the circle segment.

The objective now is to use the virtual coordinate to represent these two shapes: the hyperbola segment and the ellipse arc. As the fog server knows the location and the DV-Hop of each anchor node, the average hop distance d_{avg} in μ unit between each pair of anchor nodes can be calculated. Each pair of the anchor nodes can be seen as the foci of a set of hyperbolas and a set of ellipses. The desired hyperbola and an ellipse can be chosen by setting the a value (in hop counts) of the Cartesian equation of the hyperbola and an ellipse shown in Equation (1), (2) where x_o, y_o are the coordinate of the center of the ellipse and the hyperbola. At last, keep the hyperbola and ellipse segments which overlap most of the trajectory for data collection. The segment is the intersection of the hyperbola and an ellipse with a control circle that uses an anchor node as its center and given hops count as its radius which is shown in Equation (3) where x_c, y_c are the coordinate of the center of the circle. Then, the two following simple shapes: the hyperbola segment and the ellipse segment can be encoded with the hop constraints.

$$\frac{(x - x_o)^2}{a^2} - \frac{(y - y_o)^2}{b^2} = 1 \tag{1}$$

$$\frac{(x - x_o)^2}{a^2} + \frac{(y - y_o)^2}{b^2} = 1 \tag{2}$$

$$\frac{(x - x_c)^2}{r^2} + \frac{(y - y_c)^2}{r^2} = 1 \tag{3}$$

The computation complexity is another challenge in encoding a geospatial area using shapes. As discussed, a hyperbola or an ellipse segment is determined with three different anchor nodes (two as foci and one as the center of the control circle) and two constraints of hop-distance (one as the a value, and the other as the radius of the control circle). For a WSN with anchor nodes N_{Anchor} and hops limitation H_{max} for each anchor

node, the shapes (hyperbola and ellipse) which constitute the trajectory are chosen from $N_{shapes} = N_{Anchor}^3 H_{max}^2$ different possible shape constraints. Testing of all the combinations of the shapes has $O(N_{shapes}!)$ time complexity, which is not practical. Thus, this paper proposes a greedy algorithm that considers both the number of newly covered pixels and the effective coverage ratio (ECR). ECR, defined in Equation 4, is the ratio of the overlapping area of TAS and a shape over the mathematical area of the shape.

$$ECR = \frac{Area_{covered}}{Area_{shape}} \tag{4}$$

The trajectory encoding algorithm is divided into the three steps discussed next.

3.3.1. Find All Possible Shapes from N_{shapes} and Their Area $Area_{shape}$. The possible shapes are defined as the shapes that could constitute a portion of the trajectory. In other words, the possible shapes must overlap with a portion of the data collection trajectory. The objective of this step is to generate N_{shapes} shape constraints where $N_{shapes} = N_{Anchor}^3 H_{max}^2$. Each possible shape constraint can be represented as follows:

$$C_{shape} = [A_1, A_2, A_3, a, h_3] \tag{5}$$

Here, A_1,A_2 are the foci of the shape hyperbola or the ellipse, A_3 is the center of the control circle, a is the parameter in the Cartesian Equation (1), (2), and h_3 is the radius of the control circle. The average one-hop distance d_{avg} is used to estimate the overlapping area of each shape as shown in the Appendix.

Then the algorithm uses the following filtering criteria to filter each shape constraint (hyperbola and ellipse segments) and discard shapes that don't overlap with the TAS. First, check if the control circle overlaps with the TAS. As the control circle will intercept the shape segment and finalize the shape, it is the most important criteria to determine if a shape

overlaps with the TAS. The proposed algorithm only keeps the constraints that overlap with the TAS by comparing the distance of the center of the control circle and the convex hull of the TAS which is constructed using Chan's algorithm [30].

The second is to check if the hyperbola or an ellipse overlaps with the TAS using a filter algorithm and discard any constraint that does not overlap with the convex hull of the TAS. For the hyperbola, it uses the rectangle which is perpendicular with the line between the foci with length equal to $2H_{max}$, width equal to H_{max}, and the start point is the midpoint between the foci, to approximate the hyperbola. For an ellipse, it uses circles with radius a and the foci as the center to approximate the ellipses constraints where a is half of the long axis length of the ellipse.

Third, check if the hyperbola or ellipse overlaps with the control circle. In this step, the filter algorithm not only needs to discard the shape constraints with zero or trivial area but also needs to store the area of the shape for each constraint and send them to the GPU along with all the possible shape constraints and the TAS for further calculation.

$$Area_{Segment} = Area_{outerOverlap} - Area_{innerOverlap} \tag{6}$$

3.3.2. Calculating the Area of the Shape Segments. To ensure the accuracy of the effective coverage ratio (ECR) as shown in Equation (4), the concise computation of the shape segments' area is essential. This paper defines the area of an ellipse segment $Area_{Segment}$ (in Equation (6)) as the difference of the area of the outer ellipse overlapping with the control circle $Area_{outeroverlap}$ and the area of the inner ellipse overlapping with the control circle $Area_{innerOverlap}$(the shaded area shown in Figure 4) where the control circle has the center A_3 and the radius h_3. The outer ellipse and inner ellipse share the same foci A_1 and A_2. Also, for any point on the outer ellipse, the distance from it to the foci (h_1 and h_2) meets $h_1 + h_2 = 2 \times a$ where a is a predefined value. The distance from any point on the inner ellipse to the foci (h'_1 and h'_2) meets $h'_1 + h'_2 = 2 \times a'$.

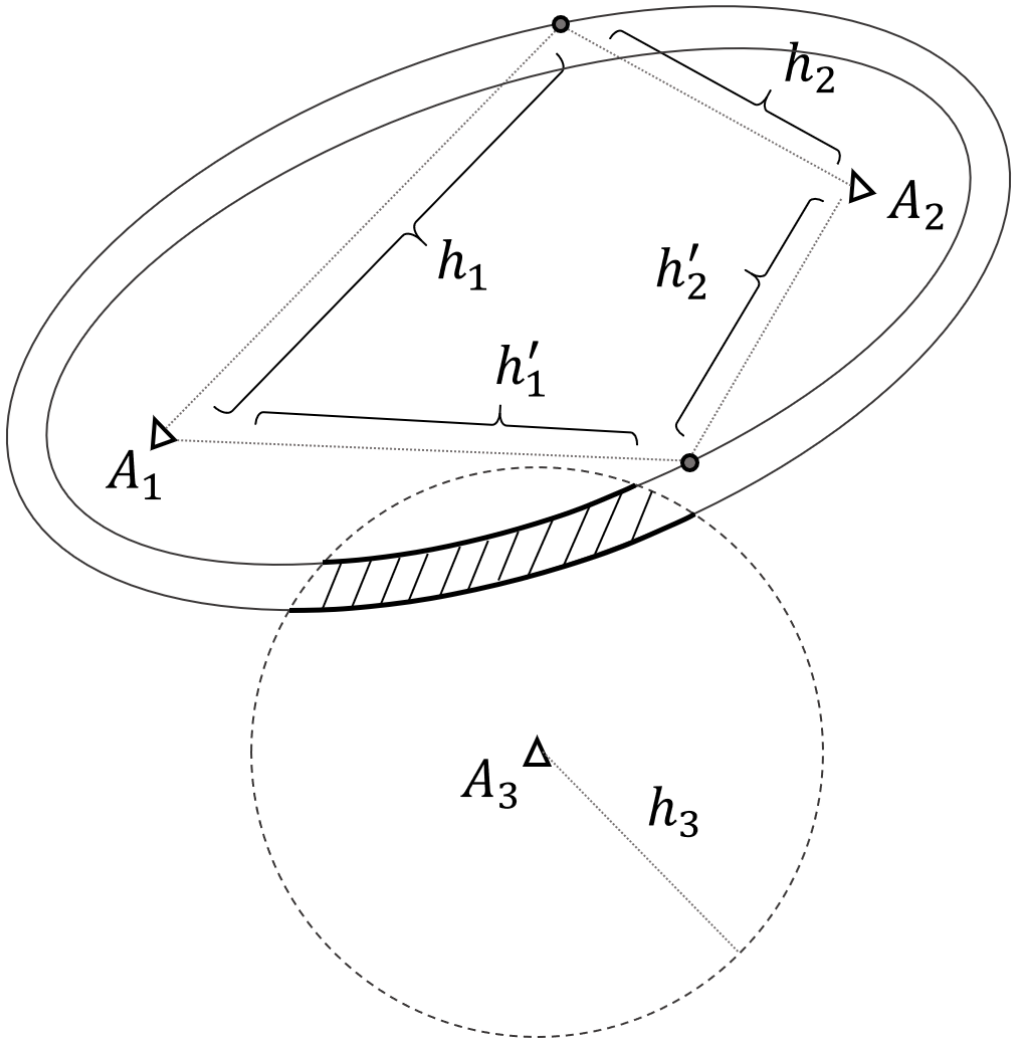


Figure 4. A ellipse segment's area estimation example

The objective now is to calculate the overlapping area of the inner ellipse and the control circle, and the overlapping area of the outer ellipse and the circle. To do that, the first step is to transfer the ellipse to standard Cartesian format as in Equation (2) by rotating the foci to be horizontally aligned and shift the center of the ellipse to the origin point of the predefined coordinate.

The second step is to find the intersection point of the ellipse and the circle by solving the simultaneous equations of the Cartesian equation of the ellipse (Equation 2) and the Cartesian equation of the circle (Equation 3 where x_c and y_c are the x and y coordinate

of the circle center). The simultaneous equations can be transformed to be a quadratic equation (Equation 7) which has zero to four real number solutions for y. Each real solution indicates one intersection point of the ellipse and the circle.

$$\begin{cases} x = \frac{a\sqrt{b^2-y^2}}{b} \\ y^4 + S_1 y^3 + S_2 y^2 + S_3 y + S_4 = 0 \\ m = b^2 - a^2 \\ n = a^2 b^2 + b^2 c_x^2 + b^2 c_y^2 - b^2 r^2 \\ S_1 = -4b^2 c_y / m \\ S_2 = (2mn + 4b^4 c_y^2 + 4b^2 a^2 c_x^2)/m^2 \\ S_3 = -4b^2 c_y n / m^2 \\ S_4 = (n^2 - 4b^2 c_x^2 a^2 b^2)/m^2 \end{cases} \tag{7}$$

The third step is to simplify and adapt the algorithm in [31] for computing the overlapping area of an ellipse and the circle. The detailed procedures and algorithms are described in the Appendix.

A hyperbola's segment constraints are defined as the overlapping area of a hyperbola with one hop width and a control circle with a given radius. The hyperbola is defined with two anchor nodes which are its foci. The control circle is defined with one anchor node which is its center. To find all possible hyperbola segments, The proposed algorithm uses three layers nested for-loop to iterate through the anchor nodes A_1, A_2, A_3 as shown in Equation (5). For each shape, it iterates hops from h_{start} to h_{end}. It also calculates the average one-hop distance between the foci of the hyperbola as d_{focus} and the average one-hop distance between all three anchor nodes as d_{avg} which is used to estimate the overlapping area of the hyperbola segment and the control circle discussed in Figure 5.

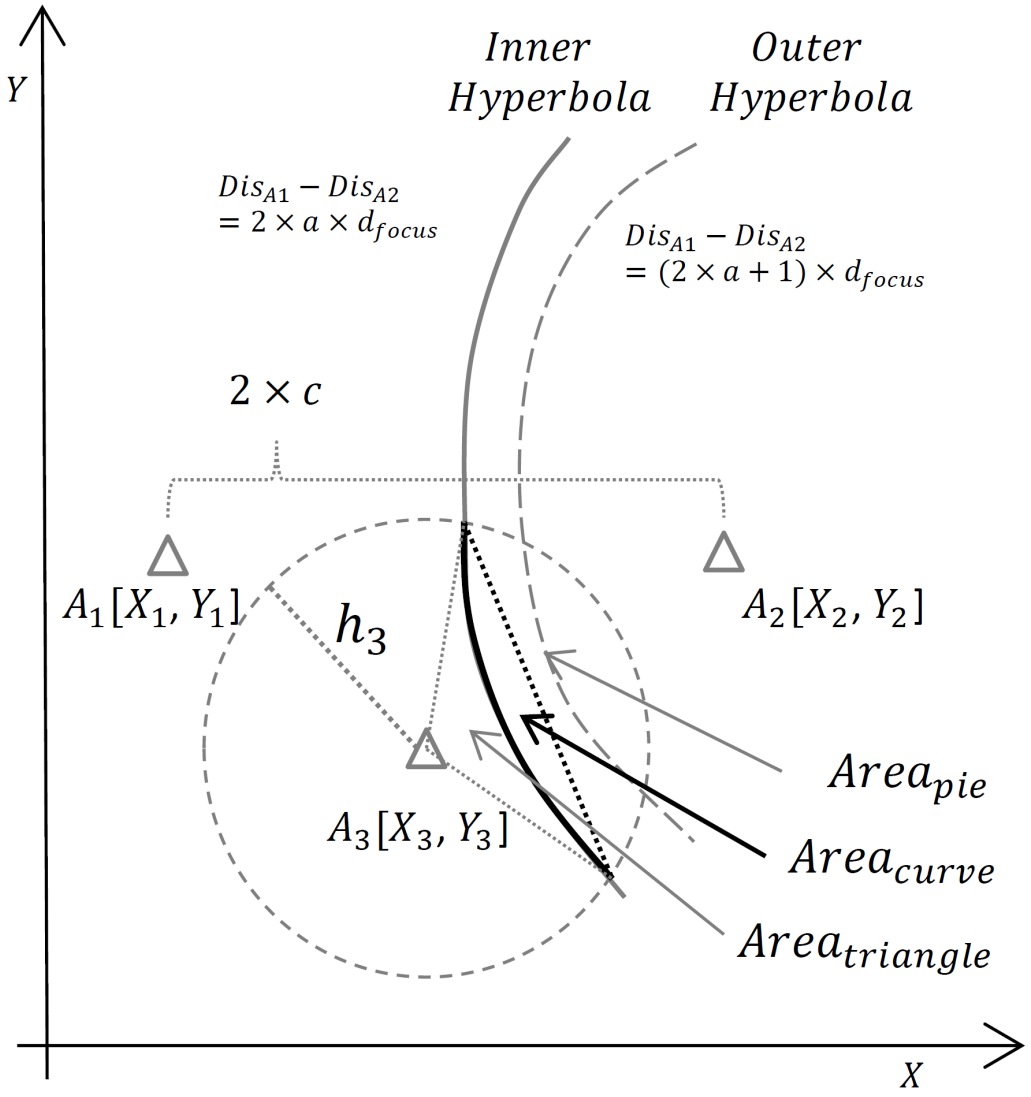


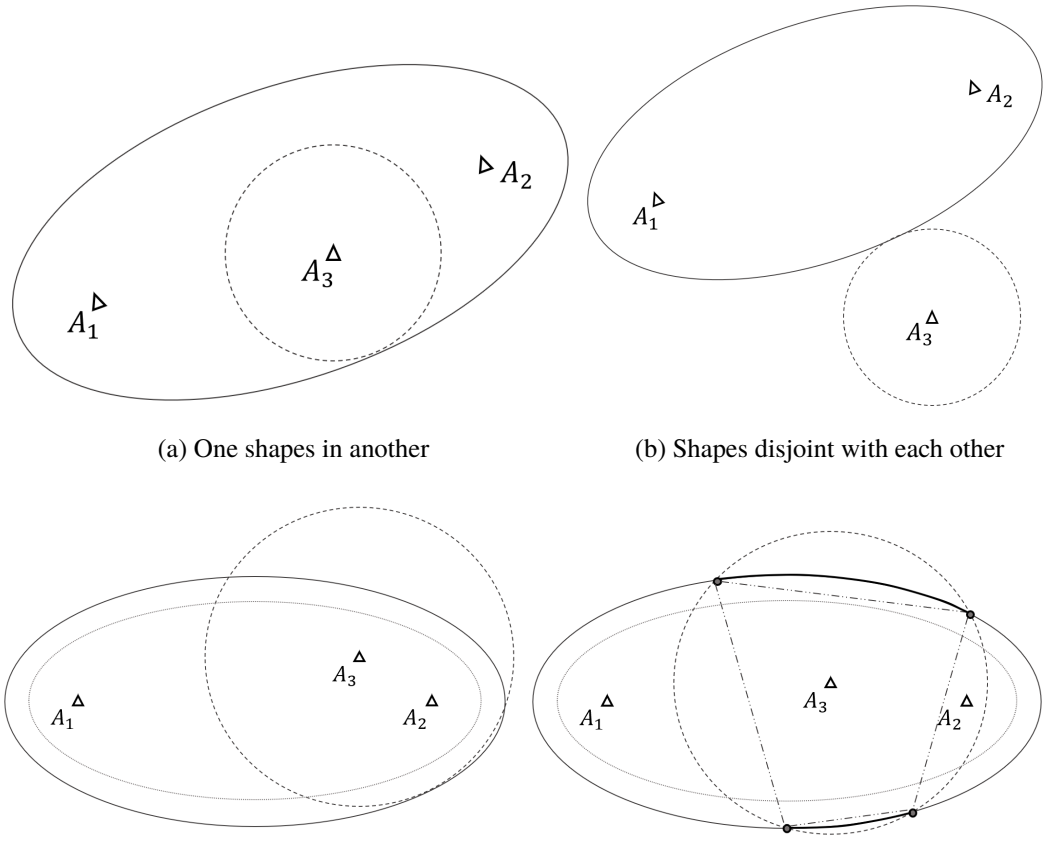
Figure 5. A hyperbola segment's area estimation example

For example, in Figure 5, assume that the x-coordinate of the anchor nodes is X_{ID} where "ID" is the anchor node's identification number and the y-coordinate of the anchor nodes is Y_{ID}. Assume that the Euclidean distance between A_1 and A_2 is $2 \times c$. The overlapping area of the hyperbola of A_1 and A_2 and the control circle of A_3 is defined as $Area_{hyper} = Area_{innerHyper} - Area_{hyperbola}$. The inner hyperbola meets $Dis_{A1} - Dis_{A2} = 2 \times a \times d_{focus}$. Here $Dis_{A_{ID}}$ is the distance between any point in the hyperbola to the focus with given 'ID'; 'a' is a positive integer less than H_{max}. The overlapping area of a hyperbola

and the control circle can be calculated as follows:

$$Area_{hyper} = Area_{pie} + Area_{triangle} - Area_{curve}$$

As shown in Figure 5, the $Area_{curve}$ is the area between the line connected to the two intersection points and the hyperbola curve between the two intersection points. This area is the result of the definite integral of the hyperbola function minus the line function. The detailed algorithms are given in the Appendix.



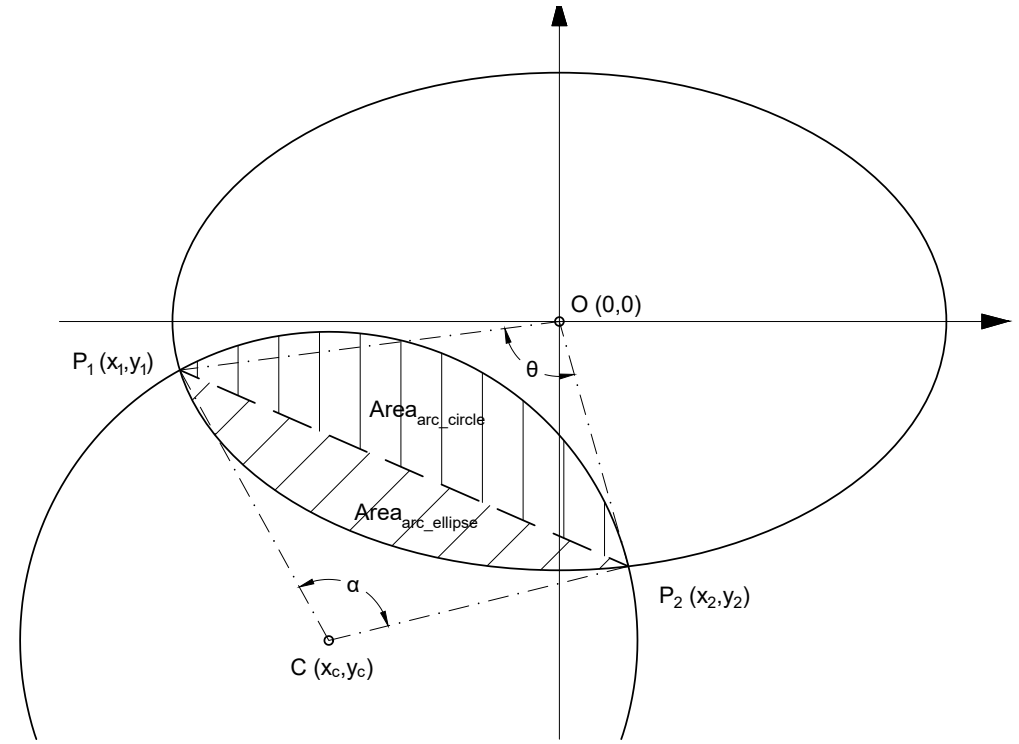
(a) One shapes in another

(b) Shapes disjoint with each other

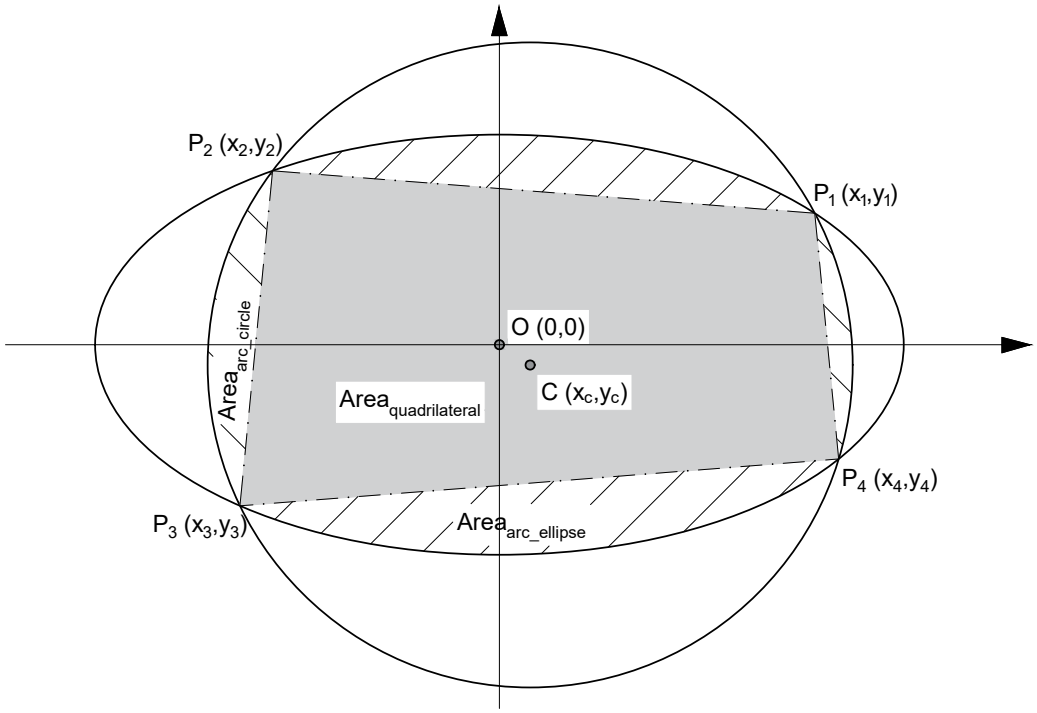
(c) Three intersection points

(d) Four intersection points

Figure 6. Ellipse and circle intersection cases



(a) Two intersection points



(b) Four intersection points

Figure 7. Example of finding overlapping area of ellipse and circle

3.4. CALCULATE THE OVERLAPPING AREA OF ELLIPSE AND CIRCLE

Here, we simplify the algorithm of finding overlapping area of ellipse and circle into five cases. When there is one or zero intersection point, we have the following two cases.

First is a single shape within another as in Figure 6-(a), and the other is two disjoint shapes as in Figure 6-(b). These two cases can be determined by checking the distance of the centers of the ellipse and the circle. When there are two intersection points, the ellipse and circle must have overlapping area which is a portion of the 'pie' area (Triangle area $Area_{triangle}$ plus arc area $Area_{ellipse}$ and $Area_{circle}$) of the control circle as shown in Figure 7. To calculate the triangle area $Area_{triangle}$, we can use Equation (8) where $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ are three vertices of the triangle. We can use Equation (12) and (13) to calculate the arc area. The α and θ in Equation (12) and (13) is the angle difference of vector (O, P_1) and (O, P_2) (for polar angle θ) or the angle difference of vector (C, P_1) and (C, P_2) (for angle α), which can be calculated with Equation (9) or (10).

For three intersection points cases shown in Figure 6-(c), one intersection point is the tangent point which we should be omit in the area computation. The last case is the ellipse crossing the control circle as shown in Figure 6-(d). In this case, four intersection points exist ($P_1(x_1, y_1), P_2(x_2, y_2), P_3(x_3, y_3), P_4(x_4, y_4)$). The total overlapping area is the sum of four arc area which can be calculated using Equation (8),(12) and(13), and a quadrilateral area which can be calculated using Equation (14).

$$Area_{triangle} = |x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)|/2 \tag{8}$$

$$Angle\ \alpha_{p_n} = \begin{cases} \arccos(x_n/r) & \text{if } y_n >= 0 \\ 2\pi - \arccos(x_n/r) & \text{Otherwise} \end{cases} \tag{9}$$

$$Polar\ Angle\ \theta_{p_n} = \begin{cases} \arccos(x_n/a) & \text{if } y_n >= 0 \\ 2\pi - \arccos(x_n/a) & \text{Otherwise} \end{cases} \tag{10}$$

$$sign = \begin{cases} 1 & \text{if } \alpha \text{ (or) } \theta > \pi \\ 0 & \text{if } \alpha \text{ (or) } \theta = \pi \\ -1 & \text{if } \alpha \text{ (or) } \theta < \pi \end{cases} \tag{11}$$

$$Area_{arc_{cirle}} = \alpha r^2/2 + sign_c * Area_{triangle} \tag{12}$$

$$Area_{arc_{ellip}} = \theta ab/2 + sign_e * Area_{triangle} \tag{13}$$

$$Area_{quad} = |(x_3 - x_1)(y_4 - y_2) - (x_4 - x_2)(y_3 - y_1)|/2 \tag{14}$$

Fourth, find the overlapping area of the inner ellipse and control circle ($Area_{outerEllipse}$) by repeating the previous steps. Then subtract $Area_{outerEllipse}$ from the overlapping area of the outer ellipse and control circle ($Area_{innerEllipse}$). The residual will be the area of the ellipse segment $Area_{arc}$. The detail algorithm can be referred to Algorithm (1) and (2).

Last, tiny ellipse segments, which only contain less than 1000 pixels, are trival and won't be count to the final result. To save computational power, all these small ellipse segments will be discarded.

Algorithm 1: Get the area of the ellipse segment

Result: Area of ellipse segment: $Area_{arc}$

Input : $outerEllipse$, $controlCircle$, d_{avg}

1 $innerEllipse = new\ Circle(arcCircle, d_{avg})$;

2 $Area_{outerEllipse}$ = CALL Algorithm(2):
$ellipseIntersectArea(outerEllipse, controlCircle)$;

3 $Area_{innerEllipse}$ = CALL Algorithm(2):
$ellipseIntersectArea(innerEllipse, controlCircle)$;

4 **return** $Area_{arc} = Area_{outerEllipse} - Area_{innerEllipse}$;

Algorithm 2: Get the overlapping area of ellipse and circle

Result: Intersect area of ellipse and circle

Input : $ellipse, circle$

$P \leftarrow EmptyList$;
$P \leftarrow getIntersectionPoints(ellipse, circle)$;
if $size(P) <= 1$ **then**
    if $is\_overlapping()$ **then**
        return $min(\pi r^2, \pi ab)$;
    else
        return 0;
    end
else if $size(P) <= 3$ **then**
    $(P_1, P_2) \leftarrow removeTangentPoint(P)$;
    //Equation (12)(13)
    $Area_{circle} = findCircleArcArea(P_1, P_2)$;
    $Area_{ellipse} = findEllipseArcArea(P_1, P_2)$;
    $Area_{overlap} = Area_{circle} + Area_{ellipse}$;
else
    $Area_{overlap} = findQuadArea(P)$;
    for $(i = 0; i < 4; i++)$ **do**
        $Area_{circle} = findCircleArcArea(P[i], P[(i+1)\%4])$;
        $Area_{ellipse} = findEllipseArcArea(P[i], P[(i+1)\%4])$;
        $Area_{overlap} += Area_{circle} + Area_{ellipse}$;
    end
end
return $Area_{overlap}$;

3.5. CALCULATE THE OVERLAPPING AREA OF HYPERBOLA AND CIRCLE

Algorithms 3 and 4 show the procedure of calculating the area of a segment of the intersection of a hyperbola and a circle. First, the hyperbola is rotated and shifted to standard format, which obeys Equation (15) by multiplying the hyperbola with the transformation matrix (16) and subtracting a shift vector shown in Equation 17. Then, multiplying the circle with center A_3 with the same transformation matrix 16, and shift using vector 17, which gives us the new center shown in Equation (18). With the transformed circle center, we can get the circular function as in Equation (19).

$$\frac{x^2}{a^2} - \frac{y^2}{(c-a)^2} = 1 \tag{15}$$

$$T = \begin{bmatrix} \frac{c\times(X_2-X_1)}{2\times((X_2-X_1)^2+(Y_2-Y_1)^2)} & \frac{c\times(Y_1-Y_2)}{2\times((X_2-X_1)^2+(Y_2-Y_1)^2)} \\ \frac{c\times(Y_2-Y_1)}{2\times((X_2-X_1)^2+(Y_2-Y_1)^2)} & \frac{c\times(X_2-X_1)}{2\times((X_2-X_1)^2+(Y_2-Y_1)^2)} \end{bmatrix} \tag{16}$$

$$V = \begin{bmatrix} (X_1+X_2)/2 \\ (Y_1+Y_2)/2 \end{bmatrix} \tag{17}$$

Algorithm 3: Get the area of hyperbola segment

Result: Area of hyperbola segment: $Area_{hyper}$

Input : $hyperbola, controlCircle, d_{avg}$

1 $innerHyper = new\ Hyperbola(hyperbola, d_{avg})$;

2 $Area_{hyperbola}$ = CALL Algorithm(4):
$hyperIntersectCircle(hyperbola, controlCircle)$;

3 $Area_{innerHyper}$ = CALL Algorithm(4):
$hyperIntersectCircle(innerHyper, controlCircle)$;

4 **return** $Area_{hyper} = Area_{innerHyper} - Area_{hyperbola}$;

Algorithm 4: Get the area of hyperbola intersect circle

Result: Area of hyperbola intersect circle

Input : $hyperbola\ h, circle\ c$

$standardize(\&h, \&c)$; //Eq:(15)-(19);

$coefficients = quarticCoefficient(h, c)$;

$roots = getRealRootsQuarticEq(coefficients)$;

$List_p = findIntersectionPoint(roots, h, c)$;

for $i \leftarrow 0$ **to** $size(List_p)$ **by** 2 **do**

  $\phi = intersectAngle(List_p[i], List_p[i+1])$;

  if $\sin\phi < 0$ **then**

    $Area_{pie} = \arccos(\cos\phi) * r^2/2$;

  else

    $Area_{pie} = (2 * pi - \arccos(\cos\phi)) * r^2/2$;

  end

  $Area_{triangle} = \sin\phi \times r^2/2$;

  $Area_{curve} = FindCurveArea(h, List_p[i], List_p[i+1])$;

  $Area_{hyper} += Area_{pie} + Area_{triangle} - Area_{curve}$;

end

return $Area_{hyper}$;

The Equations 15 and 19 can be merged into a quartic-equation. Then, we can approximate the intersection points of the hyperbola and the circle by solving the quartic-equation and sort the points list, $List_p$, based on the y-coordinate values.

$$\begin{bmatrix} X_{cycle} \\ Y_{cycle} \end{bmatrix} = T \cdot \left(\begin{bmatrix} X_3 \\ Y_3 \end{bmatrix} + V \right) \tag{18}$$

$$(x - X_{cycle})^2 + (y - Y_{cycle})^2 = (h_3 \times d_{avg})^2 \tag{19}$$

3.5.1. Compute the Effective Coverage Ratio (ECR) and Elect the Best Shapes. To calculate the $Area_{covered}$, a brute force method is used by testing hop constraints pixel by pixel. Different from the first step which uses a lot of condition branches, the second step has few branches. Thus, GPU can accelerate its computing. The NVIDIA CUDA kernel that calculates the ECR is designed as follows:

For the ellipse arc, if the distance from any pixel in the TAS to A_1 (Dis_{A_1}) and $A_2(Dis_{A_2})$ obeys $Dis_{A_1} + Dis_{A_2} \leq 2 \times d_{avg} \times a$ and any pixel in TAS to $A_3(Dis_{A_3})$ obeys $Dis_{A_3} \leq d_{avg} \times h_3$, then that pixel is covered by the arc shape. For the hyperbola, if the distance from any pixel in the TAS to A_1 (Dis_{A_1}) and to A_2 (Dis_{A_2}) obeys $Dis_{A_1} - DisA_2 = 2{\times}a{\times}d_{focus}$ and the distance from any pixel in the TAS to $A_3(Dis_{A_3})$ obeys $Dis_{A_3} \leq h_3{\times}d_{avg}$, then that pixel is covered by the hyperbola shape (note that here all the notations used are the same as in Equation (5) and Figure 5). The GPU algorithm considers both the total number of pixels in TAS covered by the element shapes and the Effective Coverage Ratio (ECR). In each iteration, the proposed greedy algorithm only selects the shape which provides the maximum value of the Greedy Factor (GF) which is the number of pixels multiplied by cubic ECR as shown in Equation (20):

$$GF = Area_{covered} \times ECR^3 \tag{20}$$

So each iteration will eliminate some pixels of TAS, which also exists in the best shape in 3.5.1. Then, the newly covered pixels is calculated for each possible shape with the updated TAS. Also, instead of calculating a new ECR for each shape, the algorithm reuses the ECR calculated in the first iteration. The shape with the maximum GF is chosen as the best shape. This procedure is repeated until the size of the updated TAS is less than $1 - Th$ of the original size, where Th is the predefined coverage threshold. This procedure is discussed in Figure 24, which generates the final sequence of shapes that cover most of

the TAS. If the trajectory encoding message exceeds the packet size limitation, shown in Table 1, the POI needs to be divided into two, and create two separate trajectories with two different gateway nodes and encode them separately. It is shown in Figure 1.

3.6. ROUTING DECISION FOR WIRELESS SENSORS

The message structure of TinyOS has an 11-byte header that includes the sender's address, the type, and the group data. The payload structure, defined in Table 1, is used for the implementation of DV-TE-R and adapted DV-TE-BR discussed in Section 4.

Table 1. Payload data structure

Descriptions	Starting Bytes	Length in Bytes
Message ID	0	4
Parent Node address	4	2
Hop counts	6	1
relaxation parameter	7	1
constraints for relax	9	6
routing constraints	14	100

The counter-based routing decision is also used to mitigate the broadcast storm effect [5]. In the proposed counter-based routing decision, each wireless sensor uses two kinds of criteria to decide if it should forward the routing packet or not. The first is to check if it meets any constraint of the encoded trajectory ($flag_C$). The second is to check if the counter used to count the number of the nearby redundant re-broadcasting for the same packet reaches the threshold ($flag_T$).

As discussed in 3.3, the ellipse constraint has a size of four bytes: A_1, A_2, A_3, a, h_3. For any sensor node, setting $flag_C$ to be true means its DV-Hop entry of A_1, A_2, A_3 obeys Equation (21). The hyperbola constraint has a size of five bytes: A_1, A_2, A_3, a, h_3. For the same sensor node, $flag_C$ should be true if its DV-Hop entry of A_1, A_2, A_3 satisfies Equation (22). To distinguish the ellipse constrains from the hyperbola constraints, the ellipse constraint sets the a value to be negative while the hyperbola constraint uses positive a value.

$$\begin{cases} floor(DV_{Hop}[A_1] + DV_{Hop}[A_2])/2 == |a| \\ DV_{Hop}[A_3] \leq h_3 \end{cases} \tag{21}$$

$$\begin{cases} floor(DV_{Hop}[A_1] - DV_{Hop}[A_2])/2 == |a| \\ DV_{Hop}[A_3] \leq h_3 \end{cases} \tag{22}$$

Table 2. Encoding result for sample trajectories

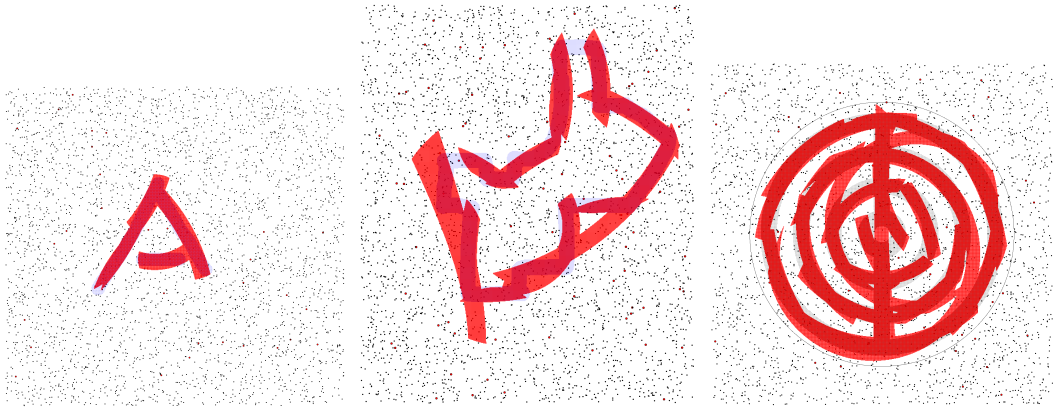
Trajectory type	A	outline	Circles
Number of circle arcs (without ellipse)	3	12	17
Number of hyperbolas (without ellipse)	2	1	4
Message length(byte)	23	54	89
Number of ellipse arcs (with ellipse)	2	6	12
Number of hyperbolas (with ellipse)	2	4	5
Message length(byte)	21	51	86
Compressed size with JPEG(byte)	867	887	934

The counter-based routing decision is first proposed in [5]. Once a sensor node broadcasts a packet to its neighbors, the neighbors will be listening to the channel for a random amount of time before it forwards the packet. During the listening period, the sensors will count the number of times the same packet has been forwarded. If it exceeds the counter threshold, it will set the $flag_T$ to be false so only the sensors with $flag_C == true$ and $flag_T == true$ will forward the routing packets. Another case is for a low-power listening WSN where sensors hibernate for most of their lifetime. The sensors will rebroadcast immediately if they find they satisfy Equation (21) and Equation (22) ($flag_C == true$) and initialize a counter with value 0. Then, they will stop broadcasting when there is a timeout or when their rebroadcasting neighbors' number reaches the counter threshold ($flag_T == true$).

3.7. SAMPLE ROUTING RESULT AND ANALYSIS

The proposed geospatial area encoding algorithm works for any shape and trajectory. Figure 8 shows the encoding algorithm on some sample trajectories. For each of the routing trajectories, assume anchor node ID is one byte long and the number of hops is one byte long. As shown in Table 2, the length of the encoded trajectory is 23, 54, and 89 bytes when encoding without the use of ellipse constraints, and the length of the encoded trajectory is 21, 51, and 86 when encoding with the ellipse constraints. The red shapes, which represent the encoded trajectory, shown in Figure 8-(a),(b),(c) are the cascaded circle arc and hyperbola shapes. Figure 9 shows the encoding result of trajectory handwriting "A" and hand drawing park boundary when encoding with ellipse and hyperbola constraints. JPEG compression algorithm is lossy for images. The above JPEG example has a 64×64 resolution while has only about 800 bytes size. Assume that there are "n" anchor nodes in the local edge network, each anchor node floods at most "r" hops, and the TAS has "m" entries. Then the time complexity of finding the best ellipse arc and the best hyperbola segment is $O(mn^3r^3)$. Another experiment is conducted with anchor nodes from 20 to 80, and the TAS is from

10000 to 100000. Figure 10 shows when the number of anchor nodes increases, the CPU time of calculating the area of all possible shapes has a polynomial growth. Figure 11 (b) shows that the GPU time of finding the best shape also has polynomial growth when the number of anchor nodes increases. However, Figure 11 (a) indicates that the GPU time of finding the best shape has a linear relationship with the size of TAS.



(a) Encoding trajectory 'A'

(b) Encoding trajectory of Breuer Park, Rolla, MO

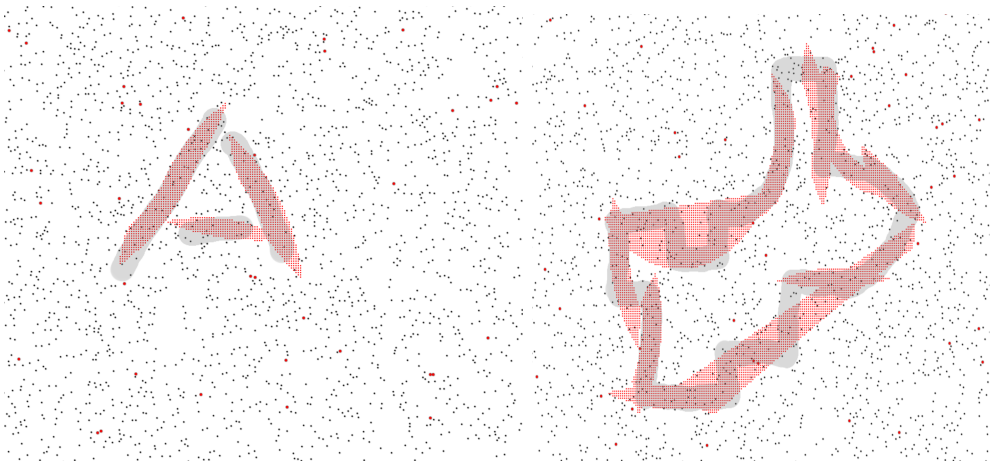
(c) Encoding trajectory 'rings'

(d) JPEG compressed hand drawing A

(e) JPEG compressed park boundary

(f) JPEG compressed nested rings

Figure 8. Example encoding with the circle and the hyperbola constraints and the compressed trajectory using JPEG



(a) Hand drawing 'A'

(b) Hand drawing boundary of Breuer Park,Rolla,MO

Figure 9. Example Encoding of ellipse and hyperbola

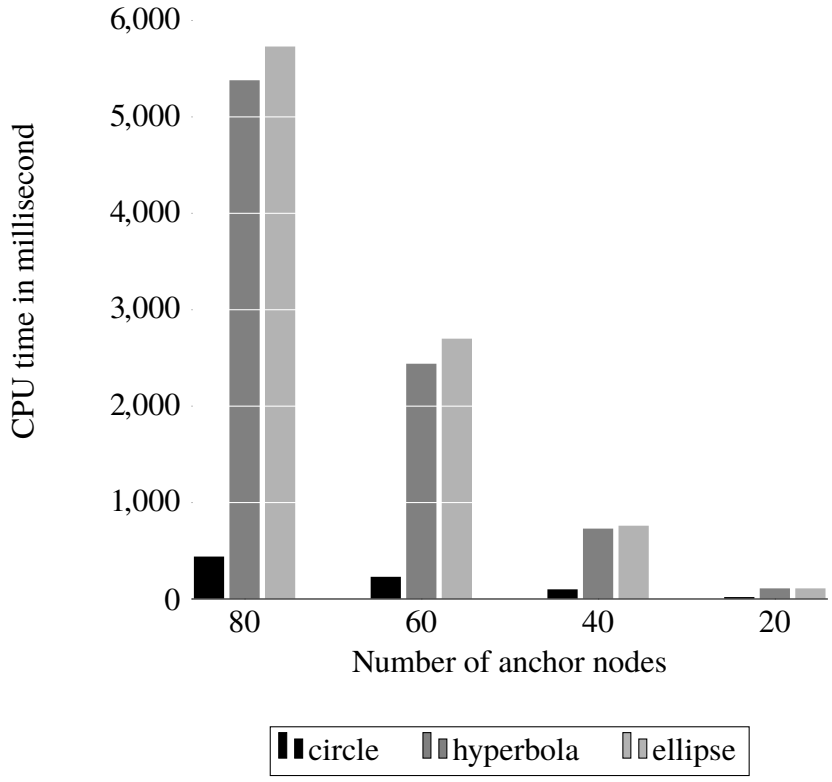
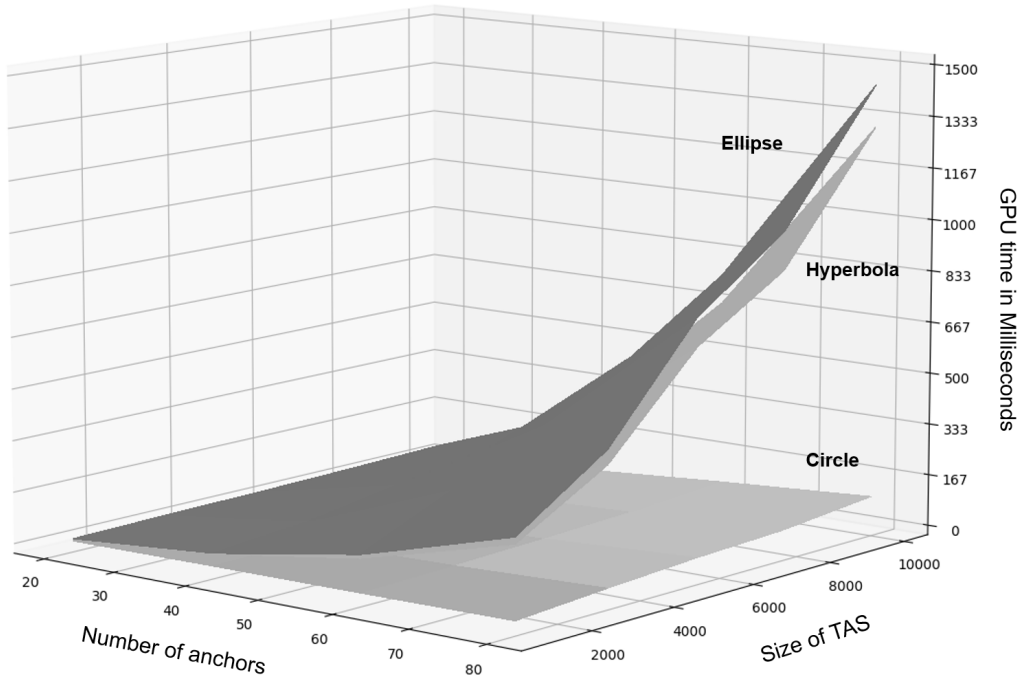
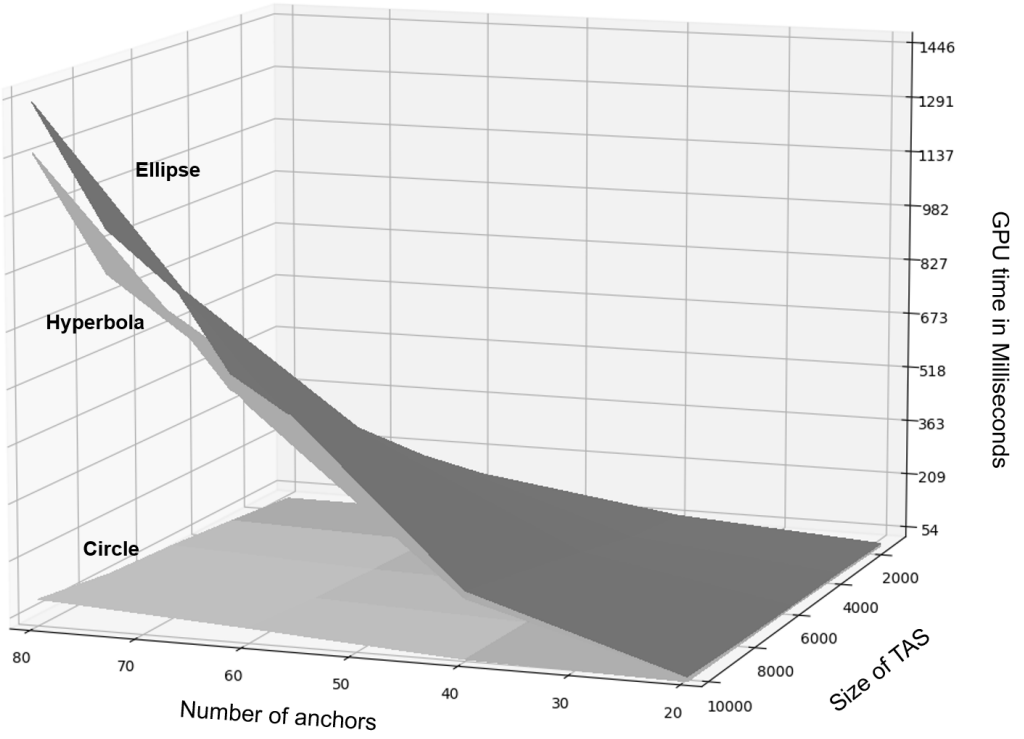


Figure 10. CPU time calculating area of all possible shapes



(a) GPU time versus size of TAS



(b) GPU time versus number of anchor nodes

Figure 11. GPU time finding the best shape

4. ADAPTED DV-HOP BASED DATA COLLECTION SCHEME FOR LOW-POWER WSN

To ensure high QoS in a WSN, the DV-Hop based trajectory encoding and routing protocol (DV-TE-R) has been proposed. For low-power WSN, which can be deployed in a harsh environment, the density of the network topology can be heterogeneous. Somewhere in the region, the sensors may be sparsely deployed, or the routing path could be obstructed by some "holes" shown in Figure 12. Now the user wants to forward a packet from node S_i to S_o through an arc with center A and radius h hops with 1 hop width. Although the DV-hop of both the nodes S_i and S_o is h, these two nodes are not directly connected because of an obstacle between them. If using local broadcasting, for example in Figure 12, to fix the routing path, at least 4 hops extra broadcast is required which is a huge overhead. Therefore, the routing protocol adopts a bridge on the edge adaption (DV-TE-BR) that could connect a broken routing path with minimum overhead.

After a forwarder node has forwarded the routing packet and has not overheard any rebroadcast from its neighboring nodes nor acknowledgment from the sinks, it will start iterating its valid constraints, relax those by one hop, note all the changes, and rebroadcast the packets again. If it receives the rebroadcast from its neighbors, it will stop iterating and go to sleep immediately. If a node receives the relaxed-constraint packet, it will tighten the constraint by one, and repeat the previous procedure until recovering the original constraint. Note that both h_1 and h_2 are relaxed for the arc constraint, and both a and h_3 are relaxed for hyperbola constraint.

For example, like Figure 12, the nodes B_1, B_2, B_3, B_4 have an increased DV-hop entry of A from $h + 1$ to $h + 2$. Thus, the sensor node S_i needs to relax the hop constraints by one. Then, the node B_1 needs to relax the hop constraint by one more. For B_2, it will hold the constraints as of B_1. The constraint is tightened by one for B_3, and so does B_4. Finally, the route is fixed after B_4 forwards the packet to S_o.

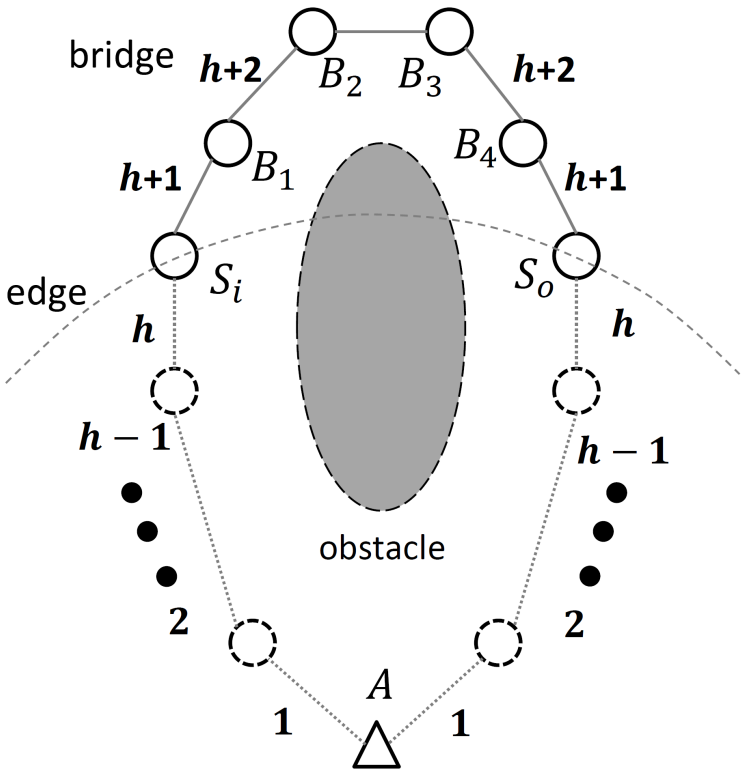


Figure 12. Example of the bridge on the edge adaption

5. EXPERIMENTS AND RESULTS

The experiments are conducted with a simulation tool TOSSIM and a real sensor network test-bed. The simulation uses a randomly created large-scale wireless sensor network as the working area which contains both heterogeneous and homogeneous topology. It uses real-world trajectory data (taxi-trajectory [32]) for sensor data collection. The experiments also include the performance evaluation of the proposed data collection algorithm in WSN with mobile nodes.

The structure of our experiments is listed as follows: First, an experiment to validate the effeteness of the proposed trajectory-based routing protocol in a real wireless sensor network with 28 nodes. Second, an experiment to evaluate the proposed data collection scheme in a large static WSN with a mobile sink using the TOSSIM simulator. The simulation uses the real-world taxi trajectory dataset. Third, an experiment to test routing

in the shortest path to the mobile sink and compared the delay and energy consumption comparing to ring routing and nested routing. Fourth, a Python program to evaluate the proposed data collection scheme in a WSN with mobile sensor nodes. It shows the data collection area scatters with the movement of sensors.

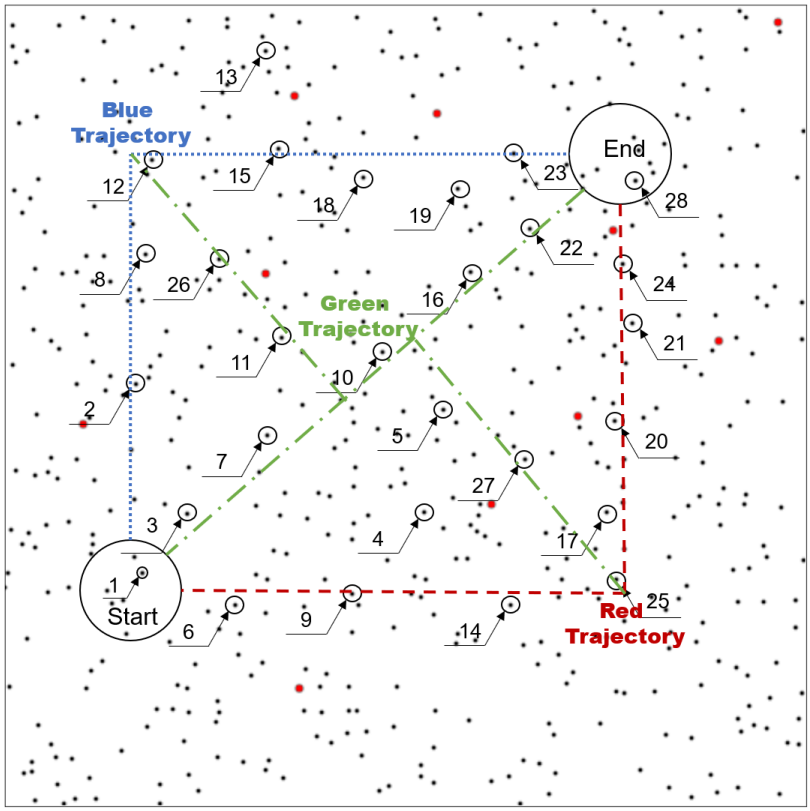


Figure 13. Experimental WSN and routing trajectory

5.1. PERFORMANCE USING A WSN TEST-BED

This real-world experiment contains a small wireless sensor network, which is a mix of 16 TelosB sensors [33] and 12 MicaZ sensors [34], with minimal radio power configuration, and hard-coded DV-Hop information of 10 virtual anchor nodes. The experiments test the performance of broadcasting the data collection requests toward the location of interest

through a given trajectory. The performance metrics were mainly focused on the coverage ratio and latency. The baseline comparison was with the counter-based broadcasting scheme discussed earlier.

The 28 wireless sensors are coded with the proposed DV-Hop based broadcasting protocol, and the counter based broadcasting scheme. The sensors are labeled and hard-coded with the DV-Hop of 10 virtual anchor nodes. The values of the DV-Hops are generated with a simulation tool that simulates a WSN with 500 nodes and 10 anchor nodes. It selects 28 nodes from the 500 nodes in the WSN. Three trajectories are used in this experiment as shown in Figure 13. The blue trajectory (dot line) has a "Γ" shape and is located near the top left corner of the WSN. The red trajectory (dash line) has a "_|" shape and is located on the bottom right corner of the WSN. The green trajectory (dash-dot line) has a shape that looks like an "X" and is located in the middle of the WSN.

Next, the wireless sensor network deployed follows the same topology as in the simulation tool based on their labels. Node 1 is the start node and node 28 is the sink node for all three trajectories. In the center of the WSN, the green trajectory is a multi-casting route that forwards packets from node 1 to nodes 28, 25, and 12 at the same time.

Then, use the simulation tool to generate the routing messages covering the colored trajectory, and sending the routing messages through the gateway sensor (source node) connecting the laptop near node 1.

Once the sensors rebroadcast the received data request messages, they will blink their LED light until enough neighbors (larger than the counter threshold) rebroadcast or when it timeouts. Also, the predefined destination nodes (POI) will route the sensing data back to the gateway sensors through its parent's node. The time elapsed between the time the source sending the data collection packets and the time the source received back the sensing data is considered to be the latency.

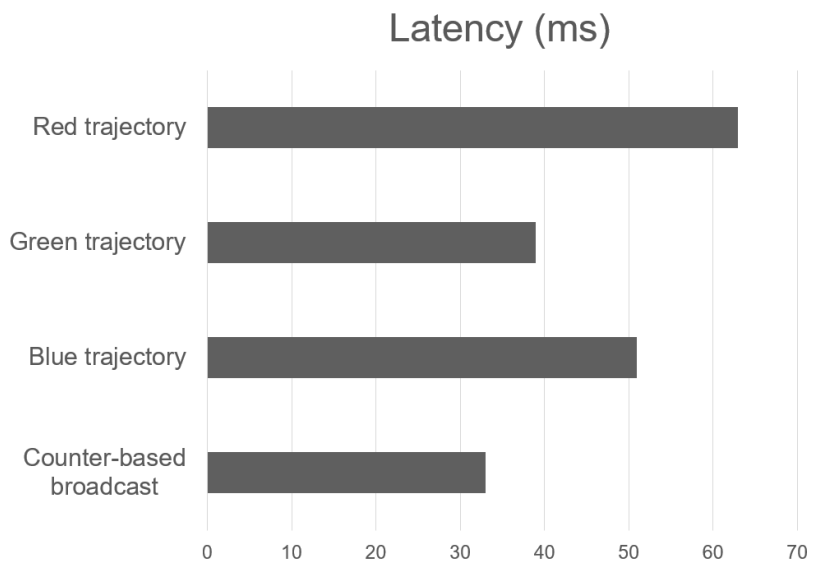


Figure 14. Latency of multi-hop routing when disseminating data collection message with proposed data forwarding approach and counter-based broadcast

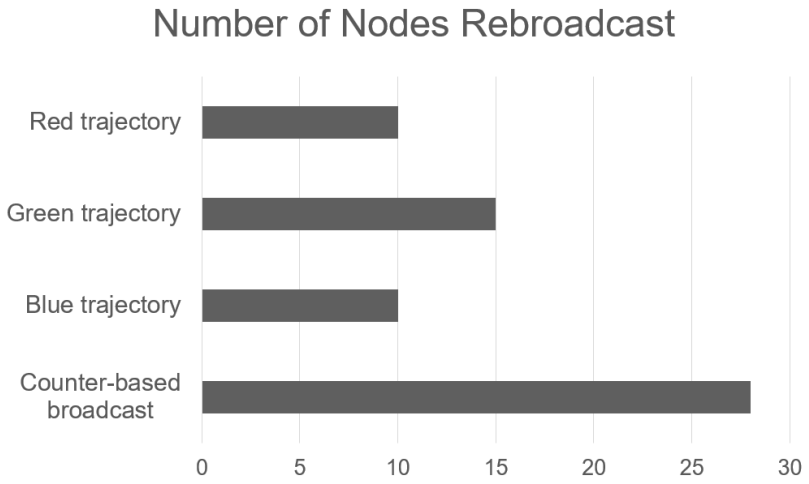


Figure 15. Total number of nodes rebroadcasting when disseminating data collection message with proposed data forwarding approach through trajectory and counter-based broadcast

The experiment shows that the proposed approach can successfully disseminate messages to the desired path by only modifying the constraints in the routing message. Only the sensors on the trajectory that meets the constraints will be activated to rebroadcast

the message. In Figure 14, the multi-hop latency is similar to the broadcasting approach for the green trajectory. The reason is that the routing decision of the proposed DV-Hop based data forwarding approach is made by comparing the routing constraints and the DV-Hop stored in the sensor, which is trivial. Most portion of the delay is caused by the predefined random channel listening period which is similar to the counter-based broadcasting approach (It is used to mitigate the broadcasting storm effect). The proposed approach has better latency when routing through the green trajectory. The reason is that the green trajectory has fewer hops from the start node to the sink node. The broadcasting approach has better latency compared to the proposed data forwarding approach because it can always route messages to the destination through the minimum hop counts. However, as a trade-off, the broadcast approach can only flood through all the nodes while the proposed DV-Hop based data forwarding approach not only can route on the trajectory but also reduce the overhead caused by the redundant rebroadcasting. Figure 15 shows that the baseline(counter-based broadcasting) approach will flood the whole network and will cause all the nodes to rebroadcast even in the small wireless sensor network.

Another experiment is executed with low power listening(LPL)[35] configured WSN. In LPL mode, the sensors fell asleep frequently and only wake up for a small period to listen to the channel. The proposed approach sets the sleeping time as 600 ms and wake up time as 10 ms in one cycle. It is challenging to do routing in an LPL WSN where the network topology is unstable as most of the neighboring sensors fell asleep. The adaption of the routing approach for LPL WSN, discussed in Section 4, doesn't need to rely on the routing table. However, unlike the existing LPL broadcasting scheme, the proposed approach lets the current broadcasting nodes to keep broadcasting until the number of re-broadcasting neighbors reaches the predefined counter threshold or it times out. Figure 16 shows that although the counter based broadcasting has a lower delay due to flooding, the proposed data forwarding protocol still has an acceptable multi-hop routing delay.

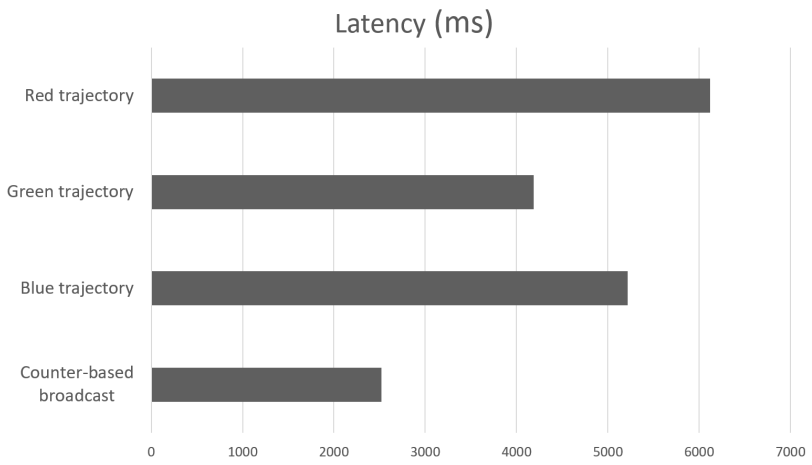


Figure 16. Latency of multi-hop routing when disseminating data collection messages with proposed data forwarding approach and counter-based broadcast in LPL WSN

5.2. SIMULATION RESULTS

The simulation experiment creates random heterogeneous WSNs and routes data collection messages on the real-world taxi trajectory [32]. Coverage performance is defined as follows. The correct coverage ratio is the number of nodes rebroadcasting while on the trajectory over the number of the nodes supposed to rebroadcast (total nodes on the taxi trajectory). It shows the effectiveness of the proposed protocol (higher ratio indicates high accuracy). The redundant rebroadcast ratio is the number of the nodes not in the trajectory while still rebroadcasting over the number of nodes supposed to rebroadcast. A higher redundant rebroadcast ratio indicates a higher overhead of bandwidth and energy consumption.

The experimental parameters are defined in Table 3. The trajectory and POI encoding are executed using a desktop, which acts as a fog server, with a Xeon E5-1620 v2 and an Nvidia RTX 2070 GPU. The POI is located within a WSN distributed in a 2800m by 1700m area. The density of the WSN is less than 0.5% where the density is defined as the average number of neighbors over a total number of sensor nodes. To save energy, all the wireless

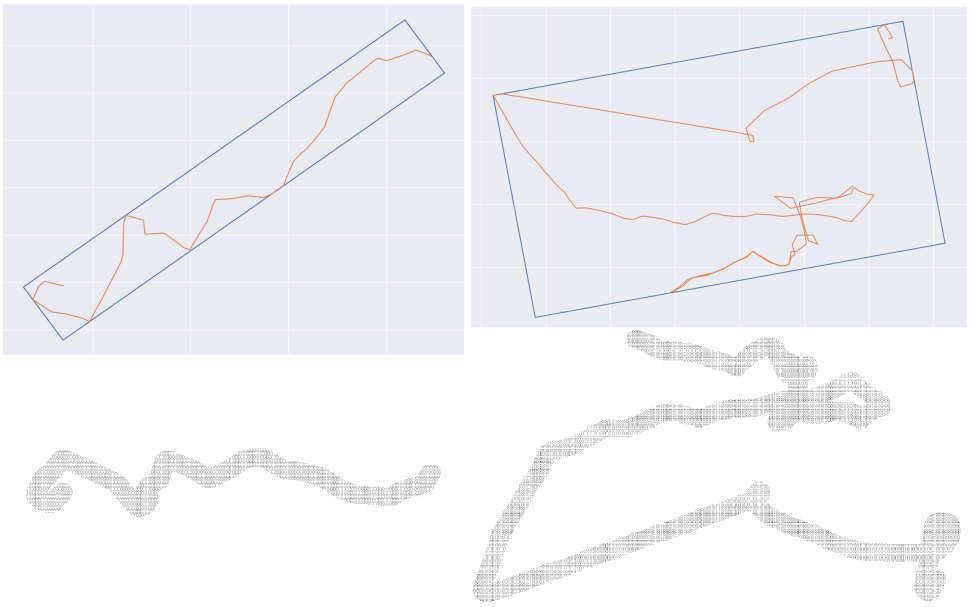
sensors work under low power listening mode (LPL), where each sensor node only wakes up for a few milliseconds to listen to the channel. The WSN is simulated using TOSSIM. The experiment also uses powerTOSSIM-Z to estimate the energy consumption of the activated sensor nodes. There are 30 local edge nodes randomly deployed in the WSN field. The local edge nodes act as the gateway that will broadcast the encoded data requests' packets and collect the data from the WSN. The performance metric used includes the compression ratio, reliability, average delay in data reporting, and energy consumption.

Table 3. Parameters for the experiments

Area of deployment	2800×1700 m
Number of sensor nodes	5000
Communication range	40-100 m
Number of edge devices	30
Broadcasting hop limitation	30 hops
LPL sleeping time	600ms
LPL wake time	10ms
Energy model	MicaZ
Coverage threshold	90%
Number of anchor nodes	(20 - 100)
Anchor cover range	20 hops

As mentioned above, the experiments use real-world taxi-trajectory data [32] as the routing trajectory. Each line of the trajectory data contains the trajectory of a taxi trip in the city of Porto in Portugal. The trajectory is represented as a list of 8 bytes of GPS data (latitude and longitude) sampled every 15 seconds. The 2 GB data-set contains trajectories of different shapes, lengths, starting, and ending locations. To use these trajectories in the

experiments, the first step is to pre-process the data-set by removing the abnormal trajectory and aligning the trajectories in the center of the WSN. To remove the outliers, trajectories that contain abnormal itinerary like where the taxi has abnormal speed, are discarded so that the data-set only contains valid GPS data points for each itinerary. The third step is to reconstruct the itinerary from GPS data points and map it to a 2D trajectory which is the area in the fixed WSN field (TAS). To determine the rectangle field of the WSN and align with the taxi trajectory on it, the fourth step is to use Chan's algorithm [30] to generate the convex hull of the trajectory and then find the minimal rectangle that could wrap the trajectory as shown in Figure 17. The largest length and width of the minimal wrapping rectangles of all the taxi trajectories are chosen as the WSN's outline. Then all the trajectories are shifted and rotated to align to the center of the WSN field. The last step is to set the thickness of the trajectory line to be the average one-hop radio distance of the sensors.



(a) (b)

Figure 17. Find convex hull and the minimal surrounding rectangle for taxi trajectory with 60 GPS data points(left) and 189 GPS data points(right) and the trajectories after pre-process

The statistics of the dataset are shown in Figure 18 where the X-axis shows the number of GPS data points of the trajectory, and Y-axis is the size of the set of node IDs that reside on the trajectory. The figure shows that the trajectory with more GPS data will cover more sensor nodes. In Figure 19 and 20, where the X-axis is the number of GPS data in a trajectory and the Y-axis is the compression ratio, 500 trajectories were sampled, the number of GPS data points ranging from [2, 20], [21, 40], [41, 60], [61, 80], and [81, 105] and the number of anchor nodes in the set [20, 40, 60, 80, 100] which is (0.4%, 0.8%, 1.2%, 1.6%, 2%) of the total number of nodes in the network. The compression ratio (CR) is defined as the uncompressed-data size over the compressed data size using the proposed encoding algorithm. Here the uncompressed data size is the size of a list of sensor IDs that reside in the trajectory.

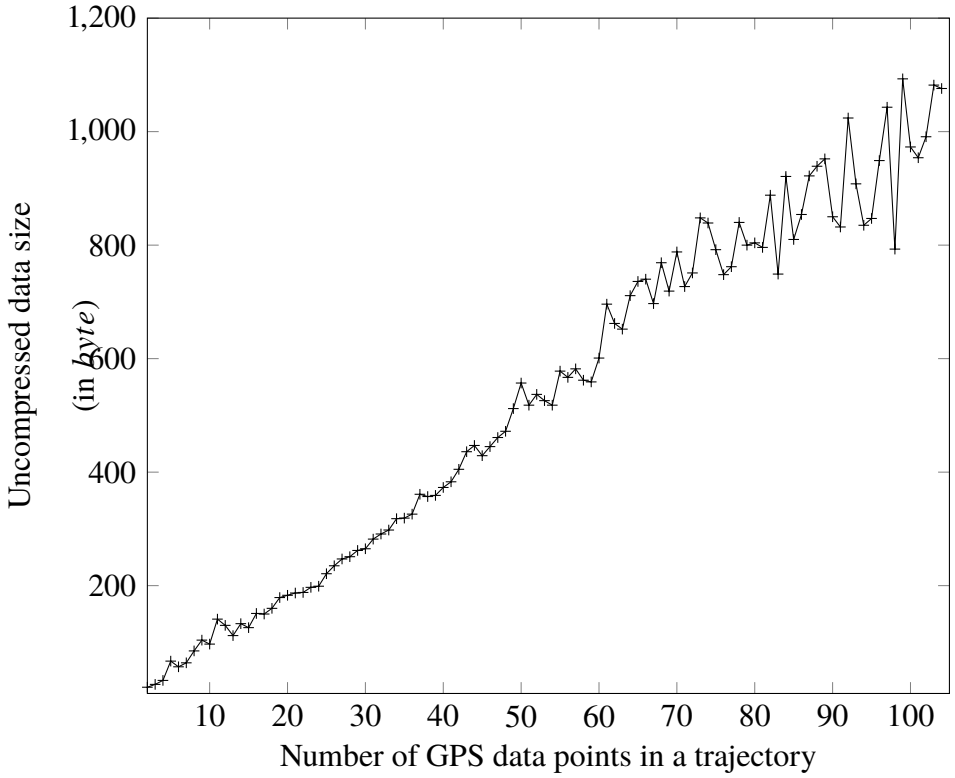


Figure 18. Property of taxi trajectory dataset

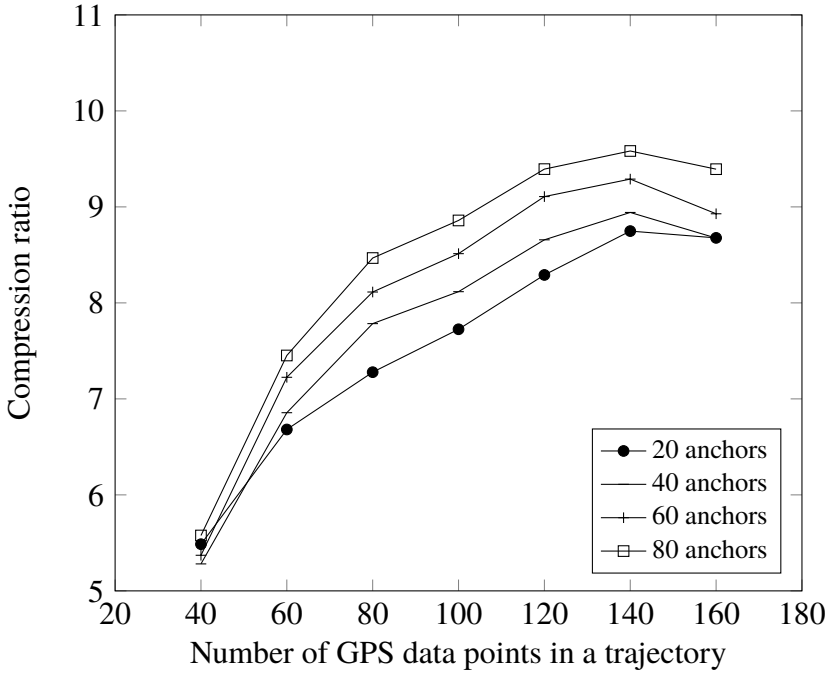


Figure 19. Compression ratio of the encoding without using an ellipse

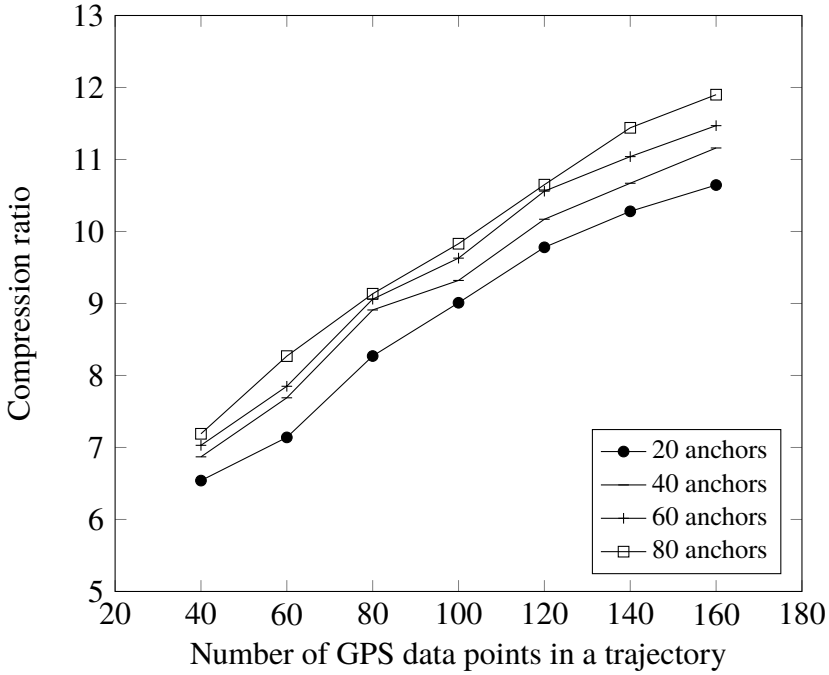


Figure 20. The compression ratio of the proposed DV-Hop based trajectory encoding algorithm (with ellipse) for different trajectory sizes

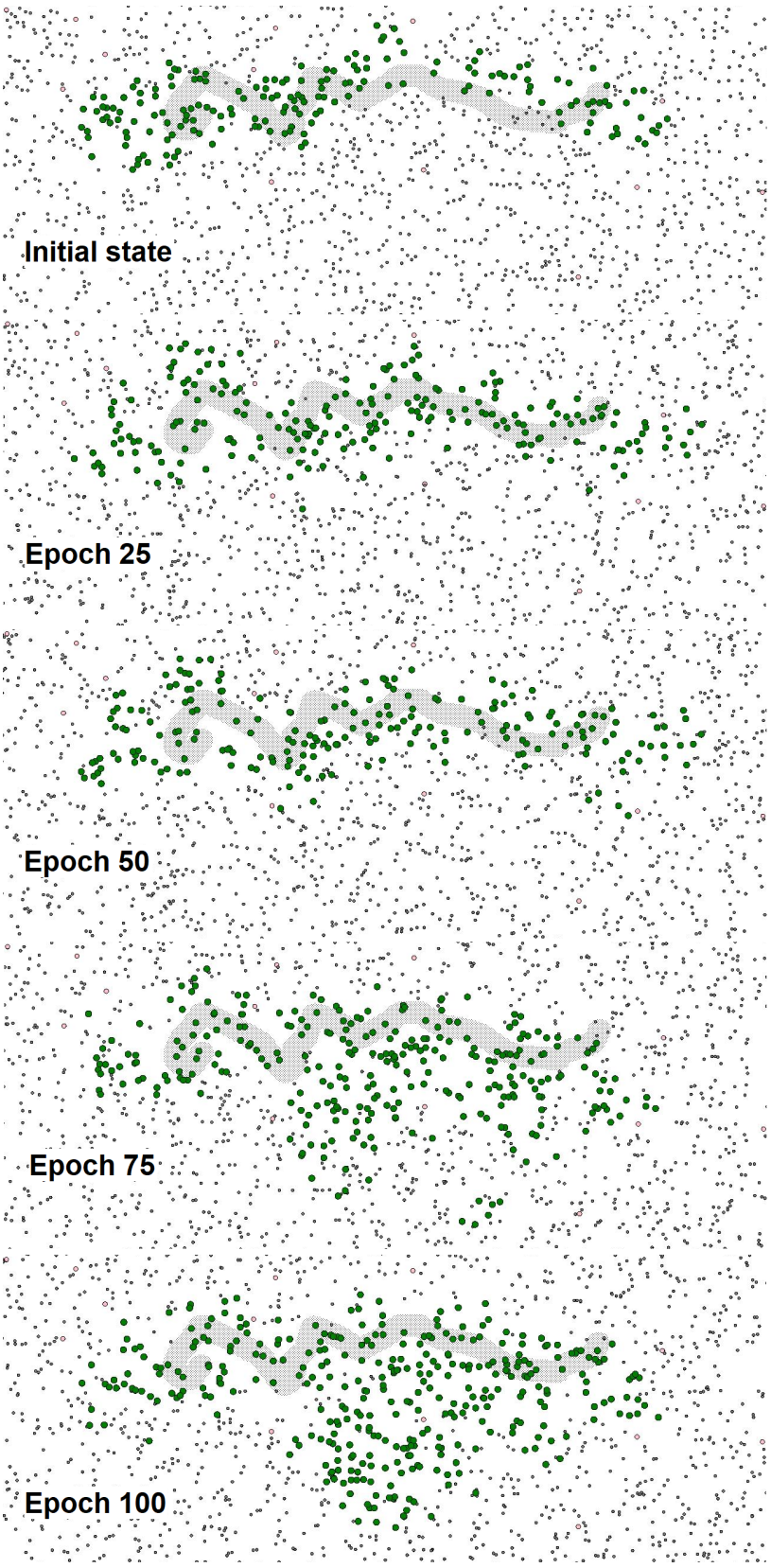


Figure 21. With 50% mobile nodes for each epoch, the changing of coverage for taxi trajectory of Figure 17(a)

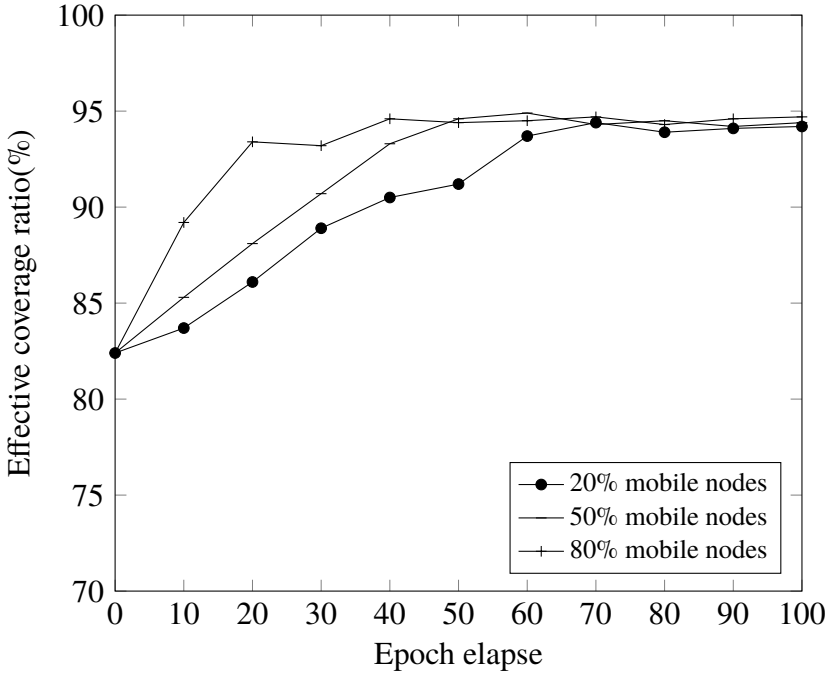


Figure 22. Average correct coverage ratio

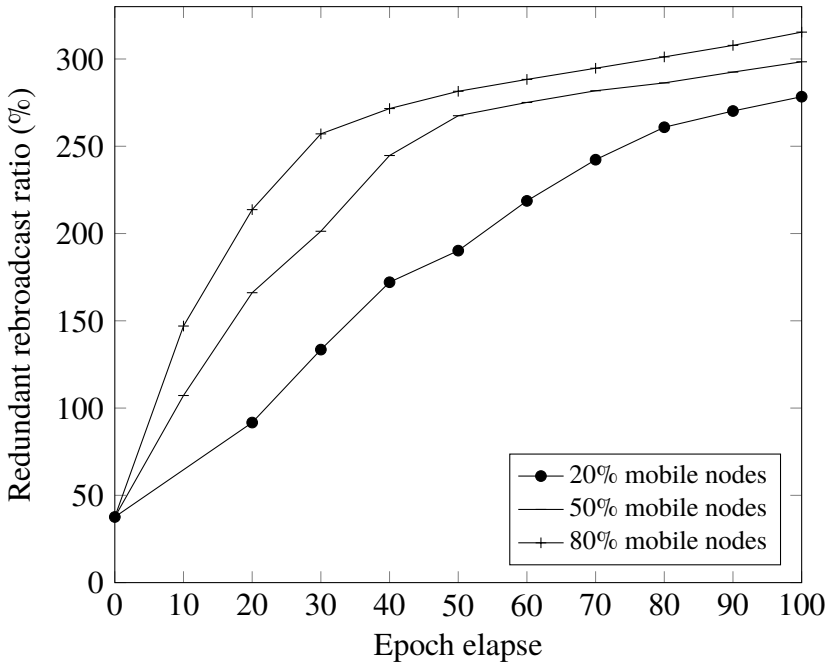


Figure 23. Average redundant rebroadcast ratio

The results show that when the number of anchor nodes increases, the CR also increases. It is more likely to find the closest 'shapes' when the number of anchor nodes is large, which costs fewer iterations to cover the area. Note that when the number of GPS data points increases, the CR is not increased when encoding with 20 or 40 anchor nodes. A possible reason is that when there are fewer anchor nodes, redundant shapes may be selected in the later iterations, which increases the size of the encoded message. For example, in Figure 8-(b)(c), more shapes that overlap with other shapes than that in the case of Figure 8-(a).

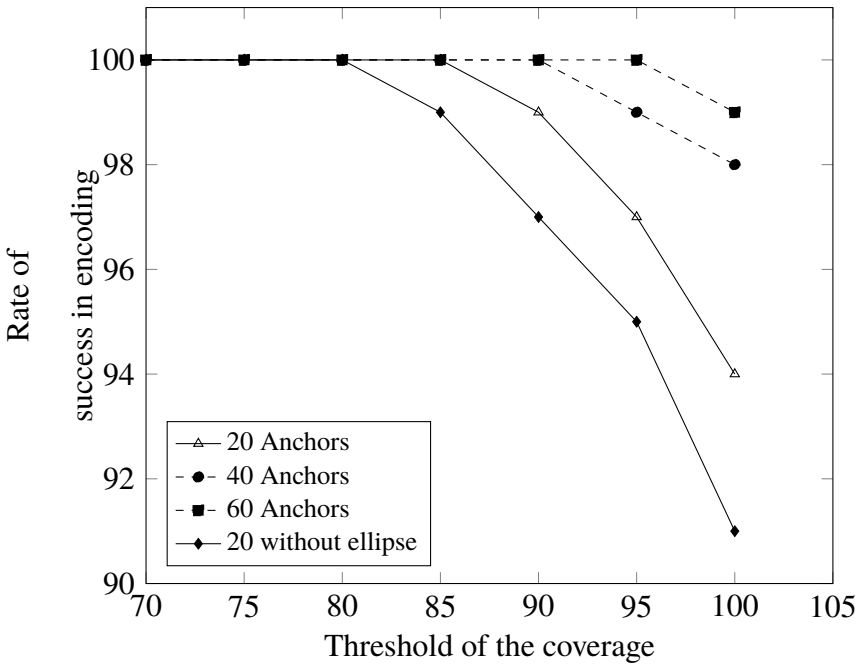


Figure 24. Experiment of successful encoding rate with different number of anchor nodes and coverage threshold

When the mobile nodes change location, they will update the DV-Hop by querying the nearby sensor nodes. As discussed in Section 3.1, the new DV-Hop to anchors will be the minimum hop count of all current neighbors plus one. However, when the neighbors contain the anchor nodes, the new DV-Hop will use the anchor nodes' minimal DV-Hop as

the only reference. A trajectory routing example for a sample trajectory with 66 GPS data in Figure 21 shows the coverage area increases steadily when the network has 50% of the mobile nodes. Figure 22 shows the result of how the correct coverage ratio changes with 20%, 50%, and 80% mobile nodes in a given period (0-100 epoch). In each epoch, the randomly selected sensors will move 1.5 times of its radio distance. The correct coverage ratio will increase due to the increase in the total coverage area. However, the trade-off is the redundant rebroadcast ratio also increases as shown in Figure 23.

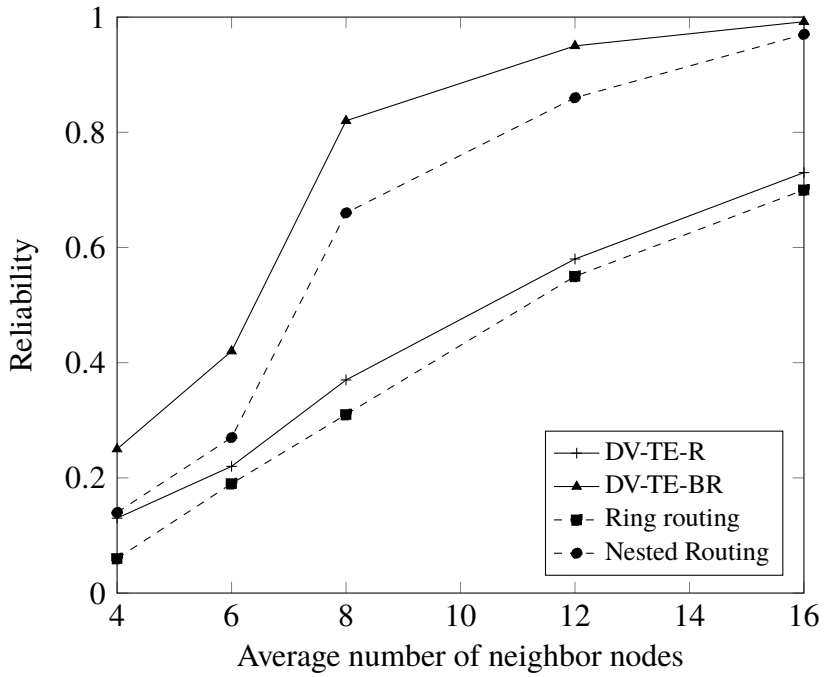


Figure 25. The reliability of DV-TE-R, DV-TE-BR, Ring routing, and Nested routing with different number of neighbors

For the reliability experiment. Since the proposed trajectory encoding algorithm uses basic shapes (ellipse arc and hyperbola segments) to approximate the trajectory, there is a possibility that the encoding will fail when no suitable shape is found. Here, an encoding failure is defined as when the encoding algorithm cannot find any combination of shapes that will cover a certain threshold ratio of TAS, which is defined as the predefined coverage

ratio. For example, when the threshold is set to 85%, the hops constraints represented shapes must overlap with more than 85% of the trajectory area. The lower the threshold, the higher is the encoding successful rate. However, a threshold lower than 85% is not recommended, as the routing reliability (the rate of successfully routing the data request message to the POI) will be affected due to the uncovered gaps between the routing paths. This experiment is to find the relationship between the number of anchor nodes, success rate (the percentage of encoding that does not fail), and coverage threshold. The result in Figure 24 shows that as the number of anchor nodes increasing, the success rate of encoding will increase. However, increasing the coverage threshold will decrease the success rate of encoding. Note that When using ellipse constraints, the success rate is higher than using naive circle constraints when the number of anchor nodes is 20. The reason is the ellipse use three anchor nodes which provide more combination than the circle constraints with two anchor nodes.

The next experiment compares the reliability of the proposed encoding and routing protocol with the ring routing [14] and the nested routing [15], which both enable data routing to a moving sink (mobile edge device that can move in some applications) by relaying the data to a circular area where the nodes know the updated location of the mobile edge device. The area of a nested ring is in the middle of the sensing field, as shown in Figure 8-(c). This experiment uses 50 anchor nodes for the proposed algorithm. The reliability of ring routing is defined as the success rate of generating a ring structure. The reliability of nested routing is the success rate of generating any one of the ring structures within its nested ring structure. The reliability of the proposed DV-Hop based trajectory encoding and routing (DV-TE-R) and its bridge on the edge (DV-TE-BR) adaption is the success rate of generating the encoded message of the trajectory and routing the data request packets to the nodes within POI. The result is shown in Figure 25. When the average number of neighbors of each node is smaller than 8, which is 0.16% of the total number of nodes, both protocols have low reliability. The nested routing protocol has better reliability performance

than ring routing because it has redundant rings. The bridge on the edge adaption improves the reliability of DV-TE-R by relaxing the hop constraints. When the average number of neighboring nodes is greater than 16, which is 0.32% of the total number of nodes, the reliability of DV-TE-BR is greater than 99%. It achieves the best reliability performance compared to ring routing and nested ring routing. In the rest of the experiments, by default, the following experiments use DV-TE-BR.

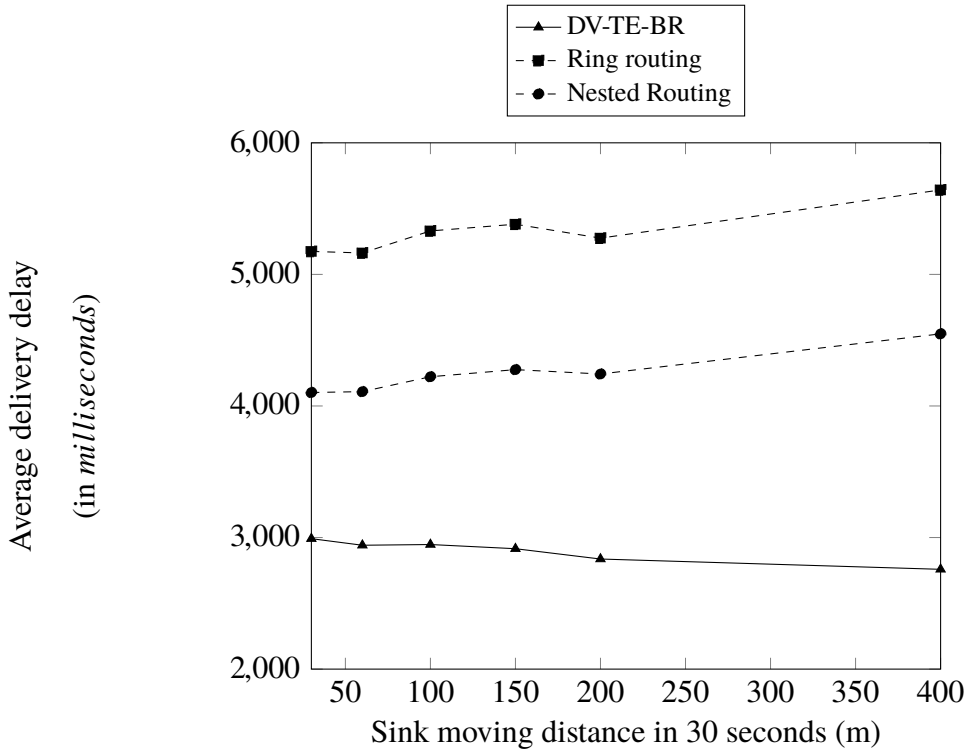


Figure 26. The average delay in data reporting compared with state-of-the-art schemes

The next experiment assumes that mobile edge devices will move randomly with different speeds in the local IoT network with the configuration, as shown in Table 3. The average delay in data reporting is the time when the moving mobile edge devices receive the data minus the time when the source reports the data. The proposed DV-TE-BR fixes the current routing path by letting the mobile edge devices update their locations periodically.

Thus, it cannot always guarantee the shortest reporting path as the ring routing and nested routing methods do. Although the ring routing and nested routing provide the current location of the mobile sink, fetching this information from the ring or nested ring for the source node causes the delay overhead of one round trip to the closest ring. Thus, the delays of ring routing and nested ring routing are still higher than the delay of DV-TE-BR. In addition, the counter-based routing strategy of DV-TE-BR reduces the waiting delay for the low-power listening WSN because the first awaked node could start routing, while ring routing and nested routing have to wait for specific routing nodes within its routing table. Nested routing has a better delay performance than ring routing because its average shortest distance from the source to the rings is shorter than the ring routing. The delay performance of data reporting is shown in Figure 26.

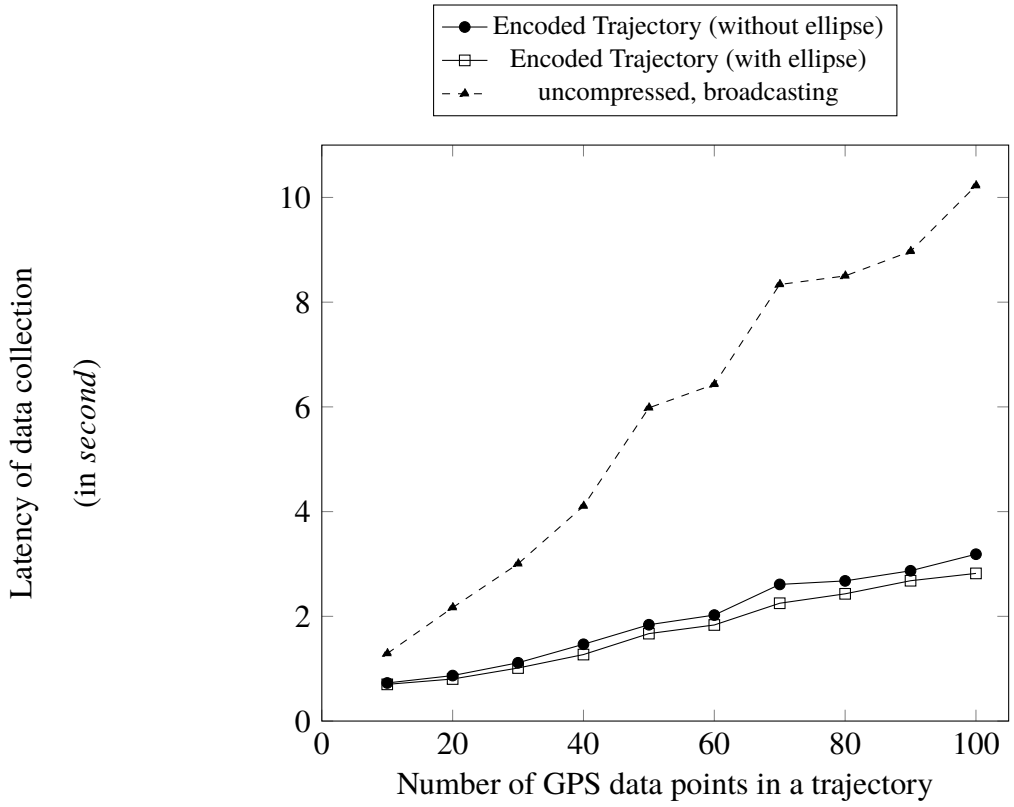
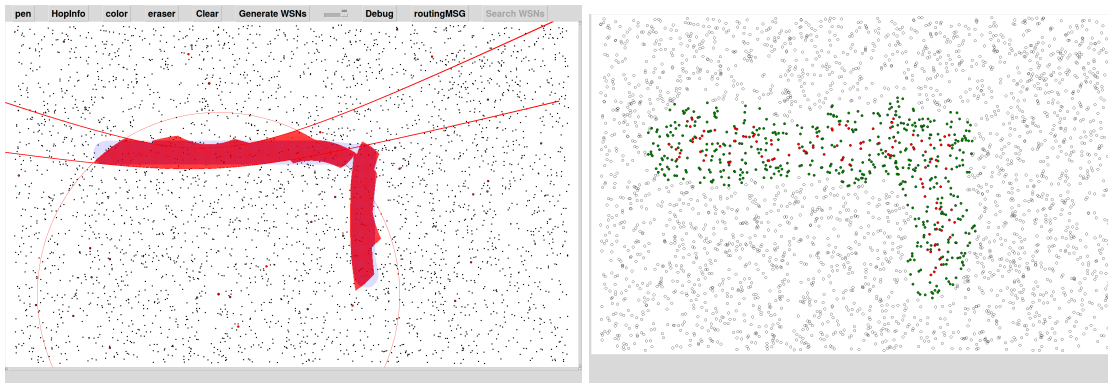


Figure 27. Average delay from starting broadcast request till receiving all the data from the POI

The following latency and energy consumption experiments compare the proposed scheme with the state-of-the-art counter-based broadcasting algorithm. The experimental set-up, which uses the taxi trajectory data, is shown in Table 3. It uses TOSSIM to simulate the routing of data request messages and data reporting packets and visualize the results using Python. Figure 28-(a) shows a sample trajectory being encoded using hops constraints represented shapes, and Figure 28-(b) shows a sample output of the visualized routing result where the red dots are the sensors that forwarded a message and the green dots are sensors that received a message.



(a) Encoded trajectory (b) Visualized routing result

Figure 28. Sample routing example

The latency in collecting the requested data is an important factor in meeting the quality of service. Broadcasting is the fastest way to flood the data request into the whole network. The proposed data collection scheme also broadcasts the data request to the POI. However, as opposed to flooding approaches [7], only the nodes in a hop constraint defined trajectory can rebroadcast. Thus, energy consumption and bandwidth usage are minimized. Figure 27 compares the latency of the local edge devices receiving all the sensing data of the POI from the local WSN, which are working under low power listening (LPL) mode with a 660 ms sleeping and waking period. All local edge devices

will broadcast the data request messages. The sensor nodes that receive the data request packets will be awake and send the data back to the nearest edge devices if they are at the POI. The experimental result shows that transmitting encoded data request messages could reduce latency because broadcasting a compressed message requires fewer packets than an uncompressed message which includes IDs of all the sensor nodes residing within the POI. The proposed scheme, which encodes trajectories with ellipse-circle constraint, achieves better latency performance than the trajectory encoding protocol with only circle-circle constraints due to its higher compression ratio.

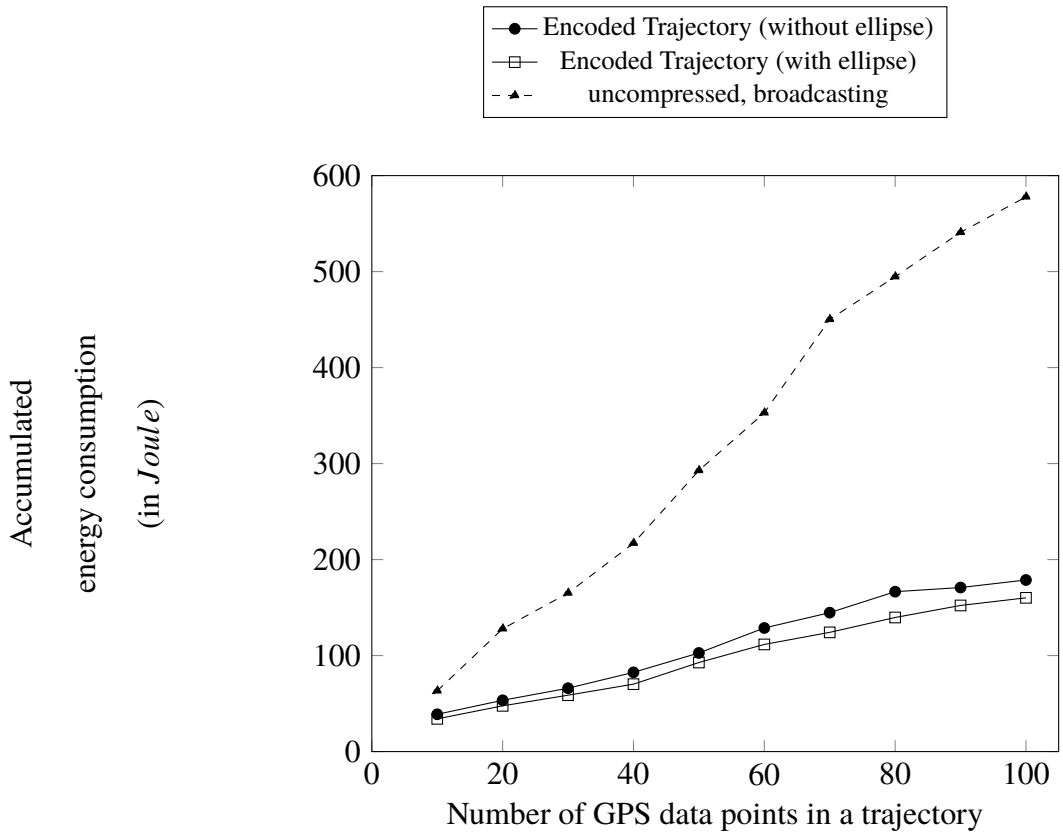


Figure 29. Accumulated energy consumption in fetching the data from the POI

The energy consumption experiment is simulated with powerTOSSIM-Z, which is an energy simulation tool for wireless sensors. It uses the micaZ energy model and can measure energy consumption at the packet level. The result in Figure 29 shows that the proposed data collection scheme consumes less energy than the broadcasting approach. The proposed scheme, which encodes trajectories with ellipse-circle constraints, achieves better energy performance than the trajectory encoding protocol with only circle-circle constraints due to the higher compression ratio. The last experiment is to compare the average number of rebroadcasting nodes of the proposed DV-hop based trajectory encoding and routing scheme (DV-TE-BR) verses the state-of-the-art counter-based broadcasting [7] for each single data request packet. The result shows that the proposed DV-TE-BR scheme reduces the number of redundant rebroadcasting packets (142 vs. 2491) by 94% and thus, saves bandwidth usage in the WSN.

6. CONCLUSION AND FUTURE WORK

The proposed trajectory encoding and data collection algorithms for IoT applications have improved energy efficiency, reduced latency, and achieved reliable performance when fetching data from the POI in the local fog network without using GPS coordinates. In addition, with the use of virtual coordinates, location anonymity is achieved for the source, sink, and intermediate nodes in the routing path, as only the secure server in the local fog knows the anchor nodes' locations. Besides, the use of ellipse and hyperbola constraints increase the encoding accuracy and compression ratio. In the future, the plan is to solve the real-time event detection problem in multi-hop IoT network using the proposed approach and the conditional random field [36]. The plan is also explore to extend the proposed scheme for under water WSN.

REFERENCES

[1] Weisong Shi and Schahram Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

[2] Shanhe Yi, Zijiang Hao, Zhengrui Qin, and Qun Li. Fog computing: Platform and applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*. IEEE, 2015.

[3] Yingzi Wang, Nicholas Jing Yuan, Defu Lian, Linli Xu, Xing Xie, Enhong Chen, and Yong Rui. Regularity and conformity: Location prediction using heterogeneous mobility data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1275–1284. ACM, 2015.

[4] Jing Wang, Jian Tang, Guoliang Xue, and Dejun Yang. Towards energy-efficient task scheduling on smartphones in mobile crowd sensing systems. *Computer Networks*, 115:100–109, 2017.

[5] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless networks*, 8(2-3):153–167, 2002.

[6] Chien Chen, Chin-Kai Hsu, and Hsien-Kang Wang. A distance-aware counter-based broadcast scheme for wireless ad hoc networks. In *Military Communications Conference, MILCOM*. IEEE, 2005.

[7] Ji-Young Jung and Dong-Yoon Seo. Counter-based broadcast scheme considering reachability, network density, and energy efficiency for wireless sensor networks. *Sensors*, 18(1):120, 2018.

[8] Kok-Poh Ng, Charalampos Tsimenidis, and Wai Lok Woo. C-sync: Counter-based synchronization for duty-cycled wireless sensor networks. *Ad Hoc Networks*, 61:51–64, 2017.

[9] Murat Yuksel, Ritesh Pradhan, and Shivkumar Kalyanaraman. An implementation framework for trajectory-based routing in ad hoc networks. *Ad Hoc Networks*, 4(1):125–137, 2006.

[10] Houda Labiod, Nedal Ababneh, and Miguel García de la Fuente. An efficient scalable trajectory based forwarding scheme for vanets. In *Advanced Information Networking and Applications (AINA), 24th IEEE International Conference on*, pages 600–606. IEEE, 2010.

[11] Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 96–108. ACM, 2003.

[12] Shrawan Kumar and DK Lobiyal. Novel dv-hop localization algorithm for wireless sensor networks. *Telecommunication Systems*, 2017.

[13] Badri Nath and Dragoş Niculescu. Routing on a curve. *ACM SIGCOMM Computer Communication Review*, 33(1):155–160, 2003.

[14] Can Tunca, Sinan Isik, Mehmet Yunus Donmez, and Cem Ersoy. Ring routing: An energy-efficient routing protocol for wireless sensor networks with a mobile sink. *IEEE Transactions on Mobile Computing*, 14(9):1947–1960, 2015.

[15] Ramin Yarinezhad. Reducing delay and prolonging the lifetime of wireless sensor network using efficient routing protocol based on mobile sink and virtual infrastructure. *Ad Hoc Networks*, 84, 2019.

[16] Suraj Sharma, Deepak Puthal, Sabah Tazeen, Mukesh Prasad, and Albert Y Zomaya. Msgr: A mode-switched grid-based sustainable routing protocol for wireless sensor networks. *IEEE Access*, 5:19864–19875, 2017.

[17] Levente Buttyán and Péter Schaffer. Position-based aggregator node election in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 6(1):679205, 2010.

[18] Omar Banimelhem and Samer Khasawneh. Gmcar: Grid-based multipath with congestion avoidance routing protocol in wireless sensor networks. *Ad Hoc Networks*, 10(7):1346–1361, 2012.

[19] Yuan-Po Chi and Hsung-Pin Chang. An energy-aware grid-based routing scheme for wireless sensor networks. *Telecommunication Systems*, 54(4):405–415, 2013.

[20] Abdul Waheed Khan, Javed Iqbal Bangash, Adnan Ahmed, and Abdul Hanan Abdullah. Qdvgdd: Query-driven virtual grid based data dissemination for wireless sensor networks using single mobile sink. *Wireless Networks*, (1):241–253, 2019.

[21] Kai Chen, Zhong-hua Wang, Mei Lin, and Min Yu. An improved dv-hop localization algorithm for wireless sensor networks. 2010.

[22] Shrawan Kumar and DK Lobiyal. An advanced dv-hop localization algorithm for wireless sensor networks. *Wireless personal communications*, 71(2):1365–1385, 2013.

[23] Penghong Wang, Jianrou Huang, Zhihua Cui, Liping Xie, and Jinjun Chen. A gaussian error correction multi-objective positioning model with nsga-ii. *Concurrency and Computation: Practice and Experience*, 32(5):e5464, 2020.

[24] Xingjuan Cai, Penghong Wang, Lei Du, Zhihua Cui, Wensheng Zhang, and Jinjun Chen. Multi-objective three-dimensional dv-hop localization algorithm with nsga-ii. *IEEE Sensors Journal*, 19(21):10003–10015, 2019.

[25] Mamoun F Al-Mistarihi, Islam M Tanash, Fedaa S Yaseen, and Khalid A Darabkh. Protecting source location privacy in a clustered wireless sensor networks against local eavesdroppers. *Mobile Networks and Applications*, 25(1):42–54, 2020.

[26] Hao Wang, Guangjie Han, Chunsheng Zhu, Sammy Chan, and Wenbo Zhang. Tcslp: A trace cost based source location privacy protection scheme in wsns for smart cities. *Future Generation Computer Systems*, 107:965–974, 2020.

[27] Anfeng Liu, Xiao Liu, Zhipeng Tang, Laurence T Yang, and Zili Shao. Preserving smart sink-location privacy with delay guaranteed routing scheme for wsns. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(3):1–25, 2017.

[28] Xiaofei Cao. Serial port reader and writer for the android gateway. https://github.com/cxfcdcpu/gateway.

[29] Xiaofei Cao and Sanjay Madria. Efficient geospatial data collection in iot networks for mobile edge computing. In *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, pages 1–10. IEEE, 2019.

[30] Timothy M Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & Computational Geometry*, 16(4):361–368, 1996.

[31] Gary B Hughes and Mohcine Chraibi. Calculating ellipse overlap areas. *Computing and visualization in science*, 15(5):291–301, 2012.

[32] Taxi trajectory data. www.kaggle.com/crailtap/taxi-trajectory.

[33] TelosB Datasheet. Crossbow Inc. Downloaded from memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf.

[34] MicaZ. Crossbow Inc. Downloaded from memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf.

[35] Michael Buettner, Gary V Yee, Eric Anderson, and Richard Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320, 2006.

[36] Huihsin Tseng, Pi-Chuan Chang, Galen Andrew, Dan Jurafsky, and Christopher D Manning. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, 2005.

V. AN EFFICIENT MOVING OBJECT TRACKING FRAMEWORK FOR WSNS USING SEQUENCE-TO-SEQUENCE LEARNING MODEL

Xiaofei Cao and Sanjay Madria

ABSTRACT

Wireless sensors can detect an object from the light it reflects, the noise it causes, or the gas molecules it disseminates. However, tracking a moving object requires the wireless sensors to perform high-frequency sensing and data transmission which consume much more energy. To save energy and prolong the lifetime of wireless sensor networks while tracking a moving object effectively, this paper proposes a framework that predicts the trajectory of the moving object using a Sequence-to-Sequence learning (Seq2Seq) model and only wakes-up the sensors that fall within the predicted trajectory of the moving object with a specially designed control packet. The framework uses DV-Hop (distance vector of hops to anchors) as the virtual coordinate that eliminates the dependency of using GPS to locate the sensors to be invoked for tracking the moving object. The framework translates the object's moving trajectory to a sequence of cascaded hyperbolas and encodes the hyperbolas with DV-Hop constraints. A control packet containing these constraints forbid sensors not in the trajectory to rebroadcast, and awake/sleep signals that control the sensors' action. The proposed Seq2Seq model predicts the target's next trajectory directly and outputs a control message that could route along the predicted trajectory. In comparison to predicting the target's trajectory then encoding the trajectory using geometric objects such as hyperbola, the proposed Seq2Seq model reduces the computation time of encoding geospatial trajectory. Also, the proposed framework preserves the location anonymity by only transmitting the hop's information instead of GPS values. The performance comparisons with the existing methods show an improvement in energy-saving and control message routing delay.

1. INTRODUCTION

Wireless sensor networks (WSNs) have been of considerable interest to the research community in recent years because of their use in many real-world applications. Among many, moving object tracking is one of the important applications of wireless sensor networks and is widely used in both civil, research, agriculture, and military applications. For example, the military can use wireless sensor networks to track military vehicles [1]. The smart city uses WSNs to fetch the trajectory of every vehicle and use the information to guide other commuters or detect abnormal driving behavior [2]. In these target tracking applications, different sensors monitoring the targets continuously during their mobility and thus faces several challenges. First, not all sensors contribute equally to target tracking. Unnecessary sensing by the sensors which are off the targets could cause excessive energy consumption. Second, detecting targets with long sensing period in a WSN could be a challenge due to limited battery power. Given low power listening (LPL)[3] and other energy-efficient MAC protocols, most of the real-world WSN applications can put the sensors into the wake/sleep cycle, and only wake up some of them for a small period for sensing and communication. However, in LPL mode, sensors may face a high risk of losing the target because of the low sensing and communication frequency. Third, for some military applications, sensors also need to prevent being detected by other enemy targets. For example, once the enemy targets detect the presence of nearby sensors through the radio signals, they may perform some adverse actions which may increase the detection difficulty. The enemy targets may also start jamming the radio transmissions in the area [4]. Fourth, obtaining the enemy targets' location is also a challenge. The sensors detecting the target need to estimate the target's location. However, due to the low cost of the WSNs, most of the sensors have no GPS modules. Though virtual coordinates [5],[6] and [7] could calculate the approximate location of sensors, fetching all the hop information of the sensors in a large scale WSN and calculating the location centrally can cause excessive

energy consumption. Lastly, location anonymity is also a challenge as the sensors are vulnerable to be eavesdropped and can be compromised, thus directly transmitting the location information is not secure.

The prediction-based methods [8], [9] are used to control the sensor states (active, sleep) in the next location of the moving object based on historical data. Cluster-based tracking approaches [10], [11] divide the WSNs into small subsets called clusters. The clustering architecture sends sleep/awake schedule messages from the cluster head to their member nodes with less communication overhead to reduce energy consumption. The grid-based target localization and tracking approaches [12] estimate the target's location with multiple nodes in the target's boundary grid to increase the localization precision. However, they are still challenges that need to be tackled. The cluster-based control message dissemination approaches introduce radio communication overhead for electing the cluster head and in maintaining the cluster topology. Also, every node in the network has to know its location, which increases costs for either the precise deployment or the extra GPS module. Furthermore, in military applications, the adversary target could detect the broadcasting of the cluster head which adds the risk of losing the target. Although the linear prediction model could predict the target's movement well, the centralized cluster-based approaches can't deliver the wake-up (for tracking) and reset messages (for putting sensors back to low power listening (LPL) mode) efficiently. The grid-based tracking approaches [12] are only a transformer of the cluster-based approaches with additional restrictions. Furthermore, when the target moves near the boundary of the cluster and the grid, the system will spend much more effort on localization and tracking because two or more clusters/grids are now participating in the tracking.

Our study will therefore focus on solving the following challenges. First, to locate and track the moving target without using any GPS-based sensors. Second, to deliver the wake-up and reset message to the area around the target's path quickly and energy efficiently. Last but not the least, to preserve the location anonymity of the sensors tracking objects.

To achieve the above goals, we proposed a Seq2Seq model that takes the target's previous estimated locations as input and outputs a geometric constraint (will be discussed in Section 2.4) that allows only the sensors within the predicted trajectory to wake up, detect and track the target and report the results to the nearest local edge server. A set of these constraints creates a path constraint that covers all the areas of the target's previous and predicted trajectory.

The proposed framework directly predicts the routing constraints that cover the target's predicted trajectory. It is much faster than predicting the sequence of future target's location first and then encoding the predicted trajectory. Compared to the cluster-based target tracking approach, the geometric constraints based routing protocol reduces the overhead of cluster generation and maintenance. The location anonymity is preserved as no GPS data are used and transmitted. The performance evaluations show the effectiveness of the proposed scheme over other competitive schemes.

Organization of the paper: In Section 2, we discuss some state-of-the-art energy-efficient object tracking frameworks, trajectory prediction algorithms, and DV-hop localization methods. In Section 3, we formulate the problem and provides the assumption and system model of the proposed framework. In Section 4, we propose an optimization approach using a Sequence to Sequence learning model to speed up the computation. In Section 7, we present the results of our performance evaluation. In Section 8, we make concluding remarks and discuss future work.

2. RELATED WORK

2.1. TRAJECTORY PREDICTION IN WSNS

In a moving object tracking problem, the key objective is the dynamic sensor tracking schedule to predict the trajectory that ensures the real-time performance of object detection and tracking. When WSNs operate in low-power-listening (LPL) mode, the radio

communication latency equals half of the duty cycle times hops counts as shown in the following equation. $Delay^{rcv}_{send} = \frac{H^{rcv}_{send} \times T_{clc}}{2}$ where $Delay^{rcv}_{send}$ is the routing delay in sending a packet from a sender to a receiver, H^{rcv}_{send} is the hop counts from a sender to a receiver, and T_{clc} is the average duty cycle of the current scheduling protocol. The prediction-based methods are used to predict the location of the mobile object after $Delay^{rcv}_{send}$ time based on historical data. Therefore, the sensor states (active, sleep) can be prepared before the target enters/leaves the area. Linear prediction is a simple prediction approach, which depends only on the previous location of the target [13]. However, linear prediction suffers from low prediction accuracy.

To improve the prediction accuracy, particle filter [14] and Kalman filter [15], [16], [17] based prediction frameworks have been proposed. Kalman filter is a linear algorithm that exploits a series of data observed overtime to boost the prediction's precision. In paper [18], the authors proposed a Kalman filter based generalized regression neural network that not only reduced the prediction error but also improved the prediction speed by combining the Kalman filter with neural networks. The particle filter based frameworks like [19] [20] [21] are also widely used in the target tracking as they are suitable for nonlinear systems. There are also works like [22] that combines both Kalman filter and Particle filter based frameworks to get reliable location prediction for real-world applications. Although the previous prediction based target tracking approaches could achieve good prediction speed and precision, they still rely on the GPS data or RSSI values, and can not directly generate the control message which can directly control the local sensors. The predicted trajectory needs to be processed by the server that has the knowledge of all the sensors' locations which has a higher risk of leaking the location privacy.

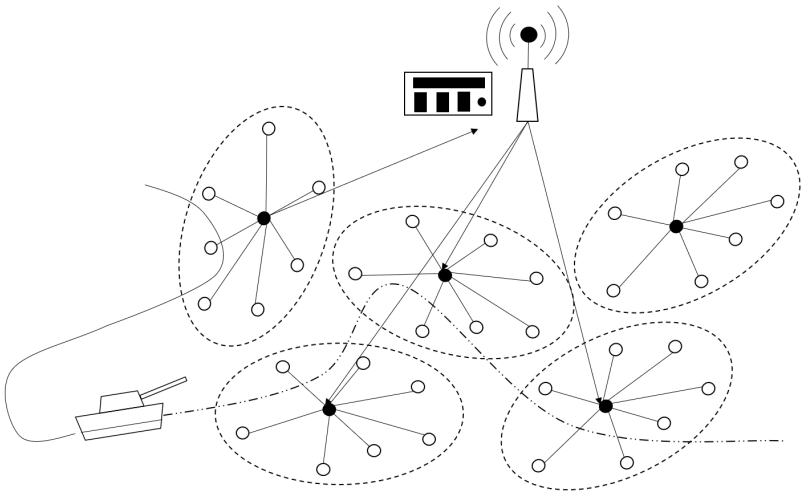


Figure 1. Cluster-based Object tracking protocol

2.2. CLUSTER-BASED OBJECT TRACKING ALGORITHMS

Cluster-based object tracking protocols are so popular that some researchers classified the previous works into only two groups: cluster-based and non-cluster-based. It is the most realistic solution that could control the message flow in large scale WSNs. In cluster-based protocols, cluster heads, which are selected by different cluster algorithms, are responsible for collecting information from the nodes in their cluster, communicating with sinks, and propagating the control messages to their cluster members. In this way, a large scale WSN is simplified to a small sink-cluster heads network with many small cluster head - slave sensors sub-networks.

For object tracking applications, once slave sensors in any cluster detect the object they report the target's information to their cluster head. The cluster head then routes the information to the sink or the local mobile edge server. For saving energy, as we have discussed in Section 2.1, the server predicts the target's trajectory and sends the control messages to the cluster heads which reside on the target's trajectory. Those cluster heads then propagate the active/sleep messages to all or some members when the target enters/leaves. For example, in Figure 1, the sink of a cluster-based WSN collects the target's current

location from a cluster-head. After predicting the target's future trajectory, the sink sends control messages to the cluster heads that reside on the target's future path. The cluster-head then controls the related sensors for detecting the target.

The drawback of the cluster-based object tracking protocols is the overhead of generating the dynamic cluster [23] and maintaining the clusters. Also, for different applications, all the nodes in the WSN need to tune their program to meet specific routing and clustering requirements. However, it is usually not practical for large scale WSNs.

Considering the above drawbacks of the cluster-based object tracking protocols, we choose to use the DV-Hop (which stands for distance vector of hops) based packet routing protocol [24] that decouples the data plane (network layer) and the control plane of the IoT network. Like software-defined networks, the DV-Hop based routing rules are encapsulated in each routing packet. So different applications could share the same WSN by just creating their own routing rules. The DV-Hop based routing protocol will be elaborated in Section 2.4.

2.3. COUNTER-BASED BROADCAST

Broadcasting is the fastest way to flood a message into the whole WSN. However, limited bandwidth causes a delay in broadcasting a sequence of messages into the network. After a node receives a given packet, the counter-based broadcasting schemes [25][26][27] require a node to wait for a short period to listen to its neighbors and count how many times the given packet has been rebroadcasted. If the broadcast count of the given packet reaches the predefined threshold, it will drop the packet. Otherwise, the receiving sensor node will rebroadcast the packet. This procedure will repeate in the whole broadcasting period until all the nodes in the network receive the data packet. Comparing to the naive broadcast approach, for counter-based broadcast, only a few of the nodes in the network will rebroadcast the given packet which saves bandwidth and thus, alleviates the congestion.

2.4. DV-HOP BASED PACKET ROUTING PROTOCOL

The previous research work [24] had devised a data collection and routing approach based on DV-Hop virtual coordinate that could deliver packets through any trajectory. DV-Hop uses the anchor nodes and the vector of minimum hop distance to the anchor nodes to estimate the distance between nodes where anchor nodes are the pre-selected nodes whose locations are known. Combining anchor nodes and their hop counts relationship follows certain geometric shapes like an arc of a circle, a wing of a hyperbola, and a segment of an ellipse. A trajectory then can be approximated with a set of these geometric constraints. Also, as all the sensors in the network have the DV-Hop table, which contains the hops counts to nearby anchor nodes, they can decide whether they are within the trajectory or not by calculating if any of these geometry constraints satisfy their DV-hop tables. Intuitively, increasing the number of anchor nodes will have a better precision of trajectory approximation. However, the computational time complexity will also have a cubic growth. We take the advantage of these cutting edge DV-Hop based routing approaches but reduce the computational complexity by proposing a Seq2Seq model to derive the relationship between the target's trajectory and the geometric constraints that cover that trajectory.

3. PROPOSED OBJECT TRACKING FRAMEWORK

In a multi-hop wireless sensor network, a user wants to detect and track one or more hidden mobile objects/events, called target, in the region of interest. The sensors of the WSN can detect the target within the detection range based on the sensing values. The proposed object tracking framework predicts the trajectory of the moving target and directly generates the control message to activate the sensors within the trajectory to perform the active sensing for fast target detection when the target approaches. It contains two parts. First is the the trajectory prediction and encoding. The second is control message dissemination within the trajectory.

3.1. ASSUMPTION

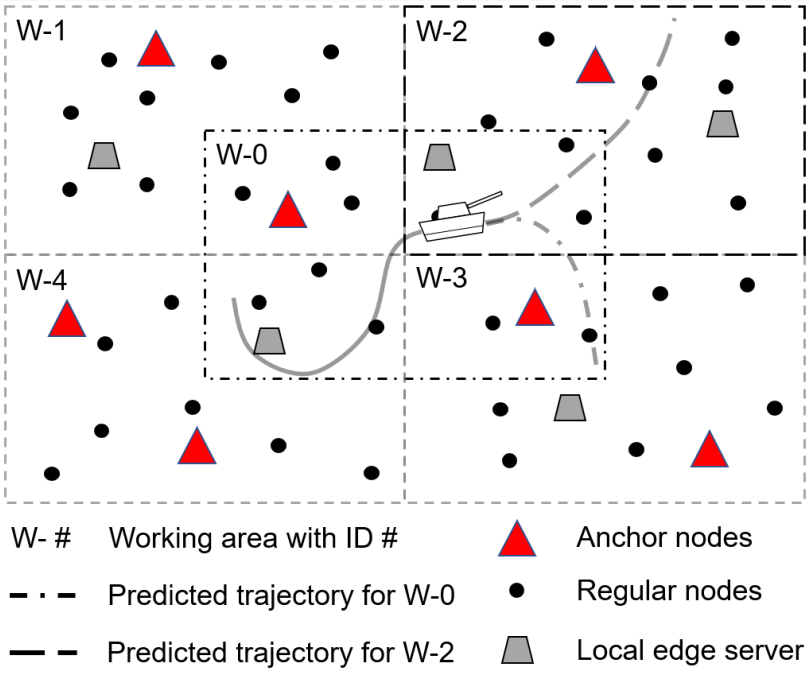
The proposed framework has the following assumptions. In the WSN, there are two types of sensors, the anchor nodes and the regular sensors. The regular sensors are deployed randomly in the network and can reach any other node within a hop count h_{Max}. They don't have GPS nor they know their locations. The anchor nodes are randomly deployed which have identical computation and storage power as other nodes in the network. However, the network owner has their location information. In the deployment time, the anchors will flood the beacon signals within a given hop counts ($h_{floodLimit}$) to help the regular sensors initialize the DV-Hop values to these anchors. No routing table or any other information is stored.

After the deployment, the sensors will keep their radio in sleep states to save energy and avoid being detected by the target. Periodically passive detection and periodically sleep/wake are also performed by the sensors to prolong the battery life. The proposed framework also assumes that the sensors' object detecting area is much smaller than the radio communication range. When a sensor detects a target, it will broadcast the target detection packet ($TDPkt$) to the nearest anchor nodes through the decreasing DV-Hop gradient path (discussed in Section 3.3) in a counter-based broadcast fashion. $TDPkt$ includes the timestamp, which indicates when the target is detected. It also contains the DV-Hop information of the sensor that detects the target.

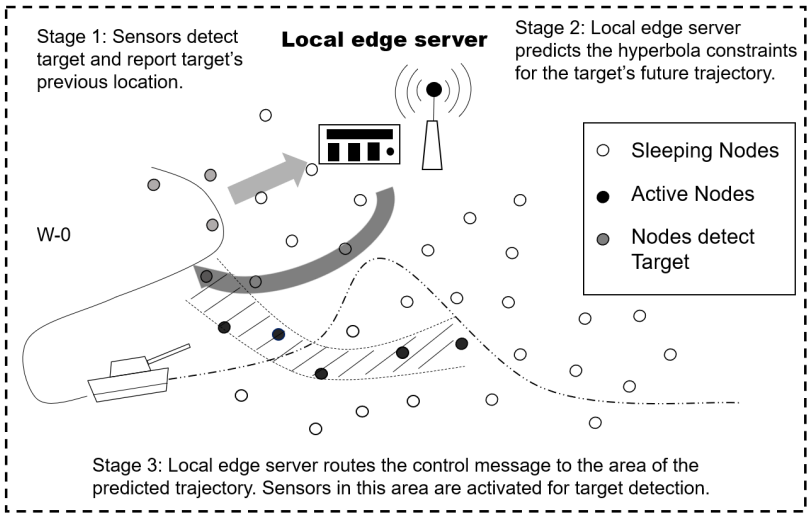
3.2. SYSTEM OVERVIEW

The proposed system includes the sensor plane and the control plane. The sensor plane is consists of low power devices like wireless sensors, RFID, and wireless actuators. The control plane contains the remote cloud, the local edge server, or mobile end users. All the devices in the control plane have more computational power and energy. However,

they may belong and affiliate to third parties, temporary contractors, or other authorized clients. As the sensors may use different radio protocols than the device in the control plane, gateways are used to bridge the devices between these two planes.



(a) A larget WSN with five overlapped working area in the middle



(b) A zoom in example of working area 0

Figure 2. System Overview

As shown in Figure 2 (a), a WSN is divided into multiple overlapped working areas. Anchor nodes (red triangle) are randomly deployed in these working areas. As we have discussed in the assumption, anchor nodes will flood the beacon messages to nearby regular sensor nodes(black dots). The regular sensors then create a DV-Hop table for each nearby anchor nodes. Local edge servers (trapezoid) act as sink nodes that collect and control the sensor nodes through the wireless gateway. They gather the target's previous location information and predict the routing constraints that cover the target's future trajectory with a Seq2Seq learning model which will be discussed in Section 4.

Figure 2 (b) is a zoom-in example of one working area of Figure 2 (a). It shows three stages of the proposed target tracking framework as follows: The first stage is called the local activation stage which is discussed in Section 3.3. When a sensor detects a target, it immediately broadcasts its current timestamp and DV-Hop information toward the nearest local edge server. The broadcast follows the hop gradient decreasing and limited broadcast count rules to reduce the risk of being detected by the target as well as help in saving energy. All the nodes receiving the broadcast message start actively sensing the nearby field for a short period $T_{LocalActivate}$.

The second stage is to predict the trajectory constraints that cover the target's future location using the proposed Sequence to Sequence (Seq2Seq) learning model which is elaborated in Section 3.4 and Section 4. The Seq2Seq model, pre-loaded in the local edge servers, uses a sequence of DV-Hop list as the input and output control packets directly. Therefore, the local edge servers can control the sensor network without the location information of the anchor nodes. Also, even when the local edge servers are compromised, the adversary won't be able to recover the target's tracking trajectory.

The third stage connects the local edge server to the sensor node that most recently detected the moving object by generating a set of path constraints which has been discussed in Section 5. These path constraints are trained with a Seq2Seq model similar to the one

that encodes the predicted target trajectory. The only difference is that the path encoding requires an input of DV-Hop of start and end points while the trajectory prediction requires a list of target's DV-Hop associated with the previous timestamp, S_{pre}.

These two types of Seq2Seq models are trained in a trusted remote cloud that has the location information of the anchors for each working area. After training, the remote cloud will assign the models of each working area to the local edge server respectively. Sensors that detect the target will broadcast the target's location to all the nearby local edge servers using a gradient broadcast which has been discussed in Section 3.3. So if the target moves over the overlapped working area, the local edge servers of all the overlapped working areas will track the packet simultaneously. When the target moves and the prediction is updated, the local edge server will send a reset message through the old path to force the sensors to fall asleep and send an activate message to wake up the sensors in the newly predicted target's trajectory.

3.3. GRADIENT-BASED BROADCAST

When a node in a WSN broadcasts a message, it only contains the packet's unique ID and the packet's rebroadcast time. Then all the receivers rebroadcast the message with the incremented rebroadcast time. If all the sensors only rebroadcast the packets with lower rebroadcast times, then, after the flooding stops, all the nodes in the network would have the minimal hops counts (referred to as gradient in some papers [28]) to that initial broadcasting sensors. Then let any node other than the initial node to broadcast a message to the initial node that has a zero gradient. The shortest path will be the path that has a decreasing gradient to the initial nodes. The gradient is zero for the initial node. The gradient-based broadcast could always route a message to the zero gradient point in the shortest hop path. However, the redundant rebroadcast is inevitable for the gradient-based broadcast. To mitigate the rebroadcast, the counter-based broadcast [26][27] can be used.

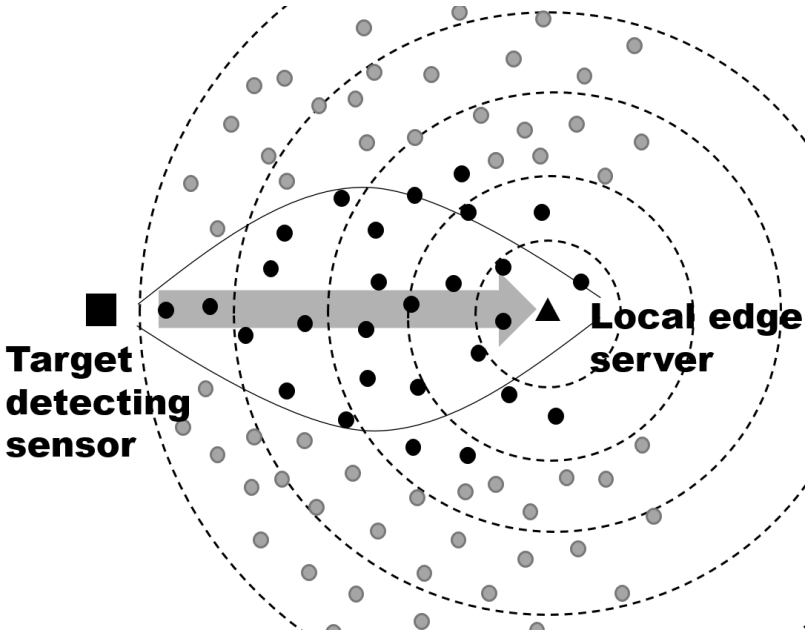


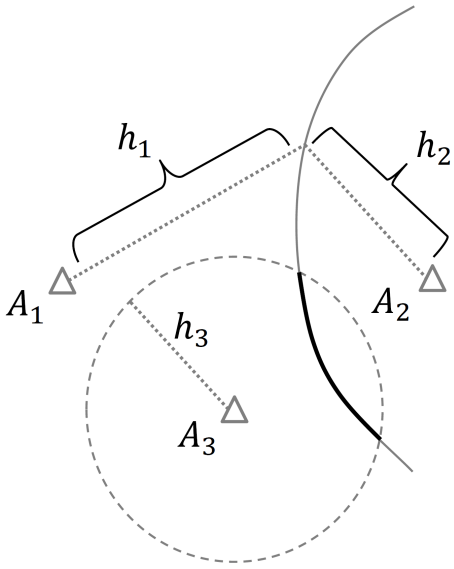
Figure 3. Sensor that detect the target report to the local edge server through a hop gradient decreasing path.

In the proposed framework, when every local edge server joins the WSN, it floods a beacon packet to the nearby sensors. Those sensors receiving a beacon packet create a DV-Hop for these local edge servers. When the local edge server leaves the network, it will also broadcast a beacon packet to inform all the nearby sensors. Thus, the sensors will delete the respective DV-Hop entry for that local edge server. When a sensor detects a target, as shown in Figure 3, it immediately broadcast the current timestamp, the hop count from itself to the local edge server, and its DV-Hop of anchors to the nearest local edge server.

3.4. HYPERBOLA CONSTRAINTS BASED CONTROL-MESSAGE ROUTING

As we have discussed in Section 1, routing of the control messages to the desired trajectory is a challenge because the local edge servers have no location information of all the sensor nodes in the large scale WSN. In the proposed framework, the sensors won't use a static routing table (as it may only good for one application), nor a dynamic routing table

(as this will have high latency and energy cost). Modified DV-Hop constraints based routing protocol [24], which only uses the hyperbola constraints, are chosen to route the control message. As shown in Figure 4 (a), a hyperbola constraint contains three anchor nodes ID and the two hop-counts represent the a value and the circle radius, respectively. Here the first anchor nodes ID represents the foci of the hyperbola and the third one represents the center of the circle. a is a constant in the following equation $h_1 - h_2 = a$ where h_1, as shown in Figure 4 (a), is the length of a point in the hyperbola to the left foci A_1 and h_2 is the length of that point to the right foci A_2. Figure 4 (b) shows the case when a sensor decides whether it should rebroadcast the control message by comparing all the entries in its DV-Hop table with all the constraints in the routing message. If the DV-Hop information of a sensor satisfy any of the routing constraint ($h_1 - h_2 == a$ and $h_3 <= r$), the sensor will rebroadcast. Otherwise, it will drop the packet.



(a) A segment of hyperbola trajectory

Anchor ID	Hop distance
...	...
A_1	$h_1 = a + h_2$
...	...
A_2	h_2
...	...
A_3	h_3
...	...

(b) The routing constraint of given hyperbola

Figure 4. Example of a hyperbola trajectory and its routing constraints

Calculating the routing constraints requires a lot of computational power and consumes a long time to get the result. For a working area with 80 anchor nodes, about three minutes are required to get the approximate routing constraints, which is unacceptable for the real-time tracking application. However, the greedy algorithm to calculate the DV-Hop based routing constraints contains redundant calculations. For example, for every new routing trajectory the area of all the possible hyperbola shapes will be calculated once. In CUDA Kernel, all the shape area will be ranked based on the new trajectory. However, because the DV-Hop for all the sensors is decided by the topology of the anchor nodes of the working area at the initialization stage, the area of all the possible shapes won't change no matter what the input trajectory looks like. The computational time should able to be reduced by reuse the shapes' area information calculated before. In this paper, the proposed framework uses a Seq2Seq model to remember the relationship between the trajectory area set and the best hyperbola constraint that covers the TAS and the future predicted trajectory. The Seq2Seq learning model will be elaborated in Section 4.

4. SEQ2SEQ MODEL FOR CONSTRAINT PREDICTION

To predict the future trajectory of the moving object as well as to generate the next hyperbola constraint, a sequence to sequence model [29] containing two LSTM layers, as shown in Figure 5, has been proposed. One of the LSTM layer act as "encoder" that processes the targets' previous trajectory, and returns its internal state. However, the outputs of the encoder LSTM are discarded. This internal state will serve as the initial state of the decoder in the next step. Note that we assumed in Section 3.1 that sensors don't know their locations and the target detection area is much smaller than the sensor's radio range. Thus, we can use the DV-Hop of the sensor that detects the target as the target's location. A list of the DV-Hop information sorted in timestamp order can be seen as the previous trajectory of the target. The encoder LSTM in the proposed Seq2Seq model takes these sequences of DV-Hops as the input.

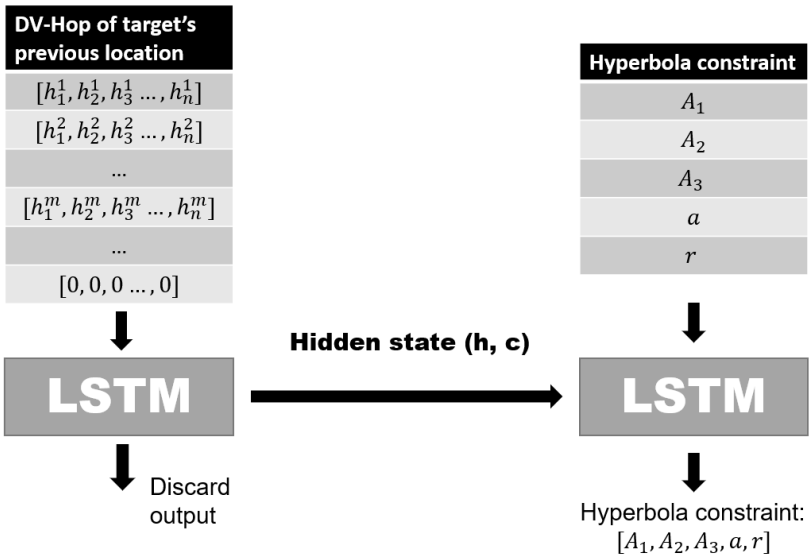


Figure 5. Trajectory prediction Seq2Seq model

Another LSTM layer acts as a "decoder" which is trained to predict the hyperbola constraint element by element. The decoder LSTM uses the encoder's hidden parameters' value (h, c) as its initial value. Specifically, it is trained to predict the best hyperbola foci (A_1 and A_2), the circle center (A_3), and the hops count of a and r of hyperbola and circle. So the decoder LSTM iterates five times to fin the following attributes of the constraint:(A_1, A_2, A_3, a, r). In each iteration, the output is dense and softmax to a list with a size of $max(Num_{anchors}, Num_{hopLimits})$ where $Num_{anchors}$ is the number of anchor nodes in the current working area and $Num_{hopLimits}$ is the predefined maximum number of hops a broadcast allows. Then the $argmax$ is used to find the anchor nodes IDs or the hop counts that provide the minimal loss in the current iteration.

Once the local edge server predicts the hyperbola constraint of the target's future trajectory, it will send an activation message to the sensors located within the trajectory through the shortest path from itself to the location where the target has been detected most recently. The nodes in the shortest path, which are also encoded to a set of hyperbola

constraints, will only wake up temporarily to forward the activate message and the reset message (to put the sensor back in LPL mode). The path encoding procedure is discussed in Section 5.

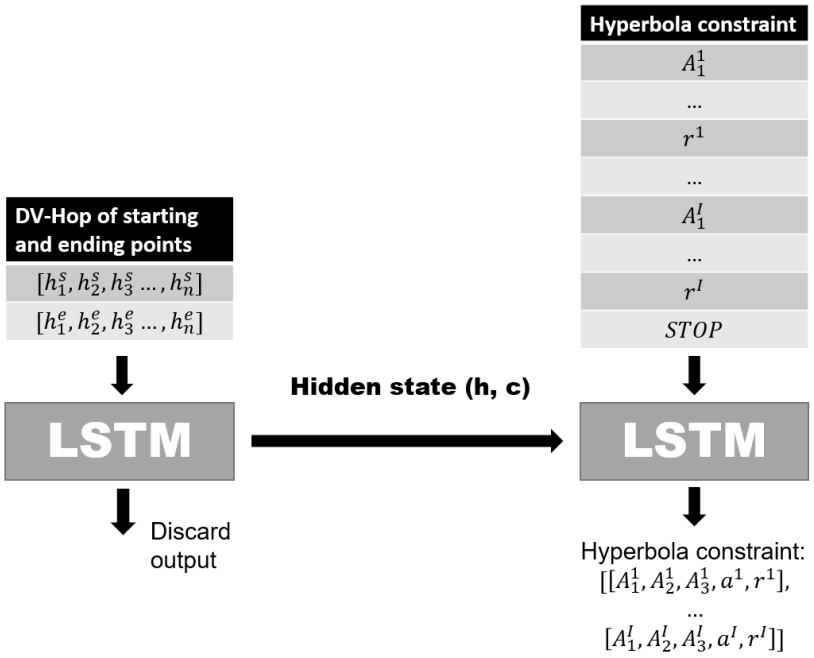


Figure 6. Path encoding Seq2Seq model

5. SEQ2SEQ MODEL FOR PATH ENCODING

The shortest path encoding problem can be formulated as follows. There is a start point's (P_s) DV-Hop of all the anchors DH_s and an end point's (P_e) DV-Hop of all the anchors DH_e. It is to find the best hyperbola constraints set that covers the line from P_s to P_e. The proposed path encoding Seq2Seq model is designed as in Figure 6 where A_k^I means the anchor node k in I'th constraint, a^I and r^I are the hyperbola and circle parameters which stand for the hop differences between hyperbola foci and the radius of the circle. The encoder of the proposed model takes two DV-Hop entry as the input and discards the

output. The decoder uses the encoder's hidden parameter value as its hidden parameter's initial value. It predicts a parameter of a hyperbola constraint in each iteration until the stop symbol.

To route the activate message to the predicted trajectory, the framework defines an activate packet in Table 1. The variable-length activation packet contains a message ID, a local edge server ID, the length of the payload, a control bit, 5 bits activation area's hyperbola constraint, and a set of hyperbola constraints of the temporary routing path.

Table 1. Activate/Reset packet's payload data structure

Descriptions	Starting Bytes	Length in Bytes
Message ID	0	4
edge server ID	4	2
payload length	6	1
control bit	7	1
active/reset area	8	5
routing constraints	13	100

6. TRAJECTORY PREDICTION USING SEQ2SEQ MODEL

To train the trajectory prediction Seq2Seq model, a real-world taxi trajectory dataset [30] is used. For the path encoding model, we use a path generator to enumerate all the possible paths in the working area. These two datasets and the training details are elaborated below.

6.1. TAXI DATASET AND TRAINING

The real-world taxi trajectory data [30] contains around two million taxi trips in the city of Porto in Portugal. Each row of the trajectory data contains the trajectory of a taxi trip in the city of Porto in Portugal. The trajectory is represented as a list of 8 bytes of GPS data (latitude and longitude) sampled every 15 seconds. The 2 GB data-set contains trajectories of different shapes, lengths, starting, and ending locations. As the road map in a city is fixed, the taxi trajectory is a perfect dataset that contains the information of the road map of an area, the hotspot in a city, popular commute routes, and daily traffic conditions. Following pre-processing steps are taken to transfer the raw data to be the training ready dataset.

First is to sanitize the data by discarding, from the taxi dataset, data containing wrong GPS values. The second is to define the working area and transfer the raw data to the coordinates in the working area. In the proposed framework, each working area has a fixed length, width, number of anchors and the topology of the anchor nodes. As shown in Figure 7, all the trajectories within the latitude of 41.188 to 41.138 and the longitude of -8.653 to -8.578 are selected. We define the working area as of length 1800, width 1200, and 60 anchor nodes. Then, we translate the GPS values of the taxi dataset to the working area pixels. As the anchors' location are transparent to the remote cloud, the DV-Hop of all the pixels to the anchors can be calculated. Thus, the taxi trajectory can be translated into a list of DV-Hop values. The first half of the trajectory's DV-Hops are stored in a 2D array. As our model requires 20 previous timestamps' locations, for a short trajectory, zero DV-Hops are appended at the end of the previous trajectory's DV-Hop list. At last, the training input data are prepared by concatenating all the trajectory's DV-Hop sequences to a 3D array.

Third, rebuild the trajectory from discrete GPS points and calculate the hyperbola constraints. As the proposed trajectory encoding the Seq2Seq model directly predicts the hyperbola constraints, the training output data need to be calculated mathematically. In this work, we first need to rebuild the trajectory area set, which is a list of pixels covered by

the trajectory, using the last half of the GPS points. Then, use a greedy algorithm to find the best hyperbola that covers most of the area of the trajectory while also overlap with the first GPS points. Last, use one-hot encoding to encode the hyperbola constraint as shown in Figure 8. Then, the one-hot encoded hyperbola constraint is used as the training output.

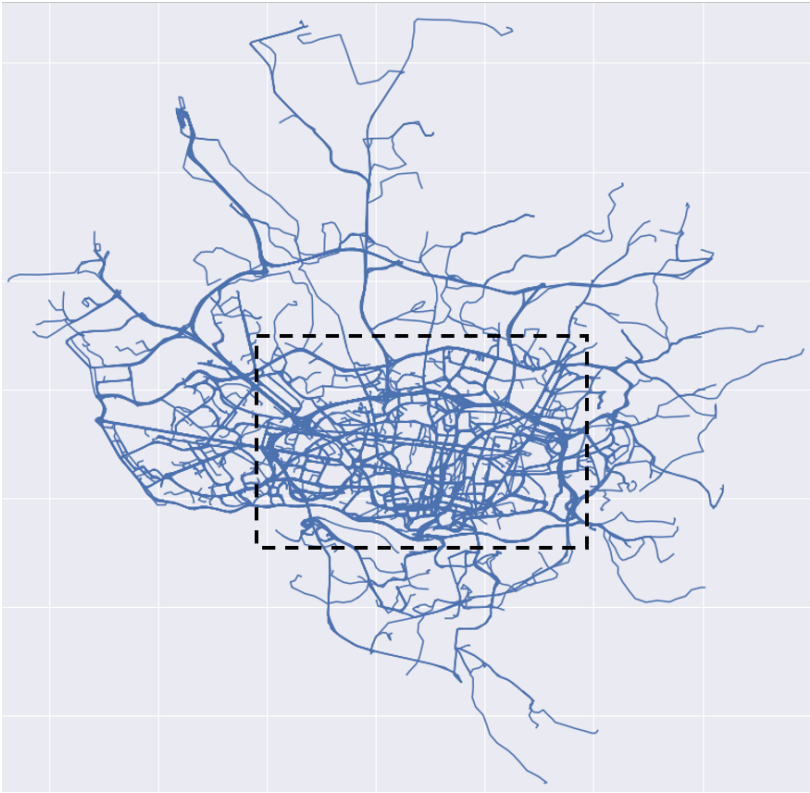


Figure 7. Translate road map to trajectories in a working area

The encoder and decoder of the proposed model are both a LSTM with 256 hidden layers. Total trainable parameters are 952,360. Total training data number is 1.4 million. Batch size is 128. We used the Adam optimizer [31] with $learning_{rate} = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-7}$. The loss function is chosen to be the cross-entropy.

6.2. DATA GENERATING AND TRAINING FOR THE PATH ENCODING MODEL

After getting the constraint that covers the target's prediction trajectory (TPT), we need to route the control message from the current local edge server to the TPT. Although the local edge server may have the location information of itself, it won't have the knowledge of the TPT as the anchors' locations are not disclosed to the local edge server. In the proposed framework, the shortest path from the current local edge server and the most recent sensor that detects the target is chosen as the route path. The beginning and ending points of the path, which are in DV-Hop format, are known to the local edge server. The proposed framework uses a Seq2Seq model to predict the routing constraints from the start point to the endpoint as shown in Figure 6.

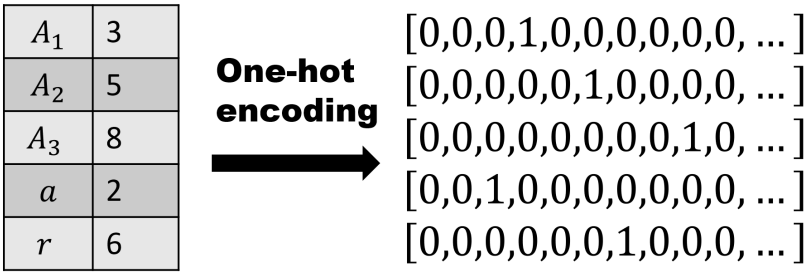


Figure 8. One-hot encoding for the hyperbola constraint

To train the model, the input data are generated from a path generator. The path generator transforms the working area into a grid with 45 horizontal cells and 30 vertical cells. For each grid cell, the DV-Hop to all the anchors is calculated. Then the path generator enumerates all the possible grid cell pairs using a two-layer nested loop. For each pair of the start and end points, a straight line is drawn as the path. Then a set of hyperbola constraints are calculated as shown in paper [24]. The encoder input of the model will be the DV-Hop of the grid cell pair. The decoder output is a set parameter of hyperbola constraints. The total training data has 0.91 million paths and constraints sequence. The encoder LSTM has

128 hidden layers and the decoder LSTM has 256 hidden layers. The batch size is 128. We also choose Adam's optimizer with the same parameters as in the training of the trajectory prediction model.

7. RESULTS

7.1. HARDWARE, FRAMEWORK, AND TRAINING TIME

In the experiment, the Seq2Seq model is trained for one working area with 60 randomly deployed anchor nodes. The anchor nodes' IDs increment from zero in the direction of left to right and top to bottom. The working area has 1800 unit length and 1200 unit width. It is transferred into a grid with a 40×40 grid cell size for training the path encoding model. The one-hop distance is 60 units for all the sensors.

Based on cross-validation, the proposed trajectory prediction model is trained on 131 epochs. The proposed path encoding model is trained on 162 epochs. The hardware for training the models is a desktop with two Xeon E5-2680 V4 and one RTX 2070 GPU. The framework is Keras of PyTorch. For trajectory prediction model training, each epoch takes 155 seconds. For path encoding model training, each step takes 237 seconds.

$$ACS = \frac{Area_{covered\_by\_constraints} \cap Area_{Trajectory}}{Area_{Trajectory}} \tag{1}$$

$$FAR = \frac{Area_{covered\_by\_constraints} - Area_{Trajectory}}{Area_{covered\_by\_constraints}} \tag{2}$$

7.2. PREDICTION ACCURACY

This experiment uses the real-world taxi trajectory dataset [30]. As the dataset only contains the GPS points, we reconstruct the trajectory by connecting adjacent GPS points with straight lines as we have discussed in Section 6.1. The trajectory's width is the same

as the radio range (60 units in this experiment). The area of the trajectory can be calculated as the summation of all the line segments of the trajectory. While each line segment can be seen as a rectangle with length equal to the distance between two adjacent GPS points and width equal to the radio range. Also, in this experiment, two types of accuracy metrics are defined. One is called the valid coverage score (ACS) which is defined in Equation 1. The other is called the false activation rate (FAR) which is defined in Equation 2. As shown in Figure 9, $Area_{Trajectory}$ is the area of the desired target trajectory. $Area_{covered\_by\_constraints}$ is the area calculated by testing every pixel of the working area with the DV-Hop constraints. All the pixels valid for the hyperbola constraints are counted. Therefore, the high ACS score means more areas of the target trajectory are predicted and covered. The higher FAR rate means more false areas are predicted which will incur redundant rebroadcast and waste sensors' energy.

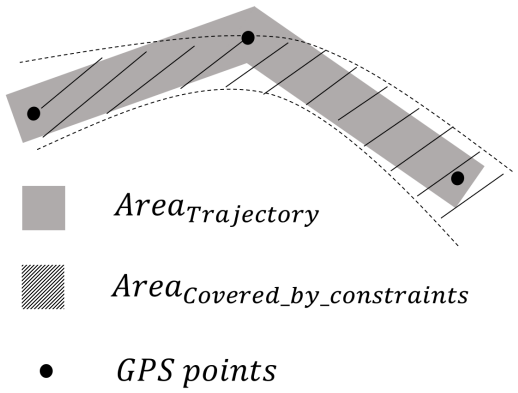


Figure 9. An example of the definition of $Area_{covered\_by\_constraints}$ and $Area_{Trajectory}$

As no previous works predict DV-Hop for target tracking, in this experiment, we designed and tested several machine learning models including LSTM, bi-direction LSTM, and Seq2Seq models. The results are shown in Table 2. The LSTM model has both the worst ACS and FAR. The bidirectional LSTM learn trajectories in both directions which has slightly better performance regarding the ACS and FAR metrics. The proposed trajectory

prediction Seq2Seq model achieves the best prediction of ACS and FAR scores. We found when the Seq2Seq model predicts different from the expected constraint, it will predict anchor nodes' IDs and hop count values close to the expected constraint's anchor nodes' IDs and hop counts. As discussed in Section 7.1, anchor nodes with similar values usually close to each other. Different anchor nodes with similar locations and similar hop constraints will usually provide a similar coverage area. Thus, the ACS of the proposed Seq2Seq could achieve around 0.8 coverage which is sufficient for the target tracking application as multiple independent predictions for different working areas could activate sensors in different trajectories.

Table 2. Accuracy comparison for four different learning models

Models	ACS	FAR
Expected constraints from greedy algorithm	0.92	0.36
Our LSTM	0.43	0.75
Our bidirectional LSTM	0.51	0.70
Trajectory prediction Seq2Seq model	0.76	0.54
Path encoding Seq2Seq model	0.85	0.42

Different from the trajectory prediction Seq2Seq model, the proposed path encoding Seq2Seq model has higher ACS and FAR scores because the training dataset and prediction dataset have the same synthetic dataset. As discussed in Section 6.2, the path encoding model enumerates all possible routing paths of the grid cells. A Large epoch number is chosen. The overfitting strategy works well in this problem.

7.3. PREDICTION SPEED

For the target tracking application, the fast prediction time is essential. The prediction must be real-time, otherwise the target can move away from the predicted location. The DV-Hop constraints based trajectory encoding algorithm [24] consumes more time in calculating the geometric constraints if the anchor nodes number is larger or the trajectory area is bigger. Although it has better encoding accuracy, it can't be used on target tracking due to the high delay. However, the model-based constraints prediction approach has constant time consumption based on the model size. In this experiment, we compare the time consumption of geometric encoding and the proposed Seq2Seq model-based approaches for encoding the same trajectories in the taxi trajectory dataset [30]. The result is shown in Figure 10.

The result shows when the number of anchor nodes increases, the time of calculating the hyperbola constraint has polynomial growth. When the number of anchor nodes is 20, the computational time is 210 milliseconds which is good for target tracking applications. However, when the number of anchor nodes is 60, the computational time becomes 4440 milliseconds which is not suitable for the real-time target tracking applications.The proposed trajectory prediction model with 952,360 parameters only takes an average of 67 milliseconds for all the prediction tasks which are only 1/70 of the time consumed by calculating the same hyperbola constraint with 60 anchor nodes and 20 GPS points. Instead of repeating the shape coverage area calculations for different trajectories, the Seq2Seq model-based approach remembers the trajectory and hop constraint patterns and store all the information in its hidden parameters. When predicting the hyperbola constraints, the calculation is like searching a state dictionary which is faster than start over all the calculations from scratch. The prediction speed of the proposed model is good enough for any real-time applications including the target tracking application.

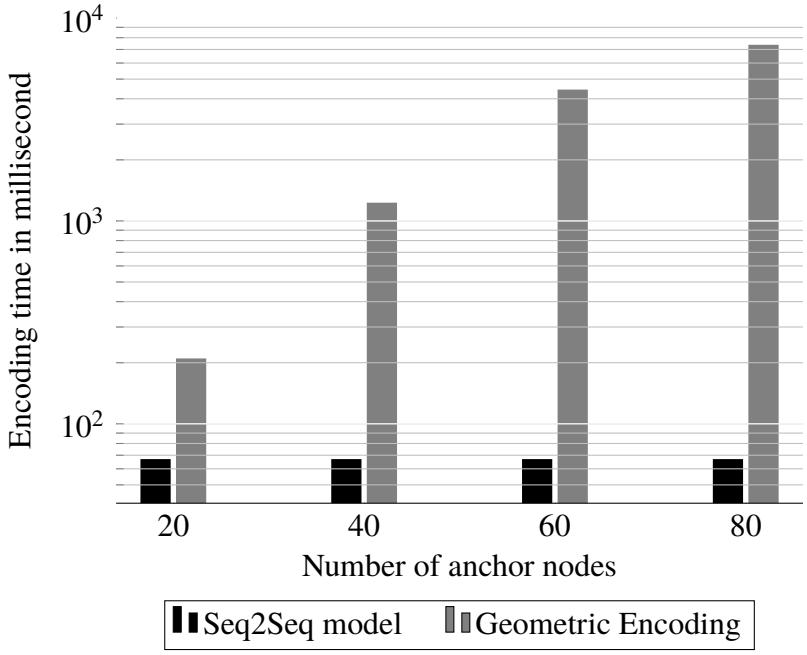


Figure 10. Time consumption of encoding trajectories with 20 GPS points to hyperbola constraints

Table 3. Parameters for the experiments

Area of deployment	1800×1200 m
Number of sensor nodes	3000
Communication range	60 m
Number of edge devices	3
Broadcasting hop limitation	30 hops
LPL sleeping time	600ms
LPL wake time	10ms
Energy model	MicaZ
Number of clusters	20
Coverage threshold	90%
Number of anchor nodes	60
Anchor cover range	20 hops

7.4. COMPARISON WITH PREVIOUS WORKS

Here, we compare the proposed trajectory prediction Seq2Seq model and target tracking framework with the state-of-the-art cluster-based target prediction and tracking framework in [9]. The authors in [9] use the target's current speed and direction to predict its future location. To save energy, they proposed an adaptive cluster head election algorithm that reduces the setup overhead by increasing the steady-state phase. We simplified their implementation by adopting a fixed cluster topology for a working area. Thus, their setup overhead is minimized. So, our experiments only compare the delay of delivering a control message and sensors reporting the target's location information and the energy consumption caused by redundant rebroadcasting. The network setup is in Table 3.

In each step of the experiment, a random sensor is chosen to report the target's location to a random local edge server. The cluster-based approach allows the sensor to route the report packet to the cluster-head first. Then the cluster-head will route the packet to the local edge server. While the sensor in the proposed framework broadcasts the report packets toward the hop-gradient decrease direction. The experiment is simulated in TOSSIM [32] which is an event-driven sensor simulator. The random topology of the WSN is generated and hard-coded for each simulated sensor. The cluster mechanism is simulated as follows. First, we simulate the distributed cluster procedure and the reporting path is calculated with the cluster information. Second, based on the randomly chosen sensor and the local edge server, we embed the routing path in the reporting packet. Then we inject the reporting packet to the node selected to report. Third, we simulate the process in TOSSIM where all radio communication information including source, receiver, and timestamp are recorded. Next, we calculate the average delay for different hop distance between the sensor and local edge server. For our proposed approach, same procedures are considered except that the injected packets contain hop constraints instead of cluster-based routing information.

Figure 11 shows the delay of a sensor in delivering the target detecting signal to the local edge server. As discussed in Section 3.3, the sensors in the proposed approach rebroadcast the target detection message only if their hop count to the local edge server is lower than the previous sensors that rebroadcast the message. The X-axis in Figure 11 is the number of hops from the sensor (that detects the target) to the local edge server. The Y-axis is the reporting message delivery delay.

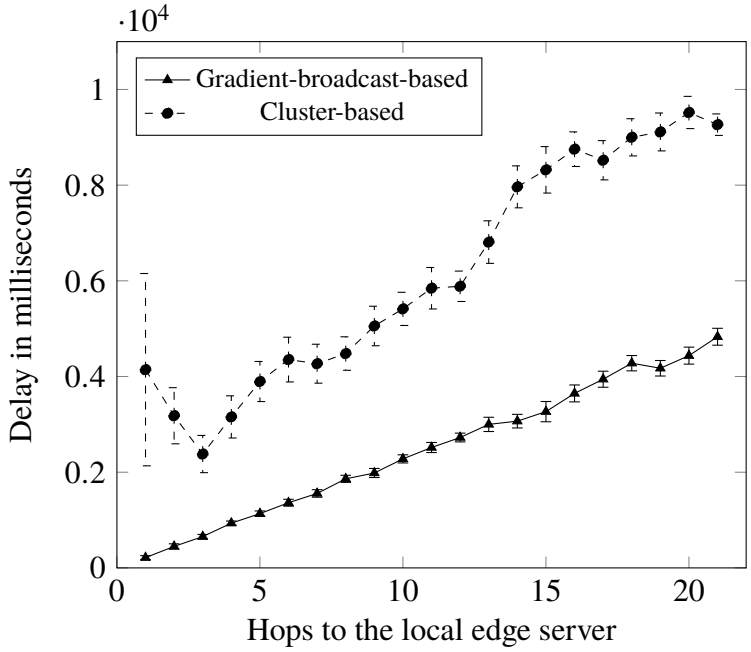


Figure 11. Delay in a sensor reporting detected target's location to the local edge server (in milliseconds)

The result shows that the gradient-based broadcasting of the proposed target tracking framework has reduced reporting delays. Also, when the number of hops from the reporting sensor to the local edge server increases, the average delay in both the approaches increase. The reason is as follows. The cluster-based approach in [9] requires all the slave nodes to report the target's information to the cluster head through a pre-defined route. The cluster head then will deliver the target's location to the local edge server for processing. However,

the gradient-based reporting protocol always reports through the shortest path with minimal hops. Thus, the reporting delay of the proposed protocol is lower than the cluster-based target detecting protocol. Also, in the experiment, the cluster size is about 6 to 10 hops in width. For the cluster-based routing approach, when the sensor that detect the target is close to the local edge server, it can't send or receive control messages to/from the local edge server directly. Thus, in the experimental results in Figures 11 and 12, the delay increases when the sensor is one or two hops away from the local edge server.

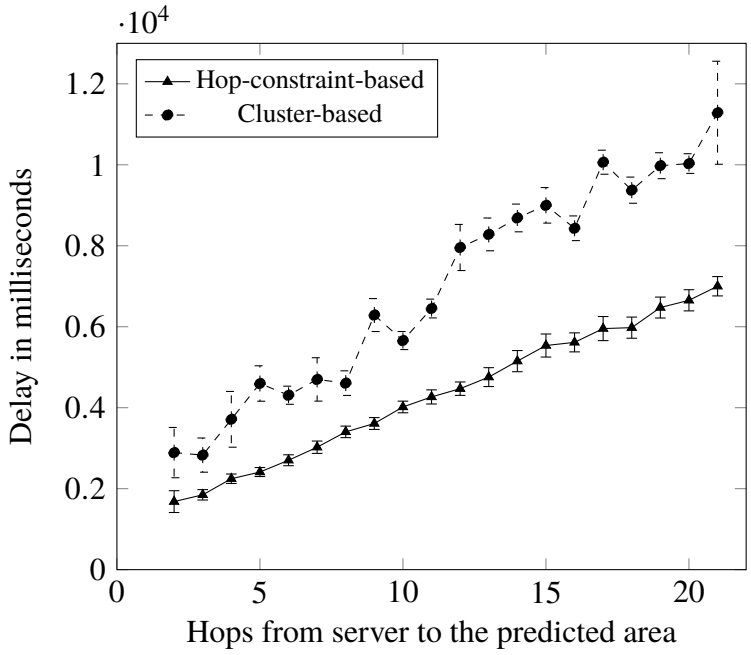


Figure 12. Delay in local edge server sending control messages to the target's predicted trajectory (in milliseconds)

Figure 12 shows the delay at the local edge server in predicting the target's future location and delivering the control message to the predicted area. The proposed approach uses the hop-constraint-based message dissemination protocol [24]. It uses hyperbola constraints and only the sensors which meet the constraints could rebroadcast. The X-axis of Figure 12 is the number of hops from the local edge server to the predicted area which

is the center sensor within all the sensors in the predicted area. The Y-axis is the message delivery delay from the local edge server to all the sensors within the predicted area. The result shows that the proposed approach has a lower delay in delivering the control message than the cluster-based approaches. The reason is similar to the previous delay reporting experiment. The control message of the cluster-based approach can't route to the location of interest directly. It needs to route through the cluster-head which requires extra hops.

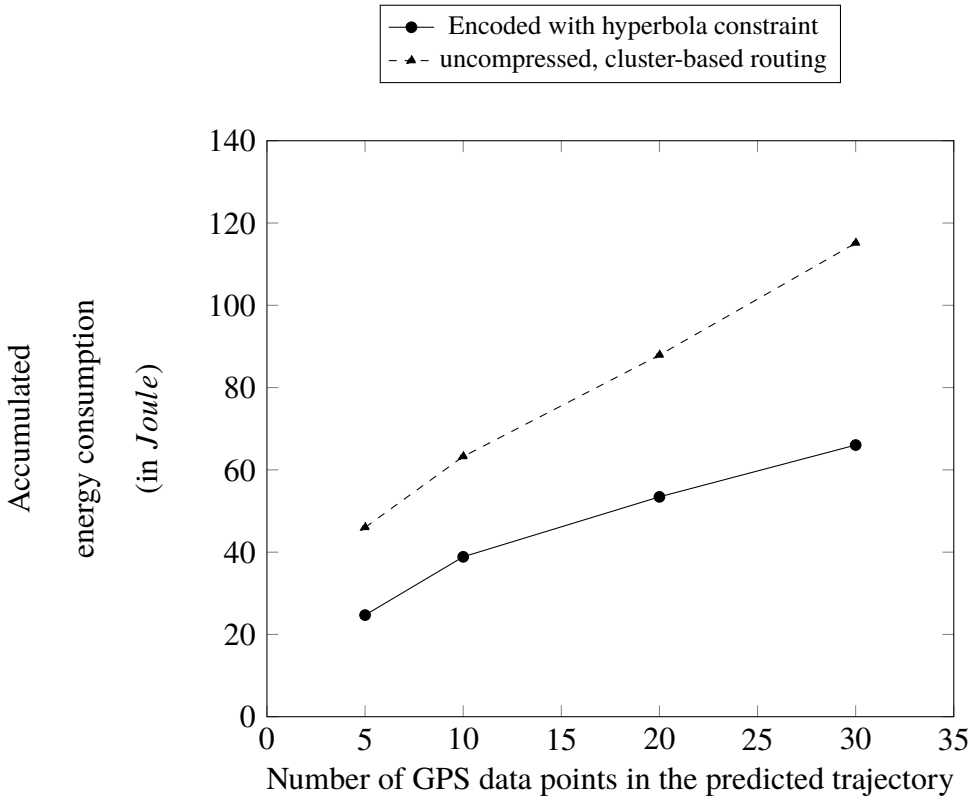


Figure 13. Accumulated energy consumption in tracking and reporting target's location

Figure 13 shows the energy consumption of the cluster-based and the proposed Seq2Seq prediction based target tracking frameworks. The experiment uses the powerTOSSIM-Z simulator [33] to estimate the energy consumption. In this experiment, we assume that the target moving speed is much smaller than the radio transmitting speed. Thus, the energy

consumption model for each framework is the energy consumed by the activated sensor nodes. When more sensors are activated, more is the energy consumed. The result of the experiment shows that the proposed prediction based target tracking approach consumes much less energy because fewer sensors are activated for tracking the moving target. The cluster-based approach will activate all the nodes in the predicted clusters which are not energy efficient.

8. CONCLUSION AND FUTURE WORK

In this paper, we proposed an efficient Seq2Seq learning model to predict the future trajectory of a moving object using WSNs. The proposed Seq2Seq model predicts and generates the DV-Hop based routing constraints without knowing the location of the anchor nodes in the object tracking area. Therefore, it enables sensors to wake up before the target passes by a given area and sleep after. The proposed framework decouples the data plane and the control plane by using the constraint-based routing protocol that enables different applications to share the same WSN. The Seq2Seq based hyperbola constraint generation model speeds up the computation significantly and thus it enables a real-time control-message generation for object tracking. It also allows any edge device to participate in the target tracking applications as location information is hidden. In the future, we plan to create an online training transformer network model that could achieve better prediction accuracy.

REFERENCES

[1] Mohamed Hamdi, Noureddine Boudriga, and Mohammad S Obaidat. Whomoves: An optimized broadband sensor network for military vehicle tracking. *International Journal of Communication Systems*, 21(3):277–300, 2008.

[2] Hong Zhang, Zeyu Zhang, Lei Zhang, Yifan Yang, Qiaochu Kang, and Daniel Sun. Object tracking for a smart city using iot and edge computing. *Sensors*, 19(9):1987, 2019.

[3] Injong Rhee, Ajit Warrier, Mahesh Aia, Jeongki Min, and Mihail L Sichitiu. Z-mac: a hybrid mac for wireless sensor networks. *IEEE/ACM Transactions On Networking*, 16(3):511–524, 2008.

[4] Taneli Riihonen, Dani Korpi, Matias Turunen, and Mikko Valkama. Full-duplex radio technology for simultaneously detecting and preventing improvised explosive device activation. In *2018 International Conference on Military Communications and Information Systems (ICMCIS)*, pages 1–4. IEEE, 2018.

[5] Shrawan Kumar and DK Lobiyal. Novel dv-hop localization algorithm for wireless sensor networks. *Telecommunication Systems*, 2017.

[6] Xingjuan Cai, Penghong Wang, Lei Du, Zhihua Cui, Wensheng Zhang, and Jinjun Chen. Multi-objective three-dimensional dv-hop localization algorithm with nsga-ii. *IEEE Sensors Journal*, 19(21):10003–10015, 2019.

[7] Penghong Wang, Jianrou Huang, Zhihua Cui, Liping Xie, and Jinjun Chen. A gaussian error correction multi-objective positioning model with nsga-ii. *Concurrency and Computation: Practice and Experience*, 32(5):e5464, 2020.

[8] Samer Samarah, Muhannad Al-Hajri, and Azzedine Boukerche. A predictive energy-efficient technique to support object-tracking sensor networks. *IEEE Transactions on Vehicular Technology*, 60(2):656–663, 2010.

[9] Khalid A Darabkh, Wijdan Y Albtoush, and Iyad F Jafar. Improved clustering algorithms for target tracking in wireless sensor networks. *The Journal of Supercomputing*, 73(5):1952–1977, 2017.

[10] Zhibo Wang, Wei Lou, Zhi Wang, Junchao Ma, and Honglong Chen. A hybrid cluster-based target tracking protocol for wireless sensor networks. *International Journal of Distributed Sensor Networks*, 9(3):494863, 2013.

[11] Chunming Wu, Chen Zhao, and Haoquan Gong. Energy-efficient target tracking algorithm for wsns. *3D Research*, 10(1):1, 2019.

[12] Chao Sha, Lian-hua Zhong, Yao Bian, Dan-dan Song, and Chun-hui Ren. A type of energy-efficient target tracking approach based on grids in sensor networks. *Peer-to-Peer Networking and Applications*, 12(5):1041–1060, 2019.

[13] Xiao-ping ZHANG and Gui-xiong LIU. Target tracking prediction in wsn based on quadratic polynomial motion modeling [j]. *Journal of Jinan University (Natural Science & Medicine Edition)*, 5, 2009.

[14] Kenji Okuma, Ali Taleghani, Nando De Freitas, James J Little, and David G Lowe. A boosted particle filter: Multitarget detection and tracking. In *European conference on computer vision*, pages 28–39. Springer, 2004.

[15] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter, 1995.

[16] H. Zhang, X. Zhou, Z. Wang, H. Yan, and J. Sun. Adaptive consensus-based distributed target tracking with dynamic cluster in sensor networks. *IEEE Transactions on Cybernetics*, 49(5):1580–1591, 2019.

[17] Gharavian FayaziBarjini. Target tracking in wireless sensor networks using ngekf algorithm. *J Ambient Intell Human Comput*, pages 3417–3429, 2019.

[18] Satish R Jondhale and Rajkumar S Deshpande. Kalman filtering framework-based real time target tracking in wireless sensor networks using generalized regression neural networks. *IEEE Sensors Journal*, 19(1):224–233, 2018.

[19] Tianzhu Zhang, Changsheng Xu, and Ming-Hsuan Yang. Multi-task correlation particle filter for robust object tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4335–4343, 2017.

[20] Tianzhu Zhang, Si Liu, Changsheng Xu, Bin Liu, and Ming-Hsuan Yang. Correlation particle filter for visual tracking. *IEEE Transactions on Image Processing*, 27(6):2676–2687, 2017.

[21] Gurjit Singh Walia, Ashish Kumar, Astitwa Saxena, Kapil Sharma, and Kuldeep Singh. Robust object tracking with crow search optimized multi-cue particle filter. *Pattern Analysis and Applications*, 23(3):1439–1455, 2020.

[22] Dhiren P Bhagat and Himanshukumar Soni. Target tracking using a hybrid kf-pso tracking model in wsn. In *International Conference on Emerging Technology Trends in Electronics Communication and Networking*, pages 83–98. Springer, 2020.

[23] Shalli Rani, Syed Hassan Ahmed, and Ravi Rastogi. Dynamic clustering approach based on wireless sensor networks genetic algorithm for iot applications. *Wireless Networks*, pages 1–10, 2019.

[24] Xiaofei Cao and Sanjay Madria. Efficient geospatial data collection in iot networks for mobile edge computing. In *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, pages 1–10. IEEE, 2019.

[25] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless networks*, 8(2-3):153–167, 2002.

[26] Ji-Young Jung and Dong-Yoon Seo. Counter-based broadcast scheme considering reachability, network density, and energy efficiency for wireless sensor networks. *Sensors*, 18(1):120, 2018.

[27] Kok-Poh Ng, Charalampos Tsimenidis, and Wai Lok Woo. C-sync: Counter-based synchronization for duty-cycled wireless sensor networks. *Ad Hoc Networks*, 61:51–64, 2017.

[28] Tao Liu, Qingrui Li, and Ping Liang. An energy-balancing clustering approach for gradient-based routing in wireless sensor networks. *Computer Communications*, 35(17):2150–2161, 2012.

[29] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[30] Taxi trajectory data. www.kaggle.com/crailtap/taxi-trajectory.

[31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[32] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137. ACM, 2003.

[33] Enrico Perla, Art Ó Catháin, Ricardo Simon Carbajo, Meriel Huggard, and Ciarán Mc Goldrick. Powertossim z: realistic energy modelling for wireless sensor network environments. In *Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pages 35–42. ACM, 2008.

VI. A WSN TESTBED FOR Z-ORDER ENCODING BASED MULTI-MODAL SENSOR DATA COMPRESSION

Xiaofei Cao, Sanjay Madria, Takahiro Hara

ABSTRACT

Wireless sensor networks (WSNs) have significant limitations in available bandwidth and energy. The limited bandwidth in sensor networks can cause higher message delivery latency. In this demo, we demonstrate working of novel Z-order based data compression schemes (Z-compression) to reduce energy and save bandwidth without increasing the message delivery latency using a TelosB motes based sensor network testbed. Instead of using the popular Huffman tree style based encoding, Z-compression uses Z-order encoding to map the multidimensional sensing data into one-dimensional binary stream transmitted using a single packet. We have also designed and developed a data concatenating algorithm which can concatenate small packets into large packets, thus increases the throughput of the WSNs.

Keywords: Sensor network, Data compression, Z-order encode

1. INTRODUCTION AND PROBLEM STATEMENT

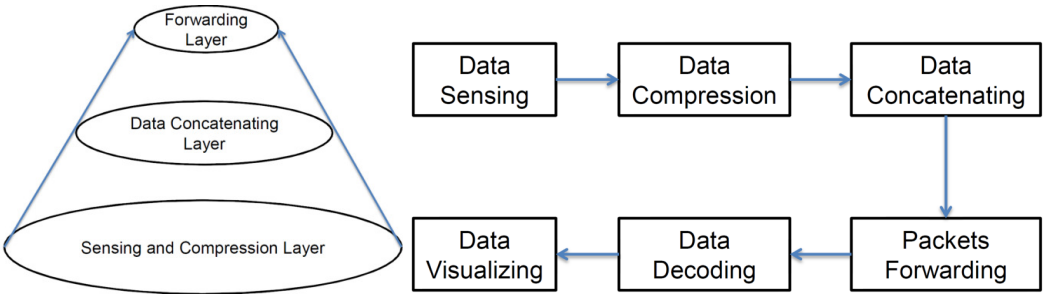
Wireless sensor networks (WSNs) are being developed for a plethora of emerging applications in wide range of disciplines. Many of these multi-modal sensor applications asking for a higher data stream rate with lossless property, and therefore, cannot tolerate high latency (or drop data) due to limited link bandwidth in WSNs. Since batteries are the typical power source for wireless sensors and cannot easily be replaced, the energy

consumption is another primary constraint in the design of multi-modal WSNs. Many research efforts have shown that the radio communication is the predominant factor in all energy consumption metrics of the WSNs.

In a multi-hop wireless sensor network with multiple different sensor types, in order to save energy and bandwidth, the multi-modal sensor data need to be compressed without affecting the data integrity. The compression procedure should not introduce delays, thus it should be spatial independent and doesn't need to wait for more data to compress. The solution must also be application independent which can compress any type and any number of data attributes without any modification. Due to the limitations of computing resource such as RAM, the algorithm also need to be light weight and computationally efficient.

2. SYSTEM ARCHITECTURE

In this demo, we classified the sensor nodes in WSNs into three layers based on their utility. When applying Z-compression [1], the sensing and compression layer takes the responsibility of sensing and compression. The data concatenating layer will concatenate compressed data thus reduces the number of packets need to be transmitted. The forwarding layer will forward the data to the sink. Figure 1(b) shows six different stage the data flow in the WSNs. The data is generated and flow through the lower layer to the upper layer.



(a) Layers of WSNs (b) Data flow

Figure 1. Layers of WSNs and the data flow

2.1. DATA COMPRESSION AND DECOMPRESSION

The local compression uses optimized Z-compression which is the optimized version of the naive z-compression. The Naive Z-compression uses Z-order encode and all-is-well scheme [2]. It fuses multi-dimensional data into a binary data stream by interleaving the binary representations of the input data.

To optimize the naive Z-compression, We can use the z-value with odd length to represent the 'skewed' data. Here, we define a two-dimensional dataset as skewed when the number of bits of the larger delta value is more than two times of the number of bits of the smaller.

2.2. CONCATENATING COMPRESSED DATA

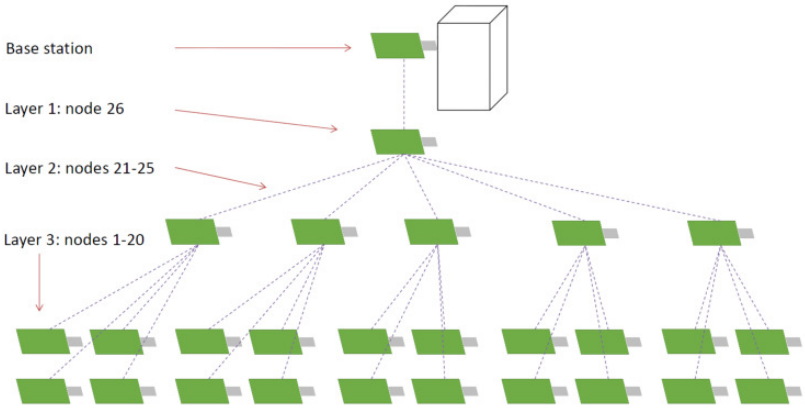
Data concatenating happens in the intermediate node shown in Figure 2(a). According to the previous experimental analysis of radio performance[3] [4], we found that reducing the payload size of leaf nodes' packets will not give much energy saving. It is the intermediate nodes rather than the leaf nodes which is the bottleneck in the network. Without reducing the number of packets the intermediate node transmits, we can't save the energy and prolong the lifetime of the whole network. We therefroe concatenate the upstream data to increase the payloads of each outgoing packets of the intermediate node to reduce the number of outgoing packets.

When concatenating, the intermediate nodes need to collect enough data from its leaf nodes. Once the sum of the size of the collected data exceeds the threshold, we sort the data based on the size in descending order. Finally, we create a byte array as the payload and insert the sorted data if its size is equal to the previous and insert zero bytes if it's size is smaller than the previous data's size.

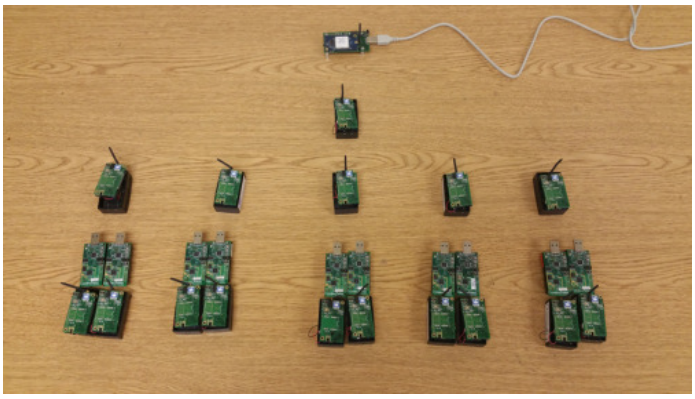
3. SENSOR TESTBED IMPLEMENTATION

3.1. SYSTEM SETUP

In this demo, we use a three-hop wireless sensor network with 27 nodes to show the effectiveness of the compression algorithm. Fig 2(a) shows the topology of the system. The sensors labeled 1-20 in the system are working under three different sensing modes fetching multi-modal sensor data. The sensors labeled 21-25 are performing as the intermediate node and will concatenating the compressed data from its children. The node 26 is connected to the base station directly and will only forward the incoming packets from node 21-25.



(a) System topology for demo



(b) real world layout

Figure 2. System topology and real-world layout

In real-world, the WSNs may have thousands of leaf nodes and hundreds of concatenating nodes and many forwarding nodes. So, even the sensing period of the sensor nodes may be long, the stream rate in the upper layer can be very high. To simulate the WSNs data flow and to figure out how much our Z-compression algorithm could mitigate the bottleneck of the WSNs in terms of bandwidth and energy, we decrease the sensing and transmitting interval of the leaf nodes. Although the minimum sensing interval is much longer than the minimum radio transmitting interval, we let the radio be able to transmit the previous sensing samples if the new samples are not updated.

3.2. DATA COLLECTION AND VISUALIZATION

We use a PC as the base station and a TelosB mote as the interface to collect the sensor data. TelosB motes communicate with other sensor nodes with IEEE 802.15.4 radio. It is connected to the PC using the serial port. The sensor nodes will collect the temperature, humidity, Visible light, invisible light, and voltage data. We visualize the sensor nodes and the sensing data with a JavaFX UI which will display the current sensing data, performance summary, control button, etc.

3.3. PERFORMANCE EVALUATION

We also provide a performance evaluation function that will visualize the performance metrics, including the compression ratio, the delivery rate, the energy consumption, the packets delivered, the average latency and the log size in the performance summary page. We compare four different compression approaches such as LEC [5], TinyPack [2], and Z-compression [6] and FELACS[7]. The datasets we use in the comparison is the logged data from our demo WSN. There are also some pre-loaded datasets including Intel Lab data, ZebraNet data and accelerometer data.

4. CONCLUSIONS

This testbed demo shows that Z-compression could achieve competitive compression ratio compared with the following three well-known compression schemes. The Z-compression is spatial independent and can be easily adapted to the WSN applications that have multi-type of sensors fetching multi-modal sensor data. The light weight, fast compressiion and decoding speed of the Z-compression make it suitable for sensor applications which suffer from energy constraint and bandwidth limitations.

REFERENCES

[1] Xiaofei Cao, Sanjay Madria, and Takahiro Hara. Multi-model z-compression for high speed data streaming and low-power wireless sensor networks. *Distributed and Parallel Databases*, 2019.

[2] Tommy Szalapski and Sanjay Madria. On compressing data in wireless sensor networks for energy efficiency and real time delivery. *Distributed and Parallel Databases*, 31(2):151–182, 2013.

[3] Dimitrios Lymberopoulos, Nissanka B Priyantha, and Feng Zhao. Towards energy efficient design of multi-radio platforms for wireless sensor networks. In *Information Processing in Sensor Networks. IPSN'08. International Conference on*, 2008.

[4] Bhaskar Krishnamachari, Deborah Estrin, and Stephen Wicker. The impact of data aggregation in wireless sensor networks. In *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*, pages 575–578. IEEE, 2002.

[5] Francesco Marcelloni and Massimo Vecchio. An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks. *The Computer Journal*, 52(8):969–987, 2009.

[6] Xiaofei Cao, Sanjay Madria, and Takahiro Hara. Efficient z-order encoding based multi-modal data compression in wsns. In *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, pages 2185–2192. IEEE, 2017.

[7] Jonathan Gana Kolo, S Anandan Shanmugam, David Wee Gin Lim, and Li-Minn Ang. Fast and efficient lossless adaptive compression scheme for wireless sensor networks. *Computers & Electrical Engineering*, 41:275–287, 2015.

VII. A TESTBED FOR DATA ROUTING IN LOW-POWER WSNS USING DV-HOP BASED TRAJECTORY ENCODING ALGORITHM

Xiaofei Cao, Sanjay Madria

ABSTRACT

Trajectory-based routing is a common data forwarding protocol while collecting sensor data during a disaster or in a battlefield for situation-awareness. However, wireless sensor networks (WSNs) have limitations in available bandwidth and energy. The trajectory-based routing protocols could reduce redundant broadcasting to save energy and bandwidth significantly. In this demo, we will demonstrate the working of a DV-Hop (Distance Vector Hop) based trajectory encoding algorithm using virtual coordinates rather than using GPS data. Using the proposed trajectory based routing protocol, we achieve energy savings, reduced latency, reliability, better coverage, while routing data from a source node to a mobile sink.

1. INTRODUCTION AND PROBLEM STATEMENT

WSNs are common tool in many disaster and battlefield applications for collecting sensor data. For example, in a battlefield, soldiers patrolling a border area need to receive real-time sensing data from RF scanners about IEDs from the locations of interest. Any delay in receiving such data may endanger their lives. For some location-aware applications, location anonymity is crucial for safety of users as an adversary could easily infer users' locations like home address and working place and predict their mobility based on the history of using the location-aware services. Thus, the challenge is to collect sensor data by preserving the location anonymity.

Routing in low-power wireless sensor networks is challenging as the network topology changes when the intermediate nodes sleep/fail. The previous works to solve this problem proposed cluster based routing algorithms, table driven proactive routing protocols, and trajectory based routing protocols. Due to lower communication rate for low-power listening WSNs (wireless sensor networks), cluster based data routing algorithms with mobile sinks like [1], and [2] suffer from high latency and energy usage, and redundant radio communication. Other cluster-based approaches like LEACH [3] are limited due to the cluster head election overhead, and the short transmission range of the IEEE 802.15.4 radio limits.

The table driven proactive routing protocols like [4] causes high overhead on updating the routing tables. For the classic reactive routing protocol like AODV [5], the higher energy consumption, and the broadcast storm issue make it unscalable. To alleviate the broadcast storm problem, variety of counter-based broadcast approaches like [6], and [7] have been proposed. However, they still don't provide the remedy as they reduce only about 60% of the redundant re-broadcasting in ideal-conditions (like known location of every node in the WSN) [6]. Although the location-based broadcast approaches like [8] reduces the broadcast overhead, they require GPS which restricts their usage in the battery limited applications. There are many trajectory encoding methods proposed earlier which route packets through wireless sensor nodes that lies more or less on the designed trajectory. The paper [9] uses the cubic Bezier curve, to approximate the desired routing path. In [10], the authors encode the trajectory using a sequence of linear functions which simplifies the encoding computation but increases decoding complexity. In [11], a sine wave trajectory encoding is proposed which takes the advantage of using trigonometric function as a single sine function representing a curve with two convex points. Further more, ring routing [12], and circular routing [13] have overheads due to inefficient handling of node failures. All the above algorithms uses 2d coordinates for trajectory encoding.

Virtual coordinate system is an option for geographic routing without using GPS. It can use local connectivity information including the number of neighbors of each node and the perimeter nodes' location as in [14]. It can also use the anchor nodes and the vector of minimum hop distance (DV-Hop) to the anchor nodes to estimate the distance between nodes as shown in [13].

To address the above shortcomings for routing in low power WSNs, we proposed a hybrid trajectory-based routing (TBR) [15] protocol which provides low latency of the proactive routing, and less maintenance overhead of the reactive routing. Instead of using the exact nodes' location information from GPS as in the classic TBRs like [9], [10], [11], we only use a vector of the minimal distance of hops (DV-Hop) to all the virtual anchor nodes initialized once in the WSN setup phase. The routing trajectory can be represented as a list of hop constraints to the anchor nodes. Also, our routing message only contains two basic geometric shapes hyperbola and arc which can be represented with two simple mathematical equations. It is different from the classic Cartesian based TBRs as we avoid the complex geometric computing at the sensor nodes which makes it suitable for the WSN environment of low-power and low-computing resources. We provide location anonymity by avoiding the use of GPS, and avoiding to transmit the location information. We also address the broadcast storm issues by integrating the counter-based broadcasting.

This demo shows how the proposed DV-Hop based trajectory encoding work, and demonstrates the routing procedure using the routing messages with the encoded trajectory.

2. SIMULATION AND TESTBED IMPLEMENTATION

There is no special requirement for space and need only Internet connection for this demo. The simulation website is hosted in the lab server which has 40 cores, 192 GB RAM, and three GTX 1060 GPUs. The estimated parallel computation power is 13 TFLOPs.

2.1. SIMULATION AND VISUALIZATION

The objective of this demo is to encode a user drawing trajectory using the DV-Hop based trajectory encoding algorithm and then simulate the routing procedure in a simulated Wireless sensor networks. The front end of the demo takes the following responsibility. Firstly, it helps the user to generate a wireless sensor network with the user defined number of sensor nodes, number of anchor nodes, wireless communication range of sensors, and the length and width of the sensor field. Secondly, it enables the user to draw the routing trajectory and save the trajectory as a set of pixels. Thirdly, it has the ability to check the information of each individual nodes and shown on the screen. Fourthly, it transfers the sensors list and the pixels' set to the server and receive the routing message calculated by the back-end program. Fifthly, it is able to display the estimated routing trajectory calculated by the routing message. Last, the routing procedure is translated from routing file to the specific animation of routing activity in the order of time and displayed on the screen. The Figure 1 shows the control box that help the user generating the Wireless sensor networks. The Figure 2 shows the menu of the demo UI and how the front-end displays the sensor nodes, and to query the detail information of each sensor node. The anchor nodes are shown as small read dots while the normal sensor nodes appear as black dots. The Figure 3 shows the menu of visualizing the encoded results and routing simulation.

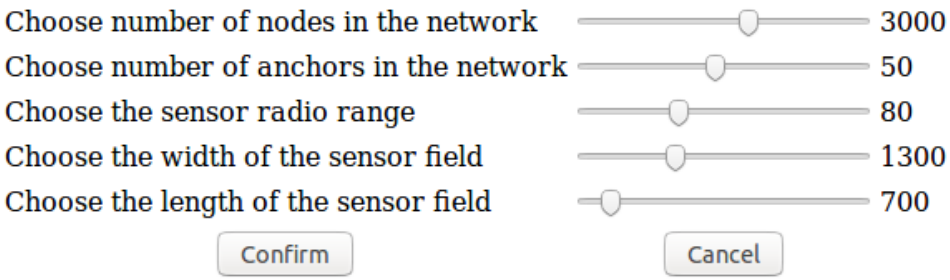


Figure 1. The control box setting up the WSNs

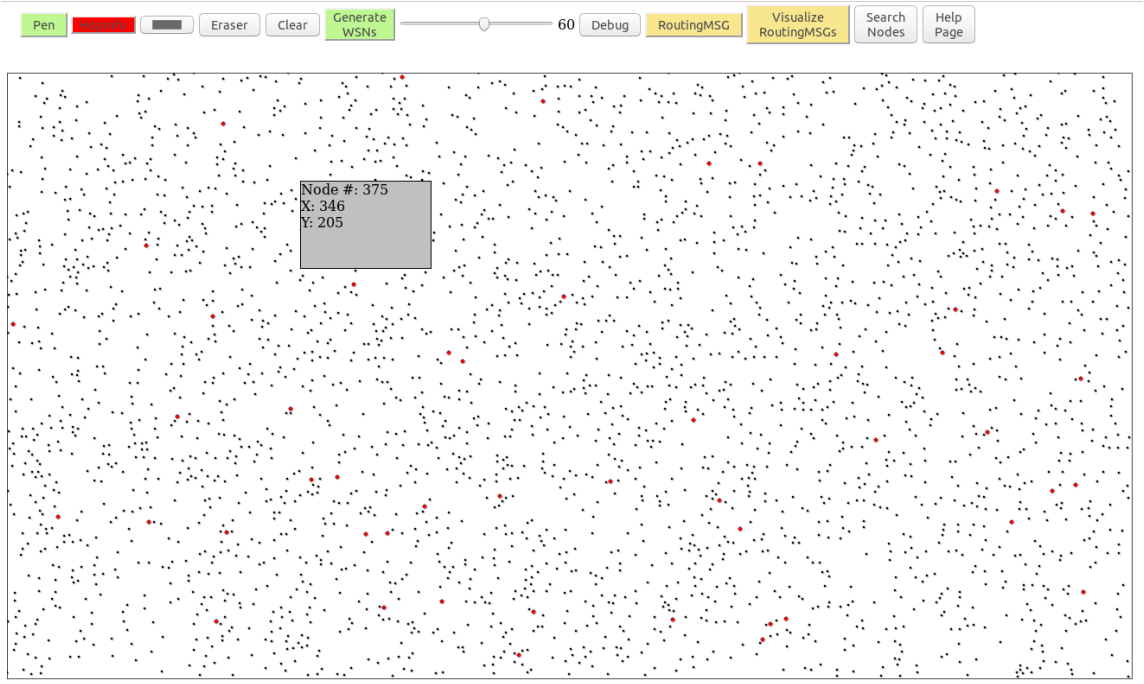


Figure 2. The visual effect of a sample WSN and the detail node information of a sensor

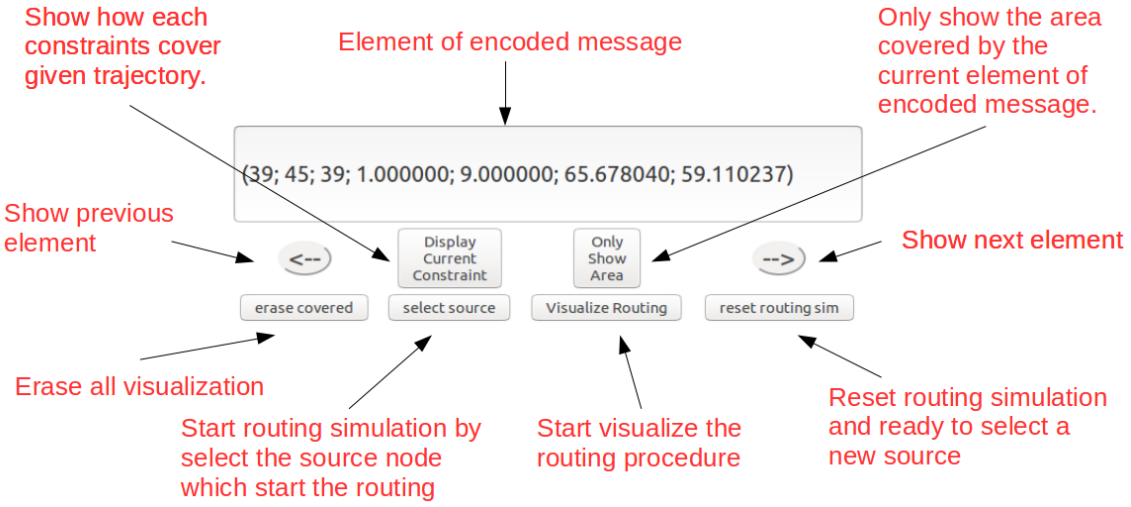
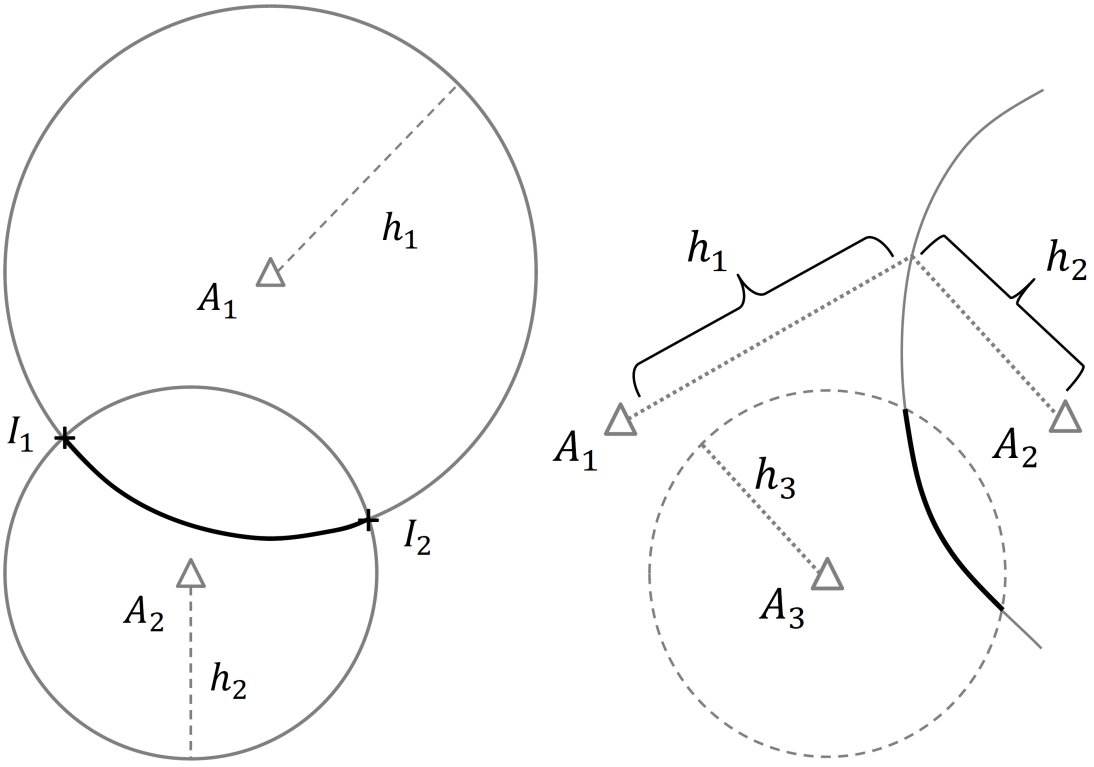


Figure 3. The routing visualization menu

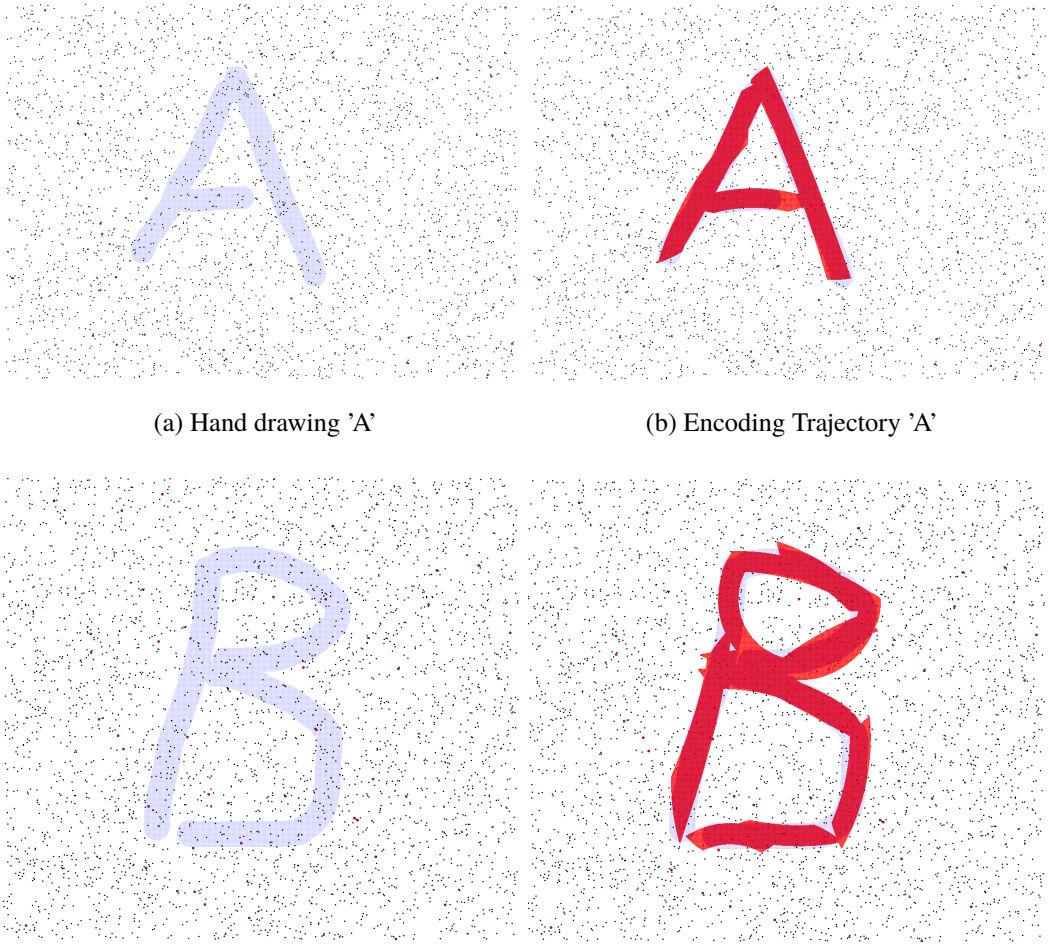
The back-end of the simulation program cooperates with the front-end to do most of the computation tasks including calculating vector of the minimal distance of hops of sensors for each anchor node, the routing message with hop constraints, and the simulation activities based on the routing message. The DV-Hop based trajectory encoding algorithm uses arc segments and hyperbola segments to represent the routing trajectory as shown in Figure 4. To calculate the best combination of arc segments and hyperbola segments that maximizing the coverage ratio while minimizing the redundant broadcasting, we use GPUs and CUDA to excute the proposed greedy algorithm in a parallel fashion. Figure 5 shows the samples of hand drawn trajectory and the visualization of the encoded trajectory.



(a) An arc trajectory

(b) A hyperbola trajectory

Figure 4. Example of an arc and hyperbola trajectory represented with hop constraints



(a) Hand drawing 'A'

(b) Encoding Trajectory 'A'

(c) Hand drawing 'B'

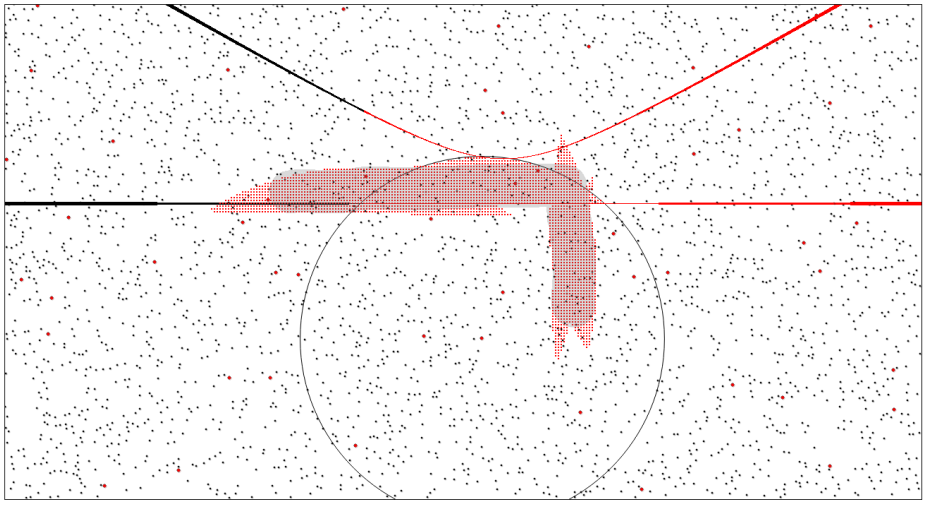
(d) Encoding Trajectory 'B'

Figure 5. Example of a circular and arc trajectory represented with hop constraints

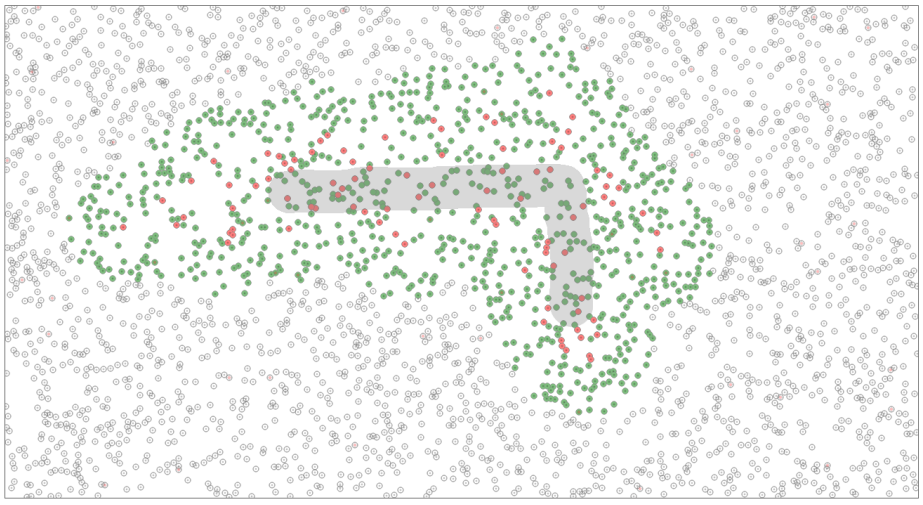
2.2. DEMO SHOWN

In this demo, we will ask users to select an area to be tracked for sensor data, select sensors nodes, define wireless radio range, select n anchor nodes where each anchor node will flood information to at most r hops, and the TAS will have m entries where TAS stands for the Trajectory Area Set which is the set containing all the pixels which the trajectory covers. With the current hardware used in this demo, it takes less than one minute to calculate the routing constraints in a WSN of 5000 nodes with 50 anchor nodes.

To evaluate the performance of routing with the trajectory message and display the routing procedure step by step, the server side will run TOSSIM [16] to simulate the routing procedure based on the user defined WSN. The visualized result will be sent back to the front end and displayed as an animation. The routing delay will be stored and can be referred by the user later after the animation is finished. Figure 6 shows a sample output of the routing procedure.



(a) Encoded trajectory



(b) Simulated actual route (in red)

Figure 6. Example of a encoded routing trajectory and simulated routing trajectory

3. CONCLUSIONS

In this demo, we show how a routing trajectory can be encoded efficiently without using GPS by applying the proposed DV-Hop based trajectory encoding algorithm. The result shows that our trajectory based routing algorithm could reduce redundant broadcasting and minimizes latency. The link of the demo is: http://www.routing-demos.com:8080/.

REFERENCES

[1] Abdul Waheed Khan, Abdul Hanan Abdullah, Mohammad Abdur Razzaque, and Javed Iqbal Bangash. Vgdra: a virtual grid-based dynamic routes adjustment scheme for mobile sink-based wireless sensor networks. *IEEE sensors journal*, 15(1):526–534, 2015.

[2] Lei Shi, Zheng Yao, Baoxian Zhang, Cheng Li, and Jian Ma. An efficient distributed routing protocol for wireless sensor networks with mobile sinks. *International Journal of Communication Systems*, 28(11):1789–1804, 2015.

[3] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System sciences. Proceedings of the 33rd annual Hawaii international conference on*, pages 10–pp. IEEE, 2000.

[4] Shio Kumar Singh, MP Singh, Dharmendra K Singh, et al. Routing protocols in wireless sensor networks–a survey. *International Journal of Computer Science & Engineering Survey (IJCSES)*, 1(2):63–83, 2010.

[5] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. Ad hoc on-demand distance vector (aodv) routing. Technical report, 2003.

[6] Ji-Young Jung and Dong-Yoon Seo. Counter-based broadcast scheme considering reachability, network density, and energy efficiency for wireless sensor networks. *Sensors*, 18(1):120, 2018.

[7] Kok-Poh Ng, Charalampos Tsimenidis, and Wai Lok Woo. C-sync: Counter-based synchronization for duty-cycled wireless sensor networks. *Ad Hoc Networks*, 61:51–64, 2017.

[8] Byungseok Kang and Hyunseung Choo. An energy-efficient routing scheme by using gps information for wireless sensor networks. *International Journal of Sensor Networks*, 26(2):136–143, 2018.

[9] Murat Yuksel, Ritesh Pradhan, and Shivkumar Kalyanaraman. An implementation framework for trajectory-based routing in ad hoc networks. *Ad Hoc Networks*, 4(1):125–137, 2006.

[10] Houda Labiod, Nedal Ababneh, and Miguel García de la Fuente. An efficient scalable trajectory based forwarding scheme for vanets. In *Advanced Information Networking and Applications (AINA), 24th IEEE International Conference on*, pages 600–606. IEEE, 2010.

[11] Badri Nath and Dragoş Niculescu. Routing on a curve. *ACM SIGCOMM Computer Communication Review*, 33(1):155–160, 2003.

[12] Can Tunca, Sinan Isik, Mehmet Yunus Donmez, and Cem Ersoy. Ring routing: An energy-efficient routing protocol for wireless sensor networks with a mobile sink. *IEEE Transactions on Mobile Computing*, 14(9):1947–1960, 2015.

[13] Rouhollah Rahmatizadeh, Saad Ahmad Khan, Anura P Jayasumana, Damla Turgut, and Ladislau Boloni. Circular update directional virtual coordinate routing protocol in sensor networks. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2015.

[14] Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 96–108. ACM, 2003.

[15] Xiaofei Cao and Sanjay Madria. An efficient trajectory-based routing using virtual coordinates for low-power wsns with mobile sinks. In *2019 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2019.

[16] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137. ACM, 2003.

SECTION

3. CONCLUSION AND FUTURE WORK

In this section, we conclude the research objectives and results reported in this dissertation. At the end, we also list some interesting problems for the future work related to this research.

3.1. RESEARCH OBJECTIVES ADDRESSED

This research reported in this research aimed to develop the efficient data and resource management frameworks for wireless networks including wireless sensor networks (WSNs), wireless actuator networks (WANs), and Internet of things (IoTs). Real-world wireless network applications require comprehensive solutions that could solve the challenge of energy and bandwidth limitation, security and privacy, and meeting the QoS requirements. In this research, we studied three typical applications including the environment data monitoring application in WSNs, the data dissemination application with mobile sensors, and the object tracking application in IoT environment.

For the static WSNs applications monitoring the environment data, the objective is to reduce the energy consumption while not compromise the QoS. Since batteries are the typical power source for wireless sensors, the energy consumption is the primary constraint in the design of WSNs. Many research efforts have shown that radio communication, including the data transmission and channel listening, is the predominant factor among all the energy consumption metrics of the WSNs.

For the applications with mobile sensor nodes, the objective is to disseminate data reliably and securely. In most of the WSN applications, sensors report their sensing data periodically. However, periodically sensing has a long delay which is not desirable for applications requiring real-time low-distorted data. Also, in WSNs with mobile nodes, to improve the reliability of disseminating sensing data is a challenge due to the dynamical topology they have. The edge/fog networks interacting with local wireless sensor networks (WSNs) provide services with higher reliability in managing data and resources locally. Also, the latency of data dissemination in mobile edge networks (MEN) is reduced because edge nodes can collect and process data faster than the remote cloud. However, with the participation of third party mobile edge devices, security is another important challenge which need to be addressed.

For the object tracking application, the objective is to correctly activate the senors on the trajectory of the moving object. In the mean time, the energy consumption and location anonymity also need to be considered. As not all sensors contribute equally to target tracking, activating those sensors that contribute most to the tracking could not only increase the tracking accuracy but also save energy. In some military applications, sensors also need to prevent being detected by other enemy targets.

3.2. THE MAIN CONTRIBUTIONS

In this research, we first proposed a Z-order [31] encoding based data compression scheme. The Z-order encoding called Z-compression can compress multi-modal sensing data at each leaf node as well as at the intermediate nodes efficiently in near real-time. The Z-compression algorithm can encode multi-modal sensor data like precipitation, water level, and wind speed (needed to detect a flood risk in a region) into a binary stream. Using our Z-compression algorithm in a WSN with a hierarchical topology [32], the nodes with limited bandwidth can tolerate higher-stream data rates coming from upstream nodes by concatenating compressed sensor data into the reduced number of packets which may be as

large as permissible by the network protocol. The proposed Z-compression algorithm also uses temporal and spatial data locality and delta encoding for better performance. Instead of using Huffman style coding which requires extra bits for each delta values, Z-order encoding is used to compress the delta values of all attributes of the input data into a binary stream. The predefined decoding rules are used to decode the Z-values and extract all the values of attributes. We conducted extensive experiments using skewed and unskewed real datasets and find that Z-order encoding based compression performs better than Huffman tree based source coding approaches. Also, we optimized the original Z-order encoding, where, for skewed datasets, we proposed the initial code library to improve the compression performance further. Our experiments show that it has much better compression ratios for the multi-dimensional datasets than the previous Huffman coding based compression approaches like LEC [41], TinyPack [33], Adaptive-LEC [42] and FELACS [44].

Second, we proposed a data dissemination scheme which enables the fog server to directly collect/send only the necessary data for the edge clients through predefined trajectories. The data packet from mobile edge devices like cellphones that are near the client's position of interest (POI) should be able to reach the targeted IoT devices (usually the wireless sensor motes) with minimum overhead and latency. Also, a data collection algorithm are proposed that have both low latency and less overhead of redundant broadcasting. Instead of using the exact nodes' location information from GPS as in [34][35][36], we only use a vector of the minimal distance of hops (DV-Hops) to all the anchor nodes selected by the secure fog server. The area of position of interest (POI) can be represented as a list of hop constraints to the anchor nodes. Our routing message only contains two basic geometric shapes: hyperbola and arc. These shapes can be represented with two simple mathematical equations. The sensor nodes could avoid the complex geometric computing, which makes it suitable for WSAN that have low-power and low-computing resources. In addition, the proposed scheme provides location anonymity by avoiding using and transmission of the GPS location information. To encode the POI, a

trajectory encoding algorithm for IoT applications is proposed that have improved energy efficiency, reduced latency, and achieve reliable performance when fetching data from the POI in the local fog network without using GPS coordinates. It uses geometry shapes to approximate the complex trajectory which achieve good compression ratio compare to JPEG. In addition, with the use of virtual coordinates, location anonymity is achieved for the source, sink, and intermediate nodes in the routing path, as only the secure server in the local fog knows the anchor nodes' locations. Besides, the use of ellipse and hyperbola constraints increase the encoding accuracy and compression ratio.

Third, we proposed an efficient Seq2Seq learning model to predict the future trajectory of a moving object using WSNs. The proposed Seq2Seq model predicts and generates the DV-Hop based routing constraints without knowing the location of the anchor nodes in the object tracking area. Therefore, it enables sensors to wake up before the target passes by a given area and sleep after. The proposed framework decouples the data plane and the control plane by using the constraint-based routing protocol that enables different applications to share the same WSN. The Seq2Seq based hyperbola constraint generation model speeds up the computation significantly and thus it enables a real-time control-message generation for object tracking. It also allows any edge device to participate in the target tracking applications as location information is hidden.

3.3. THE FUTURE WORK

In the future, we plan to study the adapted anchor nodes deployment strategy for different IoT environment and application requirements. To adapt the anchor-based virtual coordinate system fitting the needs of network topology without physical anchors, such as underwater sonar system, and hash environment networks. Thereby, extending the applicability of the framework, and improve the location privacy of the anchors. The anchor deployment problem is a challenge ignored in previous research works. In fact, many real-world applications are unable to maintain physical anchors or difficult to set up the DV-Hop

table for all IoT devices through the anchors' broadcast. Our research can be extended to design three different substitution methods for deploying anchors. First, we plan to use the landmarks and the distance from the IoT devices to the landmarks as the virtual coordinates. It expands the usability of the proposed framework for a sparse IoT network that is deployed in a vast geospatial area. Second, we plan to use the light, sound, or microwave sources and the signal strength of those signal sources as the virtual coordinates. It expands the usability of the proposed framework for a dense indoor IoT network and allows precision trajectory encoding with continuous signal strength level rather than discrete hop counts. Third, we plan to use the activate sonar array and the receiving signal phase as the virtual coordinates. It expands the usability of the proposed framework for underwater sonar network.

Also, we plan to create an online training transformer network model that could achieve better prediction accuracy by adapting our seq2seq-based trajectory prediction framework. Using online training reduces the requirement of the size of the training dataset which makes the framework more practical.

REFERENCES

[1] Sanjay Madria, Vimal Kumar, and Rashmi Dalvi. Sensor cloud: A cloud of virtual sensors. *Software, IEEE*, 31(2):70–77, 2014.

[2] John Burgess, John Zahorjan, Ratul Mahajan, et al. CRAWDAD dataset umass/diesel (v. 2008-09-14), September 2008.

[3] Mare Srbinovska, Cvetan Gavrovski, Vladimir Dimcev, Aleksandra Krkoleva, and Vesna Borozan. Environmental parameters monitoring in precision agriculture using wireless sensor networks. *Journal of cleaner production*, 88:297–307, 2015.

[4] Fan Wu, Christoph Rudiger, and Mehmet Rasit Yuce. Design and field test of an autonomous iot wsn platform for environmental monitoring. In *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–6. IEEE, 2017.

[5] Shu Shen, Lijuan Sun, Yibo Dang, Zhiqiang Zou, and Ruchuan Wang. Node localization based on improved pso and mobile nodes for environmental monitoring wsns. *International Journal of Wireless Information Networks*, 25(4):470–479, 2018.

[6] Jue Yang, Chengyang Zhang, Xinrong Li, Yan Huang, Shengli Fu, and Miguel F Acevedo. Integration of wireless sensor networks in environmental monitoring cyber infrastructure. *Wireless Networks*, 16(4):1091–1108, 2010.

[7] Biljana Risteska Stojkoska, Andrijana Popovska Avramova, and Periklis Chatzimisios. Application of wireless sensor networks for indoor temperature regulation. *International Journal of Distributed Sensor Networks*, 10(5):502419, 2014.

[8] Baoqiang Kan, Li Cai, and Lei Zhao. An accurate energy model for wsn node and its optimal design. In *2007 International Conference on Communications, Circuits and Systems*, pages 328–332. IEEE, 2007.

[9] Hai-Ying Zhou, Dan-Yan Luo, Yan Gao, and De-Cheng Zuo. Modeling of node energy consumption for wireless sensor networks. *Wireless Sensor Network*, 3(1):18, 2011.

[10] Mehmet C Vuran, Vehbi C Gungor, and Ozgür B Akan. On the interdependency of congestion and contention in wireless sensor networks. In *Proc. SENMETRICS'05*, pages 136–147, 2005.

[11] Yong Wang, Pei Zhang, Ting Liu, Chris Sadler, and Margaret Martonosi. CRAWDAD dataset princeton/zebranet (v. 2007-02-14). Downloaded from http://crawdad.org/princeton/zebranet/20070214, February 2007.

[12] Yeran Sun, Yashar Moshfeghi, and Zhang Liu. Exploiting crowdsourced geographic information and gis for assessment of air pollution exposure during active travel. *Journal of Transport & Health*, 6:93–104, 2017.

[13] Yeran Sun and Amin Mobasheri. Utilizing crowdsourced data for studies of cycling and air pollution exposure: A case study using strava data. *International journal of environmental research and public health*, 14(3):274, 2017.

[14] Grant R McKercher, Jennifer A Salmond, and Jennifer K Vanos. Characteristics and applications of small, portable gaseous air pollution monitors. *Environmental Pollution*, 223:102–110, 2017.

[15] Naonori Ueda and Futoshi Naya. Spatio-temporal multidimensional collective data analysis for providing comfortable living anytime and anywhere. *APSIPA Transactions on Signal and Information Processing*, 7, 2018.

[16] Vishal Sharma, Fei Song, Ilsun You, and Mohammed Atiquzzaman. Energy efficient device discovery for reliable communication in 5g-based iot and bsns using unmanned aerial vehicles. *Journal of Network and Computer Applications*, 97:79–95, 2017.

[17] Wei Feng, Jingchao Wang, Yunfei Chen, Xuanxuan Wang, Ning Ge, and Jianhua Lu. Uav-aided mimo communications for 5g internet of things. *IEEE Internet of Things Journal*, 6(2):1731–1740, 2018.

[18] Shivani Rajendra Teli, Stanislav Zvanovec, and Zabih Ghassemlooy. Optical internet of things within 5g: Applications and challenges. In *2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)*, pages 40–45. IEEE, 2018.

[19] Mohamed Hamdi, Noureddine Boudriga, and Mohammad S Obaidat. Whomoves: An optimized broadband sensor network for military vehicle tracking. *International Journal of Communication Systems*, 21(3):277–300, 2008.

[20] Hong Zhang, Zeyu Zhang, Lei Zhang, Yifan Yang, Qiaochu Kang, and Daniel Sun. Object tracking for a smart city using iot and edge computing. *Sensors*, 19(9):1987, 2019.

[21] Peter Corke, Carrick Detweiler, Matthew Dunbabin, Michael Hamilton, Daniela Rus, and Iuliu Vasilescu. Experiments with underwater robot localization and tracking. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4556–4561. IEEE, 2007.

[22] Injong Rhee, Ajit Warrier, Mahesh Aia, Jeongki Min, and Mihail L Sichitiu. Z-mac: a hybrid mac for wireless sensor networks. *IEEE/ACM Transactions On Networking*, 16(3):511–524, 2008.

[23] Taneli Riihonen, Dani Korpi, Matias Turunen, and Mikko Valkama. Full-duplex radio technology for simultaneously detecting and preventing improvised explosive device activation. In *2018 International Conference on Military Communications and Information Systems (ICMCIS)*, pages 1–4. IEEE, 2018.

[24] Shrawan Kumar and DK Lobiyal. Novel dv-hop localization algorithm for wireless sensor networks. *Telecommunication Systems*, 2017.

[25] Xingjuan Cai, Penghong Wang, Lei Du, Zhihua Cui, Wensheng Zhang, and Jinjun Chen. Multi-objective three-dimensional dv-hop localization algorithm with nsga-ii. *IEEE Sensors Journal*, 19(21):10003–10015, 2019.

[26] Penghong Wang, Jianrou Huang, Zhihua Cui, Liping Xie, and Jinjun Chen. A gaussian error correction multi-objective positioning model with nsga-ii. *Concurrency and Computation: Practice and Experience*, 32(5):e5464, 2020.

[27] Xiaofei Cao, Sanjay Madria, and Takahiro Hara. Efficient z-order encoding based multi-modal data compression in wsns. In *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, pages 2185–2192. IEEE, 2017.

[28] Xiaofei Cao, Sanjay Madria, and Takahiro Hara. Multi-model z-compression for high speed data streaming and low-power wireless sensor networks. *Distributed and Parallel Databases*, 2019.

[29] Xiaofei Cao and Sanjay Madria. Efficient geospatial data collection in iot networks for mobile edge computing. In *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, pages 1–10. IEEE, 2019.

[30] Xiaofei Cao and Sanjay Madria. Efficient data collection in iot networks using trajectory encoded with geometric shapes. In *Trans. on Sustainable Computing Special Issue on Energy-Efficient Edge Computing 2020*. IEEE, Under reviewing.

[31] Guy M Morton. *A computer oriented geodetic data base and a new technique in file sequencing*. International Business Machines Company New York, 1966.

[32] Bret Hull, Kyle Jamieson, and Hari Balakrishnan. Mitigating congestion in wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 134–147. ACM, 2004.

[33] Tommy Szalapski and Sanjay Madria. On compressing data in wireless sensor networks for energy efficiency and real time delivery. *Distributed and Parallel Databases*, 31(2):151–182, 2013.

[34] Murat Yuksel, Ritesh Pradhan, and Shivkumar Kalyanaraman. An implementation framework for trajectory-based routing in ad hoc networks. *Ad Hoc Networks*, 4(1):125–137, 2006.

[35] Houda Labiod, Nedal Ababneh, and Miguel García de la Fuente. An efficient scalable trajectory based forwarding scheme for vanets. In *Advanced Information Networking and Applications (AINA), 24th IEEE International Conference on*, pages 600–606. IEEE, 2010.

[36] Badri Nath and Dragoş Niculescu. Routing on a curve. *ACM SIGCOMM Computer Communication Review*, 33(1):155–160, 2003.

[37] Tossaporn Srisooksai, Kamol Keamarungsi, Poonlap Lamsrichan, and Kiyomichi Araki. Practical data compression in wireless sensor networks: A survey. *Journal of network and computer applications*, 35(1):37–59, 2012.

[38] Muneer Bani Yassein, Sanabel Fathi Nimer, and Ahmed Y Al-Dubai. A new dynamic counter-based broadcasting scheme for mobile ad hoc networks. *Simulation Modelling Practice and Theory*, 19(1):553–563, 2011.

[39] David A Huffman et al. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[40] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[41] Francesco Marcelloni and Massimo Vecchio. An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks. *The Computer Journal*, 52(8):969–987, 2009.

[42] Massimo Vecchio, Raffaele Giaffreda, and Francesco Marcelloni. Adaptive lossless entropy compressors for tiny iot devices. *IEEE Transactions on Wireless Communications*, 13(2):1088–1100, 2014.

[43] Tommy Szalapski and Sanjay Madria. Energy efficient distributed grouping and scaling for real-time data compression in sensor networks. In *2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)*, pages 1–9. IEEE, 2014.

[44] Jonathan Gana Kolo, S Anandan Shanmugam, David Wee Gin Lim, and Li-Minn Ang. Fast and efficient lossless adaptive compression scheme for wireless sensor networks. *Computers & Electrical Engineering*, 41:275–287, 2015.

[45] Christopher M Sadler and Margaret Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 265–278, 2006.

[46] Terry A Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, 1984.

[47] Nguyen Quoc Viet Hung, Hoyoung Jeung, and Karl Aberer. An evaluation of model-based approaches to sensor data compression. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2434–2447, 2013.

[48] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM SIGMOD Record*, 30(2):151–162, 2001.

[49] Chiranjeeb Buragohain, Nisheeth Shrivastava, and Subhash Suri. Space efficient streaming algorithms for the maximum error histogram. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 1026–1035. IEEE, 2007.

[50] Hazem Elmeleegy, Ahmed K Elmagarmid, Emmanuel Cecchet, Walid G Aref, and Willy Zwaenepoel. Online piece-wise linear approximation of numerical streams with precision guarantees. *Proceedings of the VLDB Endowment*, 2(1):145–156, 2009.

[51] G Kumar, K Baskaran, R Elijah Blessing, and M Lydia. A comprehensive review on the impact of compressed sensing in wireless sensor networks. *International Journal on Smart Sensing & Intelligent Systems*, 9(2), 2016.

[52] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless networks*, 8(2-3):153–167, 2002.

[53] Chien Chen, Chin-Kai Hsu, and Hsien-Kang Wang. A distance-aware counter-based broadcast scheme for wireless ad hoc networks. In *Military Communications Conference, MILCOM*. IEEE, 2005.

[54] Ji-Young Jung and Dong-Yoon Seo. Counter-based broadcast scheme considering reachability, network density, and energy efficiency for wireless sensor networks. *Sensors*, 18(1):120, 2018.

[55] Kok-Poh Ng, Charalampos Tsimenidis, and Wai Lok Woo. C-sync: Counter-based synchronization for duty-cycled wireless sensor networks. *Ad Hoc Networks*, 61:51–64, 2017.

[56] M Bani Yassein, A Al-Dubai, M Ould Khaoua, and Omar M Al-Jarrah. New adaptive counter based broadcast using neighborhood information in manets. In *2009 IEEE International Symposium on Parallel & Distributed Processing*, pages 1–7. IEEE, 2009.

[57] Thang Le Duc, Duc Tai Le, Vyacheslav V Zalyubovskiy, Dongsoo S Kim, and Hyunseung Choo. Level-based approach for minimum-transmission broadcast in duty-cycled wireless sensor networks. *Pervasive and Mobile Computing*, 27:116–132, 2016.

[58] Suraj Sharma, Deepak Puthal, Sabah Tazeen, Mukesh Prasad, and Albert Y Zomaya. Msgr: A mode-switched grid-based sustainable routing protocol for wireless sensor networks. *IEEE Access*, 5:19864–19875, 2017.

[59] Levente Buttyán and Péter Schaffer. Position-based aggregator node election in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 6(1):679205, 2010.

[60] Robert Akl and Uttara Sawant. Grid-based coordinated routing in wireless sensor networks. In *2007 4th IEEE Consumer Communications and Networking Conference*, pages 860–864. Citeseer, 2007.

[61] Omar Banimelhem and Samer Khasawneh. Gmcar: Grid-based multipath with congestion avoidance routing protocol in wireless sensor networks. *Ad Hoc Networks*, 10(7):1346–1361, 2012.

[62] Haiyun Luo, Fan Ye, Jerry Cheng, Songwu Lu, and Lixia Zhang. Ttdd: Two-tier data dissemination in large-scale wireless sensor networks. *Wireless networks*, 11(1-2):161–175, 2005.

[63] Juan A Sanchez, Pedro M Ruiz, and Ivan Stojmnenovic. Gmr: Geographic multicast routing for wireless sensor networks. In *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, volume 1, pages 20–29. IEEE, 2006.

[64] Yuan-Po Chi and Hsung-Pin Chang. An energy-aware grid-based routing scheme for wireless sensor networks. *Telecommunication Systems*, 54(4):405–415, 2013.

[65] Abdul Waheed Khan, Javed Iqbal Bangash, Adnan Ahmed, and Abdul Hanan Abdullah. Qdvgdd: Query-driven virtual grid based data dissemination for wireless sensor networks using single mobile sink. *Wireless Networks*, (1):241–253, 2019.

[66] Can Tunca, Sinan Isik, Mehmet Yunus Donmez, and Cem Ersoy. Ring routing: An energy-efficient routing protocol for wireless sensor networks with a mobile sink. *IEEE Transactions on Mobile Computing*, 14(9):1947–1960, 2015.

[67] Ramin Yarinezhad. Reducing delay and prolonging the lifetime of wireless sensor network using efficient routing protocol based on mobile sink and virtual infrastructure. *Ad Hoc Networks*, 84, 2019.

[68] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System sciences. Proceedings of the 33rd annual Hawaii international conference on*, pages 10–pp. IEEE, 2000.

[69] Ossama Younis and Sonia Fahmy. Heed: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *Mobile Computing, IEEE Transactions on*, 3(4):366–379, 2004.

[70] Khalid Haseeb, Kamalrulnizam Abu Bakar, Abdul Hanan Abdullah, and Tasneem Darwish. Adaptive energy aware cluster-based routing protocol for wireless sensor networks. *Wireless Networks*, 23(6):1953–1966, 2017.

[71] Peyman Neamatollahi, Mahmoud Naghibzadeh, and Saeid Abrishami. Fuzzy-based clustering-task scheduling for lifetime enhancement in wireless sensor networks. *IEEE Sensors Journal*, 17(20):6837–6844, 2017.

[72] Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 96–108. ACM, 2003.

[73] Hongyang Chen, Kaoru Sezaki, and Ping Deng. An improved dv-hop localization algorithm with reduced node location error for wireless sensor networks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 91(8):2232–2236, 2008.

[74] Kai Chen, Zhong-hua Wang, Mei Lin, and Min Yu. An improved dv-hop localization algorithm for wireless sensor networks. 2010.

[75] Shrawan Kumar and DK Lobiyal. An advanced dv-hop localization algorithm for wireless sensor networks. *Wireless personal communications*, 71(2):1365–1385, 2013.

[76] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on networking*, 12(3):493–506, 2004.

[77] Tijs Van Dam and Koen Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 171–180, 2003.

[78] Injong Rhee, Ajit Warrier, Mahesh Aia, Jeongki Min, and Mihail L Sichitiu. Z-mac: a hybrid mac for wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 16(3):511–524, 2008.

[79] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, 2004.

[80] Enrico Perla, Art Ó Catháin, Ricardo Simon Carbajo, Meriel Huggard, and Ciarán Mc Goldrick. Powertossim z: realistic energy modelling for wireless sensor network environments. In *Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pages 35–42. ACM, 2008.

[81] Xiao-ping ZHANG and Gui-xiong LIU. Target tracking prediction in wsn based on quadratic polynomial motion modeling [j]. *Journal of Jinan University (Natural Science & Medicine Edition)*, 5, 2009.

[82] Kenji Okuma, Ali Taleghani, Nando De Freitas, James J Little, and David G Lowe. A boosted particle filter: Multitarget detection and tracking. In *European conference on computer vision*, pages 28–39. Springer, 2004.

[83] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter, 1995.

[84] H. Zhang, X. Zhou, Z. Wang, H. Yan, and J. Sun. Adaptive consensus-based distributed target tracking with dynamic cluster in sensor networks. *IEEE Transactions on Cybernetics*, 49(5):1580–1591, 2019.

[85] Gharavian FayaziBarjini. Target tracking in wireless sensor networks using ngekf algorithm. *J Ambient Intell Human Comput*, pages 3417–3429, 2019.

[86] Satish R Jondhale and Rajkumar S Deshpande. Kalman filtering framework-based real time target tracking in wireless sensor networks using generalized regression neural networks. *IEEE Sensors Journal*, 19(1):224–233, 2018.

[87] Tianzhu Zhang, Changsheng Xu, and Ming-Hsuan Yang. Multi-task correlation particle filter for robust object tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4335–4343, 2017.

[88] Tianzhu Zhang, Si Liu, Changsheng Xu, Bin Liu, and Ming-Hsuan Yang. Correlation particle filter for visual tracking. *IEEE Transactions on Image Processing*, 27(6):2676–2687, 2017.

[89] Gurjit Singh Walia, Ashish Kumar, Astitwa Saxena, Kapil Sharma, and Kuldeep Singh. Robust object tracking with crow search optimized multi-cue particle filter. *Pattern Analysis and Applications*, 23(3):1439–1455, 2020.

[90] Dhiren P Bhagat and Himanshukumar Soni. Target tracking using a hybrid kf-pso tracking model in wsn. In *International Conference on Emerging Technology Trends in Electronics Communication and Networking*, pages 83–98. Springer, 2020.

[91] Shalli Rani, Syed Hassan Ahmed, and Ravi Rastogi. Dynamic clustering approach based on wireless sensor networks genetic algorithm for iot applications. *Wireless Networks*, pages 1–10, 2019.

[92] Tung-Wei Kuo, Kate Ching-Ju Lin, and Ming-Jer Tsai. On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: Np-completeness and approximation algorithms. *IEEE Transactions on Computers*, 65(10):3109–3121, 2016.

[93] Chen-Xu Liu, Yun Liu, Zhen-Jiang Zhang, and Zi-Yao Cheng. High energy-efficient and privacy-preserving secure data aggregation for wireless sensor networks. *International Journal of Communication Systems*, 26(3):380–394, 2013.

[94] Dylan McDonald, Stewart Sanchez, Sanjay Madria, and Fikret Ercal. A survey of methods for finding outliers in wireless sensor networks. *Journal of network and systems management*, 23(1):163–182, 2015.

[95] Jeffrey Scott Vitter. Design and analysis of dynamic huffman codes. *Journal of the ACM (JACM)*, 34(4):825–845, 1987.

[96] Azad Ali, Abdelmajid Khelil, Piotr Szczytowski, and Neeraj Suri. An adaptive and composite spatio-temporal data compression approach for wireless sensor networks. In *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pages 67–76. ACM, 2011.

[97] Sorabh Gandhi, Suman Nath, Subhash Suri, and Jie Liu. Gamps: Compressing multi sensor data by grouping and amplitude scaling. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 771–784. ACM, 2009.

[98] Henry Ponti Medeiros, Marcos Costa Maciel, Richard Demo Souza, and Marcelo Eduardo Pellenz. Lightweight data compression in wireless sensor networks using huffman coding. *International Journal of Distributed Sensor Networks*, 2014.

[99] Dimitrios Lymberopoulos, Nissanka B Priyantha, and Feng Zhao. Towards energy efficient design of multi-radio platforms for wireless sensor networks. In *Information Processing in Sensor Networks. IPSN'08. International Conference on*, 2008.

[100] Bhaskar Krishnamachari, Deborah Estrin, and Stephen Wicker. The impact of data aggregation in wireless sensor networks. In *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*, pages 575–578. IEEE, 2002.

[101] S Madden. Intel berkeley research lab data, 2003.

[102] Mohit Jain, Ajeet Pal Singh, Soshant Bali, and Sanjit Kaul. CRAWDAD dataset jiit/accelerometer (v. 2012-11-03), November 2012.

[103] Richard M. Fujimoto, Randall Guensler, Michael P. Hunter, Hao Wu, Mahesh Palekar, Jaesup Lee, and Joonho Ko. CRAWDAD dataset gatech/vehicular (v. 2006-03-15). Downloaded from http://crawdad.org/gatech/vehicular/20060315, March 2006.

[104] Nelson I Dopico, Carlos Gil-Soriano, Iñigo Arrazola, and Santiago Zazo. Analysis of ieee 802.15. 4 throughput in beaconless mode on micaz under tinyos 2. In *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*, pages 1–5. IEEE, 2010.

[105] TelosB Datasheet. Crossbow Inc. Downloaded from http://www.memsic.com/userfiles/files/Datasheets/WSN, 2013.

[106] Enrico Perla, Art Ó Catháin, Ricardo Simon Carbajo, Meriel Huggard, and Ciarán Mc Goldrick. Powertossim z: realistic energy modelling for wireless sensor network environments. In *Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pages 35–42. ACM, 2008.

[107] Luca Anchora, Antonio Capone, Vincenzo Mighali, Luigi Patrono, and Francesco Simone. A novel mac scheduler to minimize the energy consumption in a wireless sensor network. *Ad Hoc Networks*, 16:88–104, 2014.

[108] Jun Long, Mianxiong Dong, Kaoru Ota, and Anfeng Liu. A green tdma scheduling algorithm for prolonging lifetime in wireless sensor networks. *IEEE Systems Journal*, 11(2):868–877, 2017.

[109] Michael Buettner, Gary V Yee, Eric Anderson, and Richard Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320. ACM, 2006.

[110] Yanjun Sun, Omer Gurewitz, and David B Johnson. Ri-mac: a receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 1–14. ACM, 2008.

[111] Jun Bum Lim, Beakcheol Jang, and Mihail L Sichitiu. Mcas-mac: A multichannel asynchronous scheduled mac protocol for wireless sensor networks. *Computer Communications*, 56:98–107, 2015.

[112] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137. ACM, 2003.

[113] HM Künzel, A Holm, D Zirkelbach, and AN Karagiozis. Simulation of indoor temperature and humidity conditions including hygrothermal interactions with the building envelope. *Solar Energy*, 78(4):554–561, 2005.

[114] Memsic Crossbow. Telosb v2 data sheet. Downloaded from www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf, 2008.

[115] Mehmet C Vuran, Özgür B Akan, and Ian F Akyildiz. Spatio-temporal correlation: theory and applications for wireless sensor networks. *Computer Networks*, 45(3):245–259, 2004.

[116] Mehmet C Vuran and Ian F Akyildiz. Spatial correlation-based collaborative medium access control in wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 14(2):316–329, 2006.

[117] Weisong Shi and Schahram Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

[118] Shanhe Yi, Zijiang Hao, Zhengrui Qin, and Qun Li. Fog computing: Platform and applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*. IEEE, 2015.

[119] Yingzi Wang, Nicholas Jing Yuan, Defu Lian, Linli Xu, Xing Xie, Enhong Chen, and Yong Rui. Regularity and conformity: Location prediction using heterogeneous mobility data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1275–1284. ACM, 2015.

[120] Xiaofei Cao. Serial port reader and writer for the android gateway. https://github.com/cxfcdcpu/gateway.

[121] Carl T Kelley. *Solving nonlinear equations with Newton's method*, volume 1. Siam, 2003.

[122] Taxi trajectory data. www.kaggle.com/crailtap/taxi-trajectory.

[123] Jing Wang, Jian Tang, Guoliang Xue, and Dejun Yang. Towards energy-efficient task scheduling on smartphones in mobile crowd sensing systems. *Computer Networks*, 115:100–109, 2017.

[124] Mamoun F Al-Mistarihi, Islam M Tanash, Fedaa S Yaseen, and Khalid A Darabkh. Protecting source location privacy in a clustered wireless sensor networks against local eavesdroppers. *Mobile Networks and Applications*, 25(1):42–54, 2020.

[125] Hao Wang, Guangjie Han, Chunsheng Zhu, Sammy Chan, and Wenbo Zhang. Tcslp: A trace cost based source location privacy protection scheme in wsns for smart cities. *Future Generation Computer Systems*, 107:965–974, 2020.

[126] Anfeng Liu, Xiao Liu, Zhipeng Tang, Laurence T Yang, and Zili Shao. Preserving smart sink-location privacy with delay guaranteed routing scheme for wsns. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(3):1–25, 2017.

[127] Timothy M Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & Computational Geometry*, 16(4):361–368, 1996.

[128] Gary B Hughes and Mohcine Chraibi. Calculating ellipse overlap areas. *Computing and visualization in science*, 15(5):291–301, 2012.

[129] TelosB Datasheet. Crossbow Inc. Downloaded from memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf.

[130] MicaZ. Crossbow Inc. Downloaded from memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf.

[131] Michael Buettner, Gary V Yee, Eric Anderson, and Richard Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320, 2006.

[132] Huihsin Tseng, Pi-Chuan Chang, Galen Andrew, Dan Jurafsky, and Christopher D Manning. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, 2005.

[133] Samer Samarah, Muhannad Al-Hajri, and Azzedine Boukerche. A predictive energy-efficient technique to support object-tracking sensor networks. *IEEE Transactions on Vehicular Technology*, 60(2):656–663, 2010.

[134] Khalid A Darabkh, Wijdan Y Albtoush, and Iyad F Jafar. Improved clustering algorithms for target tracking in wireless sensor networks. *The Journal of Supercomputing*, 73(5):1952–1977, 2017.

[135] Zhibo Wang, Wei Lou, Zhi Wang, Junchao Ma, and Honglong Chen. A hybrid cluster-based target tracking protocol for wireless sensor networks. *International Journal of Distributed Sensor Networks*, 9(3):494863, 2013.

[136] Chunming Wu, Chen Zhao, and Haoquan Gong. Energy-efficient target tracking algorithm for wsns. *3D Research*, 10(1):1, 2019.

[137] Chao Sha, Lian-hua Zhong, Yao Bian, Dan-dan Song, and Chun-hui Ren. A type of energy-efficient target tracking approach based on grids in sensor networks. *Peer-to-Peer Networking and Applications*, 12(5):1041–1060, 2019.

[138] Tao Liu, Qingrui Li, and Ping Liang. An energy-balancing clustering approach for gradient-based routing in wireless sensor networks. *Computer Communications*, 35(17):2150–2161, 2012.

[139] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[140] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[141] Abdul Waheed Khan, Abdul Hanan Abdullah, Mohammad Abdur Razzaque, and Javed Iqbal Bangash. Vgdra: a virtual grid-based dynamic routes adjustment scheme for mobile sink-based wireless sensor networks. *IEEE sensors journal*, 15(1):526–534, 2015.

[142] Lei Shi, Zheng Yao, Baoxian Zhang, Cheng Li, and Jian Ma. An efficient distributed routing protocol for wireless sensor networks with mobile sinks. *International Journal of Communication Systems*, 28(11):1789–1804, 2015.

[143] Shio Kumar Singh, MP Singh, Dharmendra K Singh, et al. Routing protocols in wireless sensor networks–a survey. *International Journal of Computer Science & Engineering Survey (IJCSES)*, 1(2):63–83, 2010.

[144] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. Ad hoc on-demand distance vector (aodv) routing. Technical report, 2003.

[145] Byungseok Kang and Hyunseung Choo. An energy-efficient routing scheme by using gps information for wireless sensor networks. *International Journal of Sensor Networks*, 26(2):136–143, 2018.

[146] Rouhollah Rahmatizadeh, Saad Ahmad Khan, Anura P Jayasumana, Damla Turgut, and Ladislau Boloni. Circular update directional virtual coordinate routing protocol in sensor networks. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2015.

[147] Xiaofei Cao and Sanjay Madria. An efficient trajectory-based routing using virtual coordinates for low-power wsns with mobile sinks. In *2019 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2019.

VITA

Xiaofei Cao was born in China. He received a Master of Science in Computer Science from Missouri University of Science and Technology in May 2015. He received a Master of Science in Electrical Engineering from Missouri University of Science and Technology in December 2012. He received a Bachelor of Science in Engineering from Tianjin University in July 2020.

He obtained his Doctor of Philosophy in Computer Science from Missouri University of Science and Technology under the supervision of Dr. Sanjay Madria in May 2021. His core research focus was on data management in wireless sensor networks. During his Ph.D. training, he published several papers in top-tier conference and journals. He also did a research internship at AFRL in Summer 2018.