Hindawi Publishing Corporation
ISRN Sensor Networks
Volume 2013, Article ID 253257, 18 pages
http://dx.doi.org/10.1155/2013/253257

Research Article

Robust Data Compression for Irregular Wireless Sensor Networks Using Logical Mapping

Thanh Dang,[1] Nirupama Bulusu,[2] and Wu-chi Feng[2]

[1] *School of Engineering and Computer Science, Washington State University Vancouver, Vancouver, WA 98686, USA*
[2] *Department of Computer Science, Portland State University, Portland, OR 97201, USA*

Correspondence should be addressed to Nirupama Bulusu; nbulusu@cs.pdx.edu

Received 12 March 2013; Accepted 28 March 2013

Academic Editors: C.-Y. Chow, T.-Y. Juang, B. Tavli, and Y. Yu

We propose RIDA, a novel robust information-driven data compression architecture for distributed wireless sensor networks. The key idea is to determine the data correlation among a group of sensors based on the data values to significantly improve compression performance rather than relying solely on spatial data correlation. A logical mapping approach assigns virtual indices to nodes based on the data content, which enables simple implementation of data transformation on resource-constrained nodes without any other information. We evaluate RIDA with both discrete cosine transform (DCT) and discrete wavelet transform (DWT) on publicly available real-world data sets. Our experiments show that 30% of energy and 80–95% of bandwidth can be saved for typical multihop data networks. Moreover, the original data can be retrieved after decompression with a low error of about 3%. In particular, for one state-of-the-art distributed data compression algorithm for sensor networks, we show that the compression ratio is doubled by using logical mapping while maintaining comparable mean square error. Furthermore, we also propose a mechanism to detect and classify missing or faulty nodes, showing accuracy and recall of 95% when half of the nodes in the network are missing or faulty.

1. Introduction

With the continued evolution of sensor networking hardware, the ability to deploy large numbers of sensors is becoming possible [1]. Typically, sensor networks are deployed to gather environmental information with the sensors cooperating to forward data to a data sink. One of the main challenges with such sensor networks is the need to minimize wireless transmissions to conserve energy at sensors [2].

There are several basic ways to minimize the network traffic: *in-network storage*, *data aggregation*, and *data compression*. With in-network storage, sensors store data locally and only transmit data in response to a query [3–6]. The network discards old data to store newer data. With data aggregation, different sensors aggregate their data (sum, average, etc.) and only transmit the result to the sink [7–11]. Some applications may need raw data. For example, scientists might want to collect raw temperature data in an area to model and forecast the weather, as well as for archival. For such applications, data compression is preferred [6, 12–16].

Compression can be applied to the data stream from a single sensor [16]. Good compression performance can be obtained this way as a sensor typically generates similar data over time. However, if the compressed data from a single sensor is lost, either during transmission or due to the sensor node failing thereafter, then all the data collected by that sensor are lost. Moreover, high latency is incurred because sensors must collect data for a reasonable time period for effective compression. An alternative approach, that is more resilient to transmission errors and has a low latency, is to compress data across multiple sensors in a cluster one snapshot at a time. At the same time, all data needs to be transmitted at least once to be collected. In this paper, we propose a distributed compression method based on this approach, but featuring several improvements.

We propose a *cluster-based* and *information-driven* architecture (RIDA) for a wide range of compression algorithms in real-time for scalar sensor data for a popular class of network of sensors (this work is a significant extension of [17]. We present new results comparing RIDA to a state-of-the-art

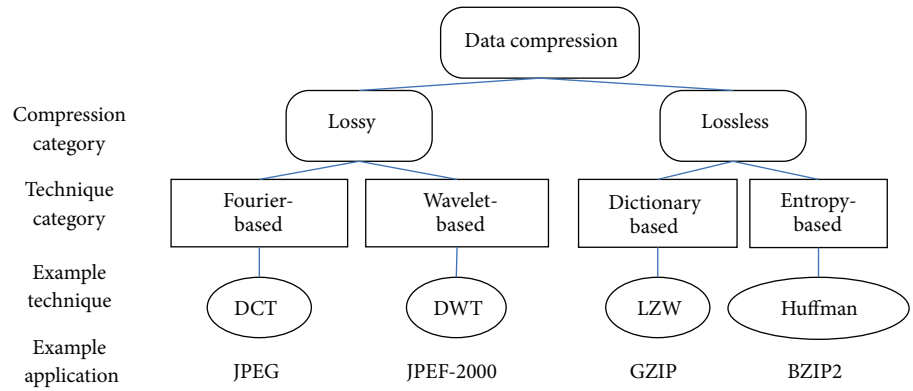

Figure 1: Main data compression methods.

distributed data compression algorithm, Wagner-DWT [18]. We also present evaluations on a new data set and study energy consumption and its asymptotic bounds). The inspiration behind RIDA is that a sensor network can be logically viewed as an image wherein each sensor is a pixel, and sensor readings are color amplitude values. With this in mind, techniques from image compression can be applied to distributed sensor networks. This is challenging because data does not originate from a single source but from multiple sensors. Moreover, sensors may not be neatly placed in a grid layout, and location information may not be available. Finally, wireless sensor networks experience high packet loss and failures, and the network topologies can become quite dynamic. RIDA addresses all these issues. RIDA makes two assumptions: the sensor network has a cluster topology, and the variation in the sensed data is relatively slow. These assumptions are typical in most existing sensor network deployments [19, 20]. We explore the nature of data correlation in sensor networks, moving beyond the notion of spatial correlation. The key idea here is that the correlation between two sensor data streams is based on the value of the data itself rather than other factors, that are irrelevant in some cases. By rearranging the data from sensors within a cluster, we can exploit more correlation in the data than by only using spatial correlation. The key contributions of this paper are the following.

(i) We propose an information-driven architecture (RIDA) with a logical mapping framework for various compression and analysis algorithms. In this approach, data reported by sensors is observed over a short period of time. After that, the data pattern can be used to assign sensors with virtual indices such that the correlation of data is utilized. Depending on the underlying compression algorithm, an appropriate logical assignment can be used.

(ii) We provide a resiliency mechanism in RIDA to address the practical problem of missing nodes in wireless sensor networks.

(iii) We design, implement, and evaluate different compression algorithms (1D and 2D, discrete cosine transform (DCT), and wavelets) on publicly available real-world sensor data [19, 20]. RIDA can achieve compression ratios of 10 : 1 and eliminate 80% of the energy consumption compared to networks not using RIDA. In particular, we evaluate one state-of-the-art distributed data compression algorithm for sensor networks [18], with and without the logical mapping framework of RIDA, and show that the compression ratio is doubled by using logical mapping while maintaining comparable mean square error.

In the next section, we will review related work. Sections 2 and 3 present a brief overview of data compression in general and related work in compression in wireless sensor networks, respectively. Section 4 highlights key observations about correlation of sensor readings that drive the design of our architecture. Sections 5 and 6 will describe the proposed information-driven architecture for compression algorithms for sensor networking, including our proposed resiliency mechanism. Section 7 will describe the experiments conducted in order to demonstrate the benefits of our approach. We discuss the limitations of our approach and future work in Section 8 and conclude in Section 9.

2. Data Compression Overview

Data compression is a process that reduces data volume while preserving the information content with some acceptable fidelity [21]. We often measure the information content using entropy [22], which represents the minimum number of bits needed to encode the data. Figure 1 shows the main data compression approaches: *lossless* and *lossy* compressions. *Lossless compression* requires decompressed data to contain exactly the same information as the original data. *Lossy compression* requires that the decompressed data contain information of the original data with only some small error.

2.1. Lossless Compression Principles. The popular lossless compression methods are *entropy-based* and *dictionary-based* compressions. In entropy-based compression, the compressor generates a statistical model of the data and maps the data to bit strings based on the generated model. An example of entropy-based compression is Huffman coding [22].

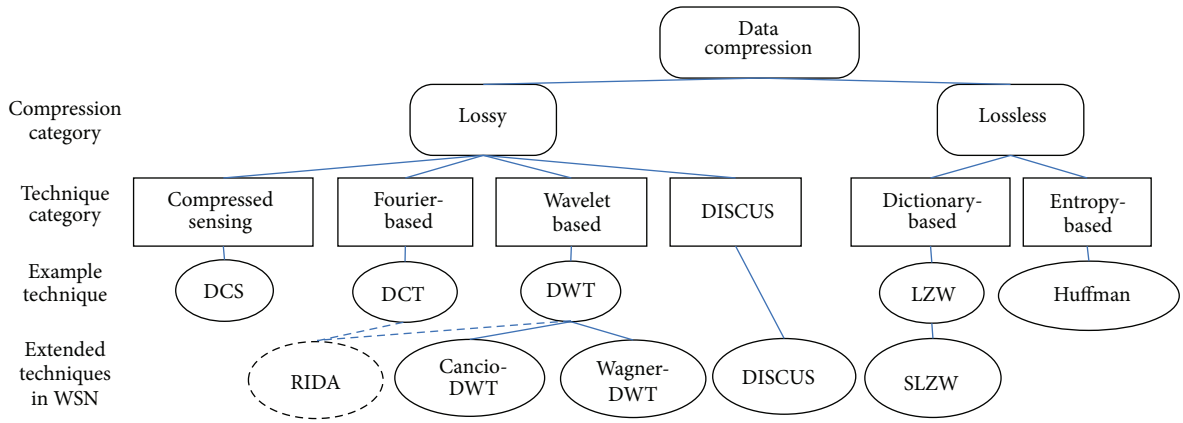

Figure 2: Data compression methods for wireless sensor networks. Highlighted methods are extended for sensor networks.

In dictionary-based compression, the compressor maintains a dictionary of encountered data and substitutes a reference to a dictionary location if the new data is already in the dictionary. An example of dictionary-based compression is the Lempel-Ziv-Welch algorithm (LZW) [22].

2.2. Lossy Compression Principles. In lossy compression, the compressor often transforms the data into a new domain using appropriate basis functions. For example, the data can be transformed into the frequency domain using Fourier basis functions. In the new domain, the information content is concentrated in a small number of coefficients that contain the data values projected using the basis functions. Therefore, the compressor can reduce the data size by selecting only those coefficients. There are two popular lossy compression methods: Fourier-based and wavelet-based compressions. Two corresponding examples of transforms are discrete cosine transform (DCT) [21] and discrete wavelet transform (DWT) [23].

The previous methods such as JPEG compression are widely used for image compression [23]. In general, we can view the network's sensor data as an image (like a visualization of temperature) or a sequence of sensor readings and apply data compression methods to compress the data. However, wireless sensor networks have distinct features such as limited computation and distributed processing, motivating new compression methods that accommodate these features. We review these methods in the following.

3. Data Compression in Wireless Sensor Networks

Several approaches [13–16, 18, 24–27] have addressed data compression in wireless sensor networks. Figure 2 shows representative work in different categories. The approaches described in [15, 18, 25] are extensions of discrete wavelet transform. We call approaches in [15, 18] Ciancio-DWT and Wagner-DWT, respectively. Other work such as Sensor LZW (SLZW) [16] is an extension of LZW for sensor networks. DISCUS [14] is a new framework for distributed compression, which is suitable for a wireless sensor network. None of these approaches explicitly address the robustness of the algorithm in the presence of missing data.

SLZW [16] extends the Lempel-Ziv-Welch (LZW) algorithm, which encodes new data based on previously encountered data. SLZW processes small blocks of data to accommodate the memory constraints of a sensor. SLZW improves the compression of sensor data by using the Burrow-Wheeler transform, which reorganizes sensor data in a way that results in better compression. SLZW is a lossless compression algorithm and exploits only temporal correlation between readings produced by an individual sensor. SLZW compresses data block by block, in blocks of 512 Bytes, with each block equivalent to 512 measurements. If the sensors take measurements once during every epoch of 30 seconds, the sink can only get the compressed data after at least 4 hours. Although it is possible to apply SLZW across sensors to exploit spatial correlation, the approach will be inefficient. SLZW builds a dictionary of repeated patterns and then uses the terms in the dictionary to reduce the coding size. In order to have SLZW work effectively, we must have long sequences of data (more than 512 values) to first build the dictionary and then utilize it. However, the number of nodes in a cluster is usually less than 100, which means that SLZW will not compress well (there is no strict restriction on the SLZW block size. However, in SLZW, the data must be observed long enough to build a dictionary of repeated patterns in the data. These repeated patterns are encoded using a shorter code word to reduce the size of the compressed data. If the block size is too short, the delay will be short, but few samples will be observed, and the compression performance will degrade severely. In contrast, distributed compression frameworks which correlate data from multiple sensors can provide good compression ratio even when the delay is very small). We can collect several snapshots to have enough measurements for compression. Nevertheless, this will cause delay in data delivery. Our approach is orthogonal to SLZW in that we explore correlation between readings of different sensors and ensure low-latency data collection. RIDA instead compresses the data after every epoch. Hence, the sink can get the compressed data within 30 seconds. We also address the important problem of detecting missing data in sensor networks.

Approaches in [15, 24, 25] transform sensor data on a routing path in the routing tree. Ciancio-DWT [15, 25] use distributed wavelet transform while [24] uses Karhunen-Loeve Transform. A sensor in the routing path partially calculates the transformed coefficients and forwards the coefficients to the next sensor in the routing path. The sensor at the end of the routing path will receive all the coefficients of the DWT of sensors along the routing path. Ciancio's algorithm performs a wavelet transform on the raw data along a path. But the paper does not explore methods to improve the compression radio obtained with wavelet transform on the same set of data. The algorithm does not explicitly exploit temporal and spatial correlations between sensor data since it depends on the routing tree. Wagner-DWT [18] is a distributed framework for data compression and analysis using wavelet lifting. Wagner-DWT also considers spatial information such as locations to decide which scale a sensor should be. Wagner-DWT exploits spatial correlation in the data based on sensor locations.

DISCUS [14] is a theoretical approach for compression. Sensors do not need to know their correlation at compression time. However, the sink needs the correlation information to decompress the data. DISCUS uses the known correlation information to classify sensor data into different sets (called *coset*). DISCUS indexes sensor data within a coset with fewer bits than it takes to index the sensor data without cosets. The values of sensor data in each coset have unique properties with the correlation information. Therefore, DISCUS compresses sensor data by sending only the index within a coset. At the decompressor, DISCUS uses the correlation information to search for the best sensor data value in the coset to recover the original sensor data. For example, we need 8 bits to index a temperature level of range from 0 to 225. If we classify the temperature levels into two cosets; one coset contains the temperature levels of range from 0 to 127, and the other contains the temperature levels of range 128 to 255. Within each coset, we need only 7 bits to index a temperature level. For example, temperature level 129 can have the index value as 1 in the second coset. If the decompressor receives 1 and it knows that the temperature level is larger than 127, it searches for the index value 1 in the second coset and gets 129. Hence, we can reduce the number of bits to index sensor data if we know some information about the sensor data in advance. In practice, it is difficult to determine the correlation information because there is no historical data available for many applications.

Compressed sensing [28, 29] allows the original data to be reconstructed exactly with high probability from only a subset of random measurements. Distributed compressed sensing [30] is an extension of compressed sensing for distributed sources such as sensor networks, which allows collection of distributed data. DCS and its related applications [31–35] are proven to be very robust to packet losses and node failures and able to capture high frequency signals. In order for DCS to work, prior knowledge about the sparsity and compressibility of the sensor data is needed. DCS does not require information exchange between sensors in a cluster, thus scaling well with the cluster size. However, it does not guarantee optimal energy savings because the total number of measurements collected is empirically at least four times the compressed data.

3.1. Lossy versus Lossless Compression in a Wireless Sensor Network. We argue that lossy compression is useful in a wireless sensor network although it might introduce some errors. Wireless sensor networks already need mechanisms to handle sensor data errors (e.g., Kalman filtering) due to the imperfect behavior of electronic circuits and analog to digital converters. Applications can compensate for error at the back-end servers, which often have large memory and high computation power. Computation at the back-end server is therefore much cheaper than at the sensors. Lossy compression often achieves higher compression ratios and thus greater energy savings, compared to lossless compression. Finally, lossy compression techniques such as JPEG and JPEG-2000 [21] do well on natural data such as natural images or temperature data of an environment. JPEG and JPEG-2000, however, can only be applied to temperature data that is sampled in a regular 2D lattice.

In comparison to previous work, our proposed architecture, RIDA, is a lossy data compression architecture that relies on only the sensor data. RIDA assumes that a sensor network is organized into physical groups (clusters) using an existing clustering protocol such as LEACH [36]. RIDA does not assume that the nodes in a sensor network are located in a regular grid and does not use any further information such as sensor location within a cluster or routing path for compression. RIDA uses a logical mapping to improve existing compression methods. In the next section, we present the key observations based upon which we developed RIDA.

4. Understanding the Correlation between Sensor Data

The key challenge in data compression is to exploit correlation between data in time, space, or frequency domains. In practice, it is difficult to predict the correlation between two data series collected over time. Therefore, we often use heuristics to approximate the correlation between sensor data. Existing approaches (such as the work in [13]) assume that sensors that are close together report similar data. However, we observe that *sensors that are not physically close together may report more correlated data*. It is not impossible to learn certain information such as correlation in the sensor data [37, 38]. However, the learning algorithms might require a lot of historical information or incur overheads. We chose to make RIDA simple by observing only a few values and proceed with compression.

Figure 3 shows light readings for one sensor cluster in the Intel Research Lab at Berkeley. We observe that sensors that are not close together but in similar environmental conditions report similar data. For example, light sensors that are under light bulbs can report similar high light levels.

We also observe that, sensors may report similar data regardless of external environmental conditions. For example, sensors report the voltage of their batteries. The voltage of the batteries is independent of external factors such as

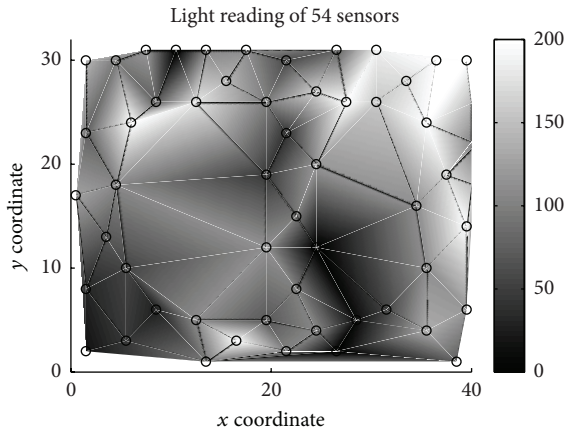

Figure 3: Mapping of light sensor data in Intel Research Lab at Berkeley. Brighter color corresponds to higher light intensity. Sensors under light bulbs can report similar high light levels.

sensor location and environment temperature. Sensors with similar voltage readings should report similar data over time provided that they consume similar amounts of energy.

From these observations, we believe that we should consider the sensor data itself to find the correlation between the sensor data streams in a cluster, rather than other heuristics such as the sensor location in a cluster. When we find the correlation between sensor data, we can assign sensors with appropriate indices, which improve the compression performance of an algorithm. For example, DCT-based and DWT-based compression algorithms perform very well on smooth or piecewise smooth signals, which frequently arise when we sort the sensor data by their values. We describe next the information-driven data compression architecture for irregular wireless sensor networks.

5. Robust Information-Driven Data Compression Architecture

RIDA assumes that nodes have been organized into *clusters*, using any existing network clustering protocol such as LEACH [36]. Within each cluster, there is a *cluster head* that is responsible for maintaining the cluster information.

The key idea in RIDA is to reorganize sensor data before doing compression in order to improve the compression performance. Existing compression algorithms such as DCT and DWT compress well when adjacent data have similar values. Hence, we use a logical mapping to group sensor data of similar values together and compress the data.

The architecture has three main components (Figure 4): *logical mapping*, *compression algorithms*, and *a resiliency mechanism*. In logical mapping, we observe the sensor data for a period of time and use a logical mapping scheme to assign each sensor a virtual index in a way that results in a good compression ratio when using a specific compression algorithm. The compression algorithms component contains data transform methods that we extend for sensor networks.

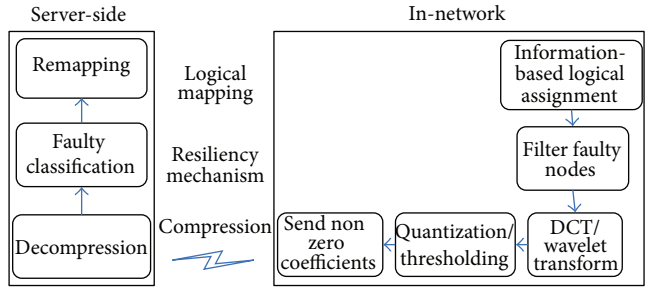

Figure 4: RIDA detailed architecture.

We have adapted the discrete cosine transform and the discrete wavelet transform for sensor networks. As a resiliency mechanism, we propose a simple classification method to enhance the robustness of the network to missing data.

There are two main phases at a cluster: *mapping* and *compression*. In the mapping phase, the cluster head assigns virtual indices to sensors and sends the logical map to the sink for remapping later. In the compression phase, sensors exchange data within their cluster. Each sensor transforms the received data using a technique such as DCT or DWT to calculate the corresponding coefficient. The transformed data are coefficients, which contain different amounts of information about the sensor data. Each sensor weighs its coefficient using a quantization matrix. The coefficients after quantization are rounded to the nearest integer values. Each sensor uses a threshold (usually zero) to decide which coefficients to keep and send to the sink. At the sink, after collecting all the compressed data, the sink decompresses the data and uses the logical map to remap the sensor data to the true sensor indices.

5.1. Logical Mapping. Logical mapping rearranges sensors logically by assigning each sensor a virtual index. The purpose of the rearrangement is to group sensors with correlated data together to achieve higher compression performance. Sensors use virtual indices, which are independent from sensor locations within a cluster. Hence, we can compress sensor data well regardless of the network topology. By carefully assigning indices to the sensors for a particular compression scheme, we can compress the data better (or at least not worse than) compression without logical mapping. We formalize the mapping as follows:

$$M_t\left(d_s\left(t\right), D\left(t\right)\right) = l, \tag{1}$$

where s denotes the sensor ID. $d_s(t)$ denotes the sensor data value of sensor s at time t. $D(t)$ is a vector of size n of data values of sensors in the cluster at time t. M_t denotes the mapping from a sensor data value $d_s(t)$ to a virtual index l such as (x, y) in two-dimensional mapping. M_t uses only the value of the sensor data $d_s(t)$ and values of other sensors in the cluster $D(t)$ to determine l. Figure 5 shows an example of logical mapping of light readings. The mapping can be application and algorithm specific. As a first step, we simply sort the data and index the sensors in a sequence based on the order of the sorted data. For example, Figure 6 shows

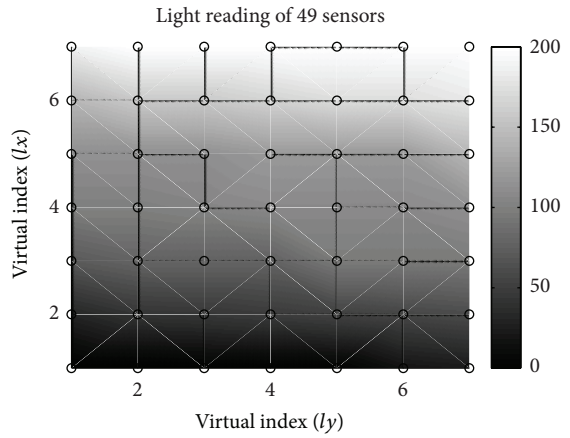

Figure 5: Logical mapping of light readings. The brighter the color is, the higher the light intensity is. Sensor data values are sorted, and the sensors are assigns virtual indices sequentially based on the sorted order. The result is a smoother map of light readings compared to Figure 3.

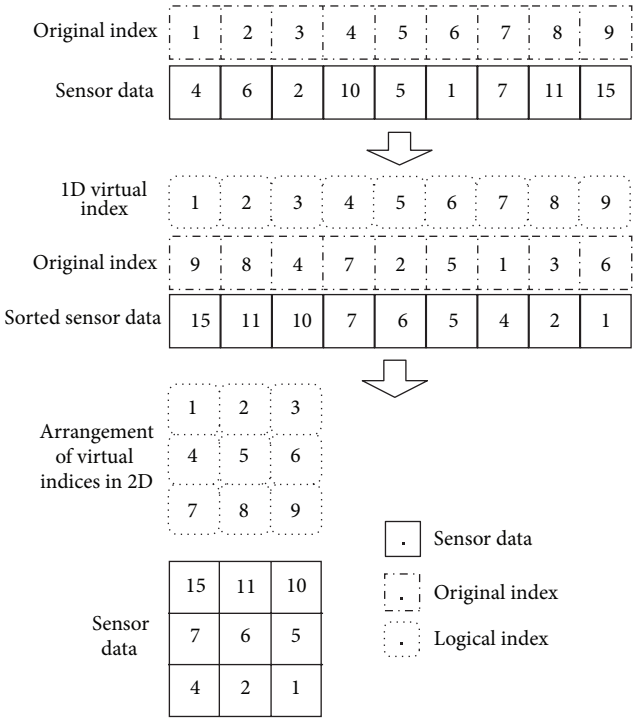

Figure 6: Logical mapping illustration.

data values from 9 sensors. We can assign each sensor a virtual index sequentially after sorting the data values. In 1D mapping, sensor 9 has virtual index 1, sensor 8 has virtual index 2, and so on. In 2D mapping, we can stack the sensors row by row or column by column and give them virtual indices. In Figure 6, the 1D virtual index of 3 is mapped to a 2D virtual index of (1, 3) corresponding to its position in a 2D matrix, and the 1D virtual index of 5 is mapped to the 2D virtual index of (2, 2).

Because the correlation relationship between sensors may change over time, the cluster may need to perform remapping. The frequency of remapping depends on the application requirement and the characteristic of the environment. When a cluster needs to remap, the cluster head broadcasts a *begin-mapping* message. A sensor, upon receiving the *begin-mapping* message, sends its data to the cluster head. The cluster head collects data from each sensor for a short period of time. The cluster head computes the average data value for each sensor and sorts the averaged values. The cluster head assigns each sensor an index and broadcasts the new map to the cluster and to the sink. The cluster head waits for all acknowledgments and may rebroadcast the map if necessary. After receiving all acknowledgments, the cluster head broadcasts an *end-mapping* message. The cluster resumes to normal sensing mode.

Figures 7(b) and 7(c) show the distribution of the coefficients and the error versus compression ratio using DWT with and without logical mapping from a real data set collected from the LUCE Deployment in SensorScope [20] (Figure 7(a)). Most of the coefficients in DWT-logical mapping are near zero. Hence, DWT Logical mapping has a lower error compared to DWT without logical mapping.

5.2. Compression Algorithms

5.2.1. Data Transformation. We have adapted DCT and DWT for sensor networks. Within a cluster, sensors use the wireless broadcast capability to exchange data with each other. After receiving all data, each sensor calculates only its coefficient, which corresponds to its virtual index. Pseudocode 1 shows the pseudocode for the distributed DCT. From lines 8 to 12, a sensor calculates the coefficient that corresponds to its virtual index. If all the sensor coefficients were calculated centrally, the computational complexity would be $O(N \log(N))$ [21], where N is the number of sensors in the cluster. In our distributed DCT implementation, each sensor calculates only its own coefficient. The complexity for each sensor is $O(N)$. The total complexity for the whole cluster is $O(N^2)$. Fortunately, the computation cost is very small compared to sensing and communication costs. The coefficient of the corresponding virtual index k is calculated using the following formula for one-dimensional transform:

$$\text{coef}f\,(k) = w\,(k) \sum_{n=1}^{N} \text{data}\,\,(n) \cos \frac{\pi\,(2n-1)\,(k-1)}{2N}, \quad k = 1, \ldots, N, \tag{2}$$

where

$$w\,(k) = \begin{cases} \dfrac{1}{\sqrt{N}} & \text{if } k = 1, \\ \sqrt{\dfrac{2}{N}} & \text{if } 2 \le k \le N. \end{cases} \tag{3}$$

In the matrix form, it can be viewed as the dot product of the corresponding row in the DCT matrix B and the vector

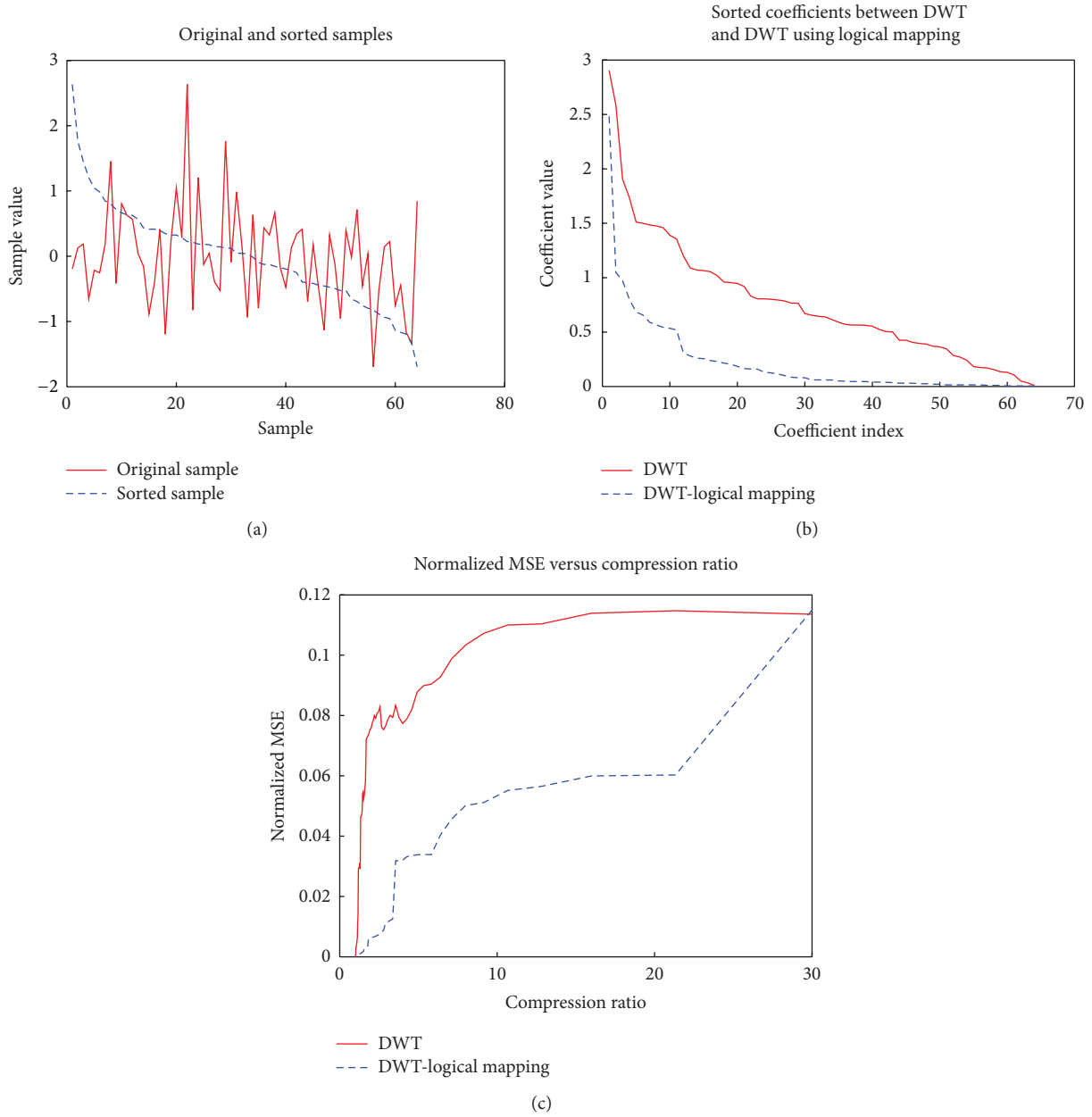

Figure 7: (a) Original and sorted samples. Sorted samples result in a much smoother signal compared to the original samples. (b) Sorted coefficients between DWT and DWT using logical mapping. Nearly half of the coefficients in the DWT-logical mapping are near zero. (c) Normalized MSE versus compression ratio. DWT-logical mapping has a much lower error compared to DWT without logical mapping.

of sensor data. Hence, the computational complexity is $O(N)$ per sensor.

5.2.2. Quantization. Quantization weighs the coefficients using a quantization matrix. Important coefficients should be nonzero, while unimportant coefficients can be quantized to be zero. Sensors transmit only nonzero coefficients to the sink. In this work, when we map sensor data in one dimension, we use a threshold to drop coefficients. The sorted data is often much smoother than the original data. This property is similar to the property of most natural images, which observations have shown are smooth. Therefore, when we map sensor data in two dimensions, we use the JPEG quantization matrix, which is primarily intended for natural images. The choice of a quantization matrix may directly affect the compression performance. We discuss quantization in detail in Section 8.3.

5.3. Energy Balancing. An algorithm should ensure approximately equal energy consumption among nodes to increase the total network lifetime. The data after transformation and quantization often results in a few nonzero coefficients,

```
//Broadcast local reading to
//the cluster
1   broadcast (reading)

//collect neighbors' reading in
//specified interval
2   while not timeout
3     do
4       data [packet·virtual index]
          = packet·reading

//Compression with DCT transform
//Calculate only row k of B ⋆ data
//k is the virtual index of this node
5   for i← 1 to cluster_dim
6     coeff← coeff + B[k] [i] ⋆ data [i]

//Quantization
7   coeff = round (coeff/Q[k])

//Only transmit a nonzero coefficient
8   if coeff ! = 0
9     send (coeff)
```

Pseudocode 1: Pseudocode for 1-dimensional DCT transform. From lines 5 to 6, a sensor calculates only the coefficient that corresponds to the sensor virtual index k.

concentrated in some specific indices. For example, the 2D DCT coefficients often concentrate at the top left of the transformed data. Nodes with those corresponding indices may have to transmit the coefficients more frequently and deplete energy more quickly than others. To avoid this situation, the logical mapping can alternate different mapping schemes, so that sensor nodes have an approximately equal chance to transmit coefficients. A simple method is to alternate the sorting between ascending and descending orders for each remapping. Alternatively, nodes can maintain a random map where indices are assigned randomly to assign which coefficient a node should calculate. The random map can be regenerated in each remapping phase.

5.4. Error Detection and Classification. Reliability of data is of paramount importance because network nodes fail frequently. Even when nodes have not failed, their operation is typically unstable. Figure 8 shows the reading history of 54 sensors in a controlled environment [19]. As observed, 53 out of 54 nodes are working. However, the number of nodes reporting data is always around 50% within each epoch. Better design of routing protocols could help increase this rate, but we still have to address the problem of faulty and missing nodes. When these nodes do not report data within an epoch, we call the expected data from those nodes as *missing data*. In such a case, a node will not have enough data to perform compression using the logical map. To do compression, the node must fill the missing data with some values. This situation motivates us to design a simple

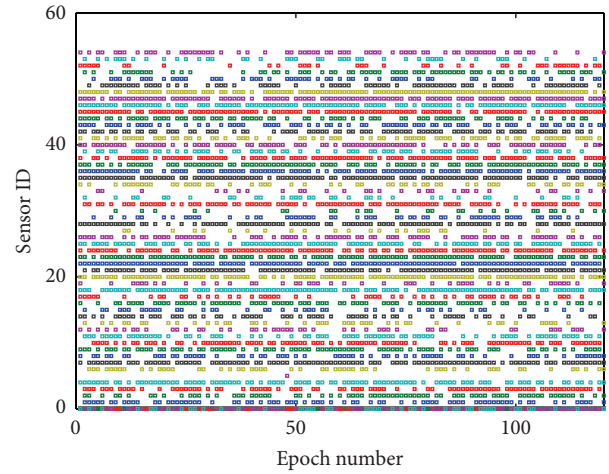

Figure 8: Reading history over a one hour period from a real sensor network at Intel Research Lab at Berkeley [19]. A dot indicates that a reading has been received.

mechanism to distinguish between missing data and real data at the sink after decompression.

All the nonzero data will be projected to an interval (e.g., [128,255]). Different types of data have different ranges. Although the data value is obtained from the same 10-bit analog-to-digital converter (ADC), the data ranges are different. Therefore the projection will unify the way we drop coefficients through quantization or thresholding. We choose the interval [128,255] for projection, so that we can use the same quantization scheme as JPEG. The pixels in JPEG have values from 0 to 255. Missing data will be set to zero. We have a set of data from the interval [128,255] for normal data and 0 for missing data of all different scalar sensor types like temperature, humidity, light, and voltage. These zero values would result in low values in the reconstructed data. The use of only half of the range [0,255] is to allow enough gap between missing data and real data after decompression. Hence, we can use a threshold to classify them. The threshold we used is 64 which has been shown to classify correctly most of the time. We can extend the projection range [128,255] and use a smaller threshold to classify missing data. However, there is an inherent trade-off in the ability to detect missing readings and the decompression error. In addition, the choice of the projection range also depends on the level of noise in the sensor readings.

6. RIDA Operation Details

Having described the overall RIDA architecture, we now describe its operation in detail. There are three different classes of nodes with slightly different operations: the sink, the cluster head, and the child node.

The Cluster Head. Upon receiving a query for sensor data from the sink, the cluster head performs the following actions. These actions can be repeated after a *remapping period*, which is sent from the sink or the default value.

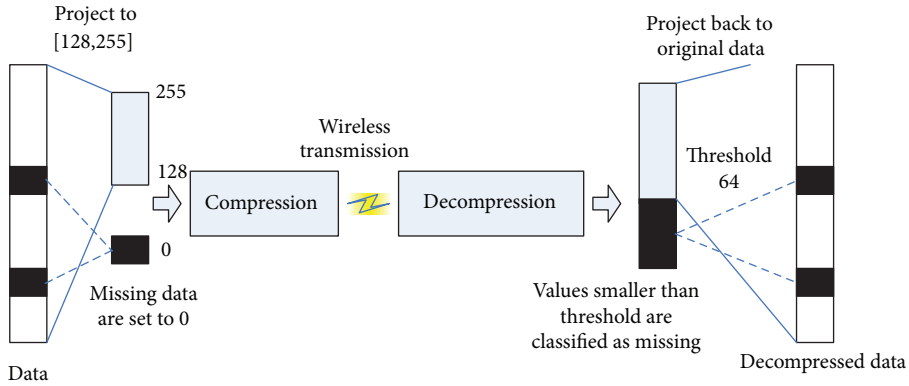

Figure 9: Missing data classification mechanism: missing data are set to zero. They can later be classified by thresholding the reconstructed data.

(i) Send a *begin mapping*, which queries for data for s epochs from all nodes in its cluster and performs the following actions.

(ii) Calculate the mean data values of each sensor and sort the means, alternating between a descending order and an ascending order each round, to ensure load balancing among nodes.

(iii) Assign virtual indices to the sensors and create a logical map. The map is a set of tuples < id, vindex > where id is the node ID and vindex is the virtual index of the corresponding node.

(iv) Broadcast the logical map to all nodes in its cluster.

(v) Send the logical map to the sink.

(vi) Wait for all acknowledgments.

(vii) Send an *end-mapping* message to all nodes in its cluster.

The Child Node. Upon receiving a *begin-mapping* message from the cluster head, a child node returns its measurements. Upon receiving the logical map, the child node sends an acknowledgment. It must calculate the coefficient corresponding to its virtual index on the logical map. Upon receiving an *end-mapping* message, it must perform the following actions for each epoch.

(i) Take a measurement, and broadcast it to the cluster.

(ii) Listen to all readings from other sensors, and calculate its coefficient using the procedures described in Pseudocode 1 and Figure 9.

(iii) If the coefficient is nonzero, send the coefficient to the sink.

(iv) If there are no more tasks, go back to sleep or idle mode.

The Sink. Upon receiving a map, the sink saves it for decompression. For each epoch, the sink collects all the coefficients from the network nodes and performs the following actions.

(i) Decompress data for each cluster.

(ii) Use the received logical map to match data values and sensor nodes.

(iii) Classify missing data using the procedures described in Figure 9.

(iv) Combine data from all clusters to obtain network-wide measurements.

7. Experimental Design and Analysis

This section describes the experiments conducted to evaluate the architecture and analyzes the results obtained.

7.1. Goals and Metrics. We investigate five main questions.

(1) Can existing compression methods be adapted for RIDA?

(2) Does logical mapping improve compression performance, particularly for state-of-the-art distributed compression algorithms?

(3) How sensitive is RIDA performance to network parameters, such as network density and number of hops between the cluster and the sink?

(4) How sensitive is RIDA performance to the mapping period (the time duration between two remappings)?

(5) How robust is RIDA when there is missing sensor data?

To answer the first question, we show that we can adapt three main compression methods, DCT, DWT, and Wagner-DWT, for RIDA. In addition, RIDA needs to change only the logical mapping to suit the compression algorithms.

To answer the second question, we evaluate RIDA over a real-world data set. We compare DCT and DWT compressions with RIDA's logical mapping to DCT and DWT without logical mapping. In particular, we evaluate one state-of-the-art distributed data compression algorithm for sensor networks [18], with and without the logical mapping framework of RIDA. We study metrics such as the normalized mean square error (NMSE) with respect to the original data and the compression ratio.

To answer the third and the fourth questions, we study the reduction in energy consumption as a function of network density, number of hops between the cluster and the sink, and mapping period. We study the compression performance as a function of the cluster size from real sensor data. We measure sensor energy consumption using PowerTOSSIM [39], a power simulator for wireless sensor network. We implemented the compression algorithms for the MicaZ platform and simulated the system in PowerTOSSIM. We measure energy consumed by CPU operations and RF transmission. We analyze how sensitive the results are to the number of hops between the cluster and the sink and the mapping period over a time period.

Finally, to answer the last question, we emulate missing data in the data set by randomly setting some sensor readings to zero. We analyze the classification accuracy, classification recall, and the compression ratio in relation to the normalized mean square error.

The following notations are used in the formulas.

(i) e_b is the energy used to transmit all raw data back to the sink.

(ii) e_c is the energy used to transmit data back to the sink using RIDA.

(iii) n is the average number of sensors in a cluster (cluster size). We assume that within a cluster, sensors can communicate directly with each other. We further assume that the energy required for communication is the same across all sensor pairs in a cluster.

(iv) h is the average number of hops from a cluster to a sink.

(v) e_r, e_x are the average transmission and reception energy for one message.

(vi) e_s is the energy used for sensing. We ignore the energy required for computing the logical map because it is much smaller than the energy used for transmission and sensing when the mapping period is large compared to the sampling period.

(vii) n' is the number of nonzero coefficients.

(viii) T is the total time period in minutes under consideration.

(ix) m is the number of times sensors collect data during the mapping phase.

(x) k is the mapping period, which is the time between two mappings in minutes. Hence, T/k is the number of remappings in time T. $(T/k)n(e_s + e_x + e_r)m$ is the total energy consumed by remapping in time T.

(xi) s is the sampling period, which is the time between two samples in minutes. Hence, T/s is the number of times sensors collect data in time T.

(xii) n_i' is the average number of nonzero coefficients in the ith mapping.

(xiii) $(k/s)\sum_{i=1}^{T/k}(e_s + e_x + e_r)n + h(e_x + e_r)n_i'$ is the total energy consumed by collecting, compressing, and transmitting data from sensors to the sink in time T.

The total energy consumption when the network does not compress the data is

$$e_b = \frac{T}{s} n \left(e_s + \left(e_x + e_r\right) h\right). \tag{4}$$

The total energy consumption using compression is

$$\begin{aligned} e_c = {} & \frac{T}{k} n \left(e_s + e_x + e_r\right) m \\ & + \frac{k}{s} \sum_{i=1}^{T/k} \left(e_s + e_x + e_r\right) n + h \left(e_x + e_r\right) n_i' \\ & + \frac{T}{k} h \left(e_x + e_r\right), \end{aligned} \tag{5}$$

where the individual components are

(i) energy consumed in remapping as follows:

$$\frac{T}{k} n \left(e_s + e_x + e_r\right) m, \tag{6}$$

(ii) energy consumed in transmitting the map back to the sink as follows:

$$\frac{T}{k} h \left(e_x + e_r\right), \tag{7}$$

(iii) energy consumed in sensing, exchanging data within the cluster, and transmitting the compressed data to the sink as follows:

$$\frac{k}{s} \sum_{i=1}^{T/k} \left(e_s + e_x + e_r\right) n + h \left(e_x + e_r\right) n_i'. \tag{8}$$

The normalized reduction in energy consumption is

$$r_h = \frac{e_b - e_c}{e_b}, \tag{9}$$

where r_h basically tells us how much energy a sensor can save relatively to the amount of energy consumed without using RIDA.

7.2. Methodology. We evaluate RIDA on two real-world data sets from (i) an indoor deployment at the Intel Research Lab at Berkeley [19] and (ii) an outdoor deployment from SensorScope project [20] at Ecole Polytechnique Federale de Lausanne (EPFL). (i) uses TinyDB, discussed previously, to collect raw data such as temperature, humidity, and voltage from 02/01/2004 to 04/01/2004. TinyDB collects data every epoch (a fixed time interval). We chose these data sets because they are publicly available and used by other researchers working on compression. First, we would like to evaluate how RIDA performs in the ideal case where all sensors report data. However, the data set has many missing values. Therefore, we interpolate the missing data. We interpolate the missing data in time for two reasons. First, we assume the physical phenomena changes slowly in time. Second, time

interpolation should have minimal effect when evaluating compression algorithms that exploit spatial correlation. We later emulate missing data. We emulate 50% missing data, which is the worst case we observe in this testbed. The latter sensor network is deployed in an outdoor environment. The network collects meteorological as well as monitoring data every 30 seconds from around 11/01/2006 to 05/09/2007.

In addition, we implement RIDA with DCT transform on the Mica2 embedded devices [40] and simulate the network using PowerTOSSIM [39] in Tinyos-1.x [41]. We measure the energy consumed by radio for receiving and transmitting packets (e_x+e_r) and the energy consumed by CPU for sensing and performing calculation (e_s). It turns out that ($e_x + e_r$) is about 2.5 e_s. We then use this ratio in our calculations for Figure 11.

7.3. Results. We now analyze the results obtained from our experiments.

7.3.1. Compression Performance: Mapping versus No Mapping. We assign indices to sensors based on the order of sensor data values. In the mapping phase, we collect and sort the sensor data. Then we assign indices to sensors sequentially based on the sorted data values. We evaluate a one-dimensional (1D) mapping, in which we arrange sensors logically into a vector, and a two-dimensional (2D) mapping, in which we arrange sensors into a matrix by stacking the sorted data values column-wise. We compare how RIDA performs when using logical mapping relative to *no logical mapping* and 1D mapping relative to 2D mapping. In *no logical mapping* case, the indices are assigned based on the node ID. In the physical map of the testbed, two nodes with close IDs have their locations close together. Hence, the mapping in some senses takes account into spatial correlation based on sensor locations.

We also compare a state-of-the-art compression method using distributed wavelet transform (Wagner-DWT) [18] with and without logical mapping. In the former case, we apply logical mapping to assign sensor nodes virtual indices before the compression and use the same transformation algorithm in Wagner-DWT. The logical indices are assigned based on the data values (not the sensor locations). We use the same 2D logical mapping as described in Figure 6. We sort the sensor data values and assign each sensor a virtual index sequentially based on the sorted data value. In the latter case, Wagner-DWT has an algorithm that assigns indices to sensors based on their locations. We chose Wagner-DWT for comparative evaluation since it is a transform-based approach, and the main contribution of RIDA is the logical mapping that is added on existing compression systems that use transformation. The logical mapping makes the transformation result in a sparser set of coefficients. In contrast, the DISCUS theoretical framework [14] does not fall in this category—transform compression. Hence, RIDA does not apply for DISCUS, and DISCUS was not chosen for comparative evaluation.

In the indoor case, Figure 10(a) shows the compression ratio versus compression error of three compression techniques 1D DCT, 2D DCT, and Wagner-DWT with and without logical mapping (note that we do not include 2D DWT LM as it is the same as Wagner-DWT LM in terms of compression ratio). In general, with the same compression error, using a logical mapping improves the compression performance by approximately 200% relative to not using a logical mapping. In addition, the 1D mapping compresses twice as well as the 2D mapping. The reason that 1D DCT is better than 2D DCT can be explained by the fact that given the same number of data values, the 1D signal is actually longer than that of the 2D signal. For example, an 8×8 2D signal can be considered as 8 short signals, each of length 8. If we arrange all these signals into 1D, we have a signal of length 64. Longer signals can contain more information of high frequencies compared to short signals. Therefore, the coefficients of high frequency components can be retained during quantization. This avoids information loss during compression. Wagner-DWT with logical mapping outperforms the others. However, it gets worse than 1D DCT with logical mapping when the compression ratio is greater than 15.

In the outdoor case, Figure 10(b) shows similar results from the SensorScope data set [20] to the indoor Berkeley data set. In general, with the same compression error, using a logical mapping improves the compression performance of existing techniques. Wagner-DWT with logical mapping is slightly better than Wagner-DWT without logical mapping and significantly better than other techniques.

RIDA achieves similar results for humidity and voltage data. However, RIDA performs poorly on light sensor data. The compression ratio for light sensor data is only 1.5 with a compression error of 30%. In our opinion, the reason for the poor performance is that the light sensor data has a large range from 2 to 900 lux. Projecting a wide light range [2,900] to [0, 255] also incurs significant information loss. This loss can be minimized by using a wider range (e.g., [0,512] to project the data on. However, the quantization and error classification also need adapt to that range.

7.3.2. Compression Ratio versus Network Density. RIDA assumes that the sensor network is dense—there are many sensors in a cluster—and sensors in a cluster communicate directly and with equal power consumption. In this experiment, we analyze how RIDA performs as we vary the number of sensors. We use the DCT compression algorithm with 1D logical mapping. We vary the number of sensors in a cluster from 16 to 49 and observe the compression ratio for compression errors ranging from 1% to 5%.

Figure 10(c) shows that for the same compression error, the compression ratio increases with the number of sensors. The compression ratio of a cluster with 49 sensors is twice the compression ratio of a cluster with 16 sensors. Figure 10(c) shows a reasonable result because data transformation often performs better for long sequences of data than short ones. Thus, denser sensor networks will typically achieve better compression of sensor data.

7.3.3. Reduction in Energy Consumption versus Number of Hops. We evaluate energy consumption in RIDA by varying

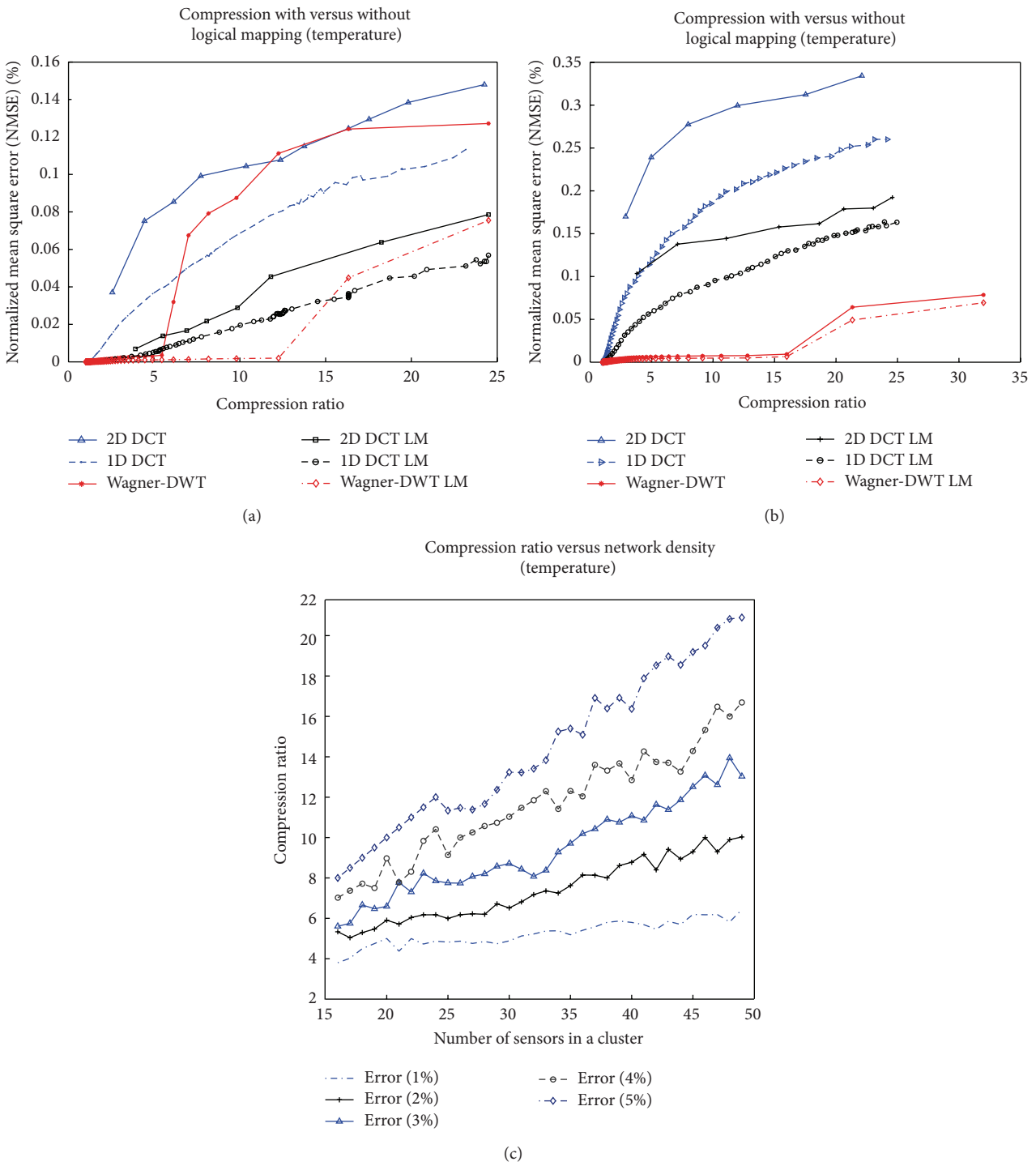

FIGURE 10: (a) Compression performance of temperature (indoor—Berkeley data set): mapping versus no mapping. Logical mapping improves the compression performance by approximately 200% relative to no mapping. Compression ratio using 1D mapping is twice as large as that using 2D mapping. (b) Compression performance of temperature (outdoor—SensorScope data set): mapping versus no mapping. Logical mapping improves the compression. Compression ratio using 1D mapping is twice as large as that using 2D mapping. Wagner-DWT with logical mapping outperforms all other techniques. (c) Compression ratio versus network density: denser sensor networks typically achieve better compression of sensor data.

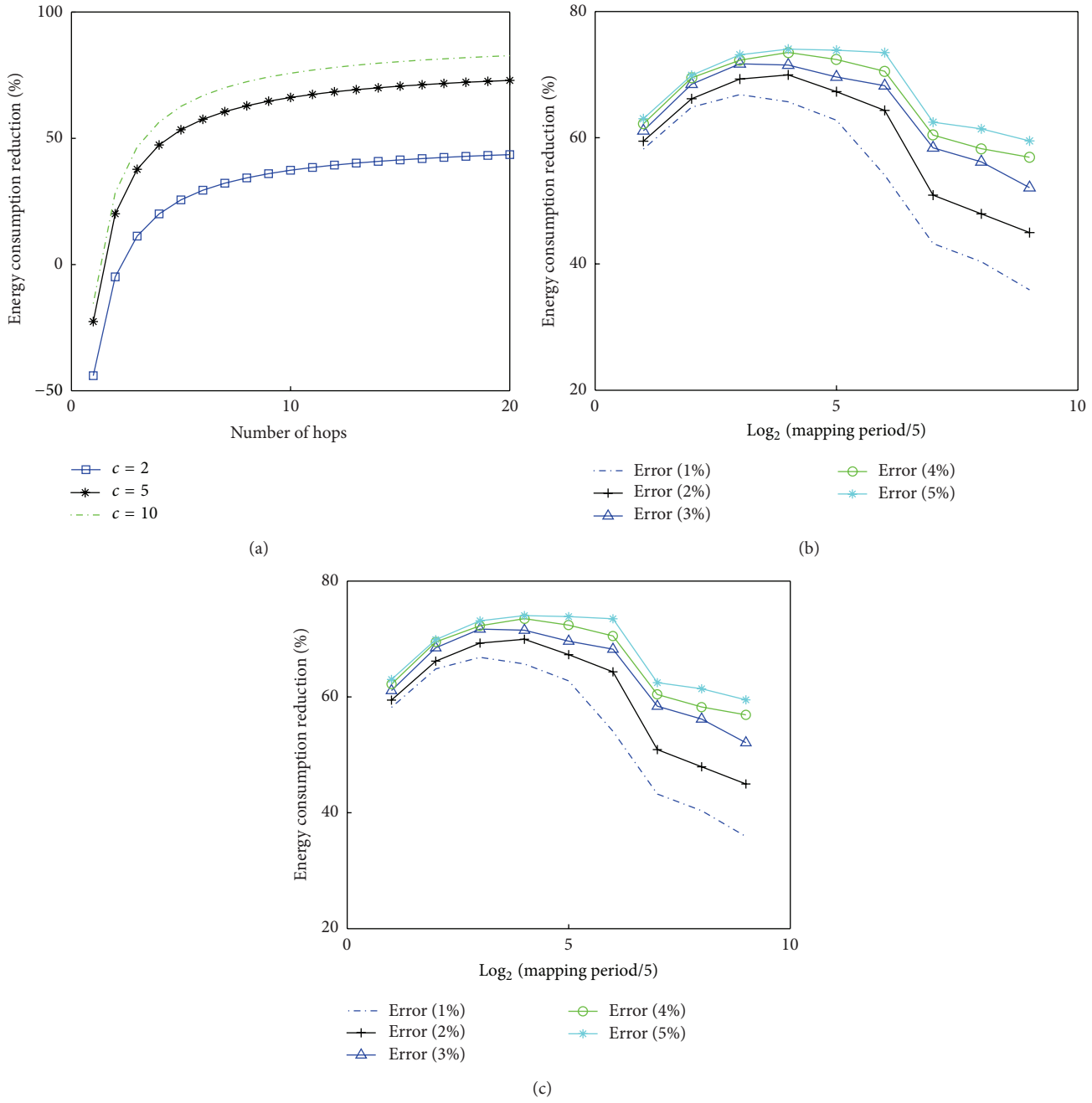

FIGURE 11: (a) Energy consumption reduction versus average number of hops. c is the compression ratio. The higher the number of hops from a cluster to a sink, the greater the reduction in energy consumption. If the average number of hops is small, the reduction in energy consumption is negative, showing that transmitting raw data directly to the sink is more energy efficient than compression. (b) Compression performance of different mapping periods. Each line indicates the compression ratio versus the normalized MSE. For a fixed error level, the shorter the mapping period is, the higher the compression ratio is. (c) Energy consumption reduction for different mapping period.

the average number of hops from the sources to the sink. In this experiment, we keep the cluster size fixed and simulate the average number of hops from the sources to the sink. The reason we can only simulate this parameter is because there are only 54 sensors in the Berkeley data set we used. We increase the average number of hops from the sources to the sink by increasing the number of clusters in the network. We vary the average number of hops from a cluster to a sink from 1 to 20 and calculate the energy consumption reduction with compression ratios equal to 2, 5, and 10. As we expected, Figure 11(a) shows that the higher the number of hops from a cluster to a sink, the greater the reduction in energy consumption. Even when the compression ratio is 2, we can still reduce energy consumption by 20% when the average number of hops from a cluster to a sink is 6. However, if the average number of hops is small, the reduction in energy

consumption is negative, meaning that compression is less energy efficient than transmitting raw data directly to the sink. We observed similar behavior when studying the impact of compression ratio (for a fixed hop count) on the reduction in energy consumption. The energy reduction increases logarithmically with the compression ratio. However, if the compression ratio is too low, the energy reduction is actually negative.

7.3.4. Impact of Mapping Period. Figure 11(b) shows how the compression ratio decreases with the mapping period. The intuition is that the shorter the mapping period (the more frequently RIDA remaps the sensors), the more up to date the correlation information between sensor data. The logical mapping assigns virtual indices for sensors more precisely. Therefore, RIDA compresses data better. However, the more frequently the RIDA remaps the sensors, the more energy the RIDA consumes. Hence, there is a trade-off between energy consumed by remapping and the compression ratio.

Figure 11(c) shows the reduction in energy consumption of different mapping periods in minutes for compression errors ranging from 1% to 5%. We plot the mapping period on a logarithmic scale for better visualization. The figure shows that RIDA does not reduce energy consumption significantly if the mapping period is too short or too long. The good mapping periods are from 40 minutes to 2 hours. However, this result can vary depending on the environment being observed.

7.3.5. Missing Data Classification. In this section, we show how RIDA resiliency mechanisms detect and classify missing data after decompression. Missing data is randomly injected into the data set before compression. The nonzero data is scaled to [128,255] interval, and we use a threshold of 64 to classify faulty data in both cases.

Figures 12(a) and 12(b) show that when the number of faulty nodes increases from 1 to 30, the normalized mean square error in both DCT-based and DWT-based transforms is less than 2%. The normalized mean square error is calculated based on only the real sensor data because it is meaningless to calculate the error of the missing sensor data. The compression ratio also decreases gradually from 10 : 1 to 3 : 1. This is reasonable because nature of DCT-based transform is suitable for a smooth signal, and wavelet-based transform is more suitable for piecewise constant data. Whereas there are spikes in the sensor data caused by filling zeros for missing data.

To our surprise, Figures 12(c) and 13(a) show that both DCT and wavelet have very high accuracy and recall rates even when more than half the network is faulty. Haar wavelets can maintain a performance of up to 97% for both accuracy and recall. DCT performance is slightly lower but still above 90% for accuracy and 97% for recall. Both of these values decrease gradually as the number of faulty nodes in the network increases. Similar results can be seen for other types of data such as humidity and voltage.

8. Discussion

In this section, we discuss the asymptotic performance of RIDA with different parameters such as the number of nodes in a cluster, the number of hops in the network, and the mapping period. We also discuss several factors that can affect RIDA performance in practice such as optimal logical mapping, quantization, packet loss, and some potential extensions.

8.1. Asymptotic Performance. From (9), considering the *reduction in energy consumption* within one mapping period, we have

$$\begin{aligned} r_h &= \frac{e_b - e_c}{e_b} \\ &= 1 - \Bigg(\Big(n\left(e_s + e_r + e_x\right) m + h\left(e_r + e_x\right) \\ &\qquad + \frac{s}{k}\left[\left(e_s + e_r + e_x\right) n + h\left(e_r + e_x\right) n_i\right] \Big) \\ &\qquad \times \Big(\frac{s}{k} n \left(e_s + \left(e_x + e_r\right) h\right) \Big)^{-1} \Bigg) \\ &\approx 1 - \frac{sm}{kh} - \frac{s}{kn} - \frac{1}{h} - \frac{1}{c}, \end{aligned} \tag{10}$$

where c is the compression ratio.

We can see that the reduction of energy consumption depends on many factors: the number of readings required for remapping m, the ratio between the sensing period s and the mapping period k, the number of hops in the network h, and the compression ratio c. The larger the mapping period is compared to the sensing period, the larger the amount of saved energy is. The more the number of nodes in a cluster is, the more the energy is saved. If the number of hops in the network is very large, the reduction in energy consumption is asymptotically equal to $1 - (s/kn) - (1/c)$. From Figure 11(b), we see that if the environmental conditions change quickly, the correlation between sensor data changes quickly. Under these conditions, RIDA does not compress sensor data well and cannot reduce much energy consumption. Hence, RIDA is suitable for applications where the environmental conditions change slowly. When the environmental conditions change quickly, single-source compression scheme such as SLZW is more suitable because SLZW does not consider the correlation between sensor data at two locations. However, SLZW needs to collect and store enough sensor data to compress. Hence, SLZW delays the delivery of sensor data.

8.2. Optimal Mapping. The logical mapping is basically a permutation function that rearranges the data values. In matrix form, the permutation is the product of a permutation matrix P and the data vector d. If d has length n, P has dimension $n \times n$, and P has exactly only one element of 1 in each row and each column. All the other

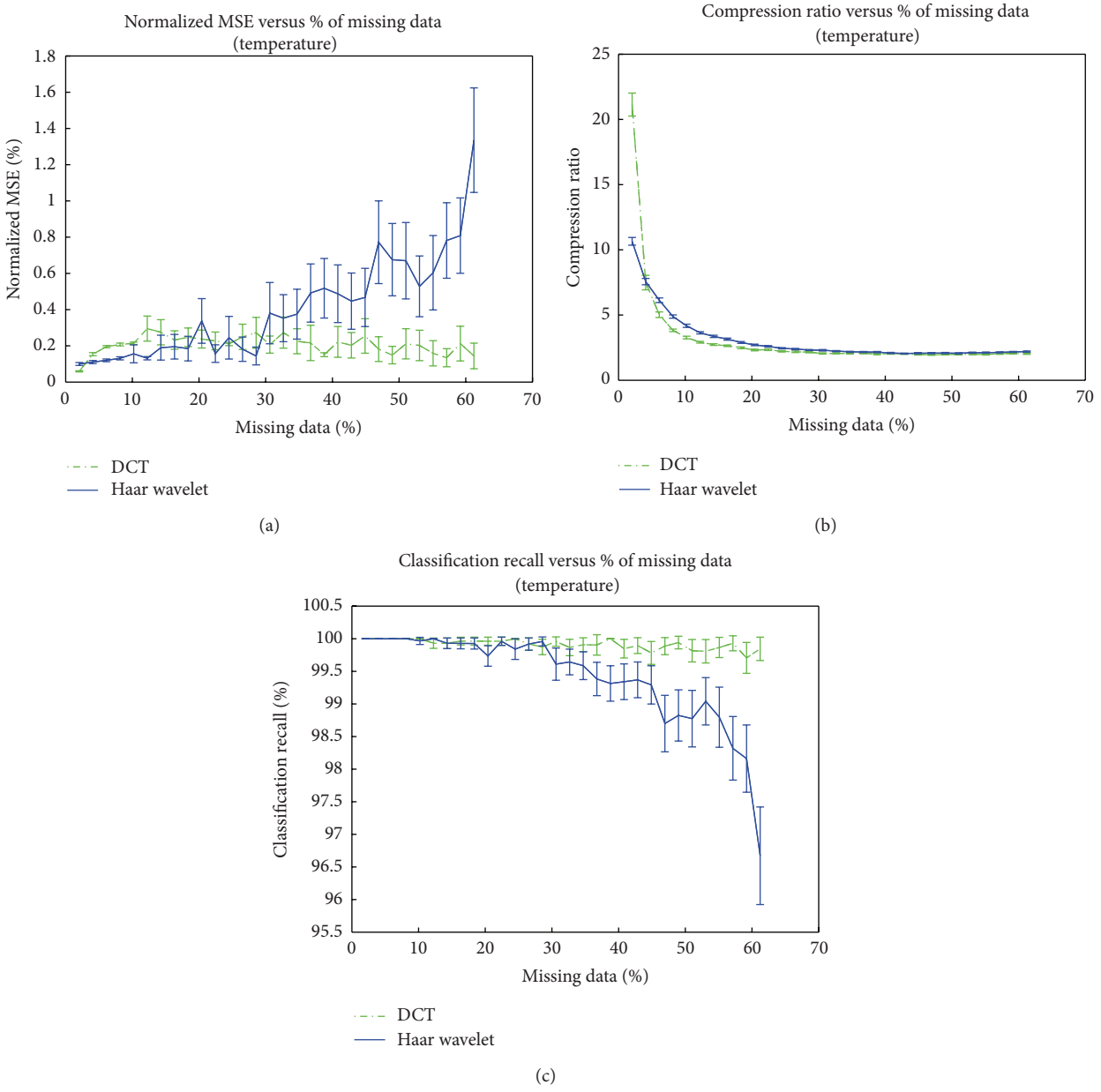

Figure 12: (a) Normalized mean square error. The error increases slightly with the percentage of missing data when RIDA uses DWT. However, the error remains constant when RIDA uses DCT. (b) Compression performance of temperature data. The compression ratios degrade slowly with the percentage of missing data. (c) Classification recall. When half the network data is missing, RIDA can still have a recall of more than 95% using either DCT or DWT.

elements are 0. For example, a 2×2 permutation matrix is

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{11}$$

which will reverse the order of a two-element vector.

Given a basis Ψ and a vector d, we try to permute the vector d in a way that maximizes compressibility. Unfortunately, considering all possible permutation matrices P will take nonpolynomial (NP) time. We have, however, used exhaustive search for determining the optimal P and compared the compression performance with our logical mapping sorting scheme, which is to simply sort the data values and assign indices sequentially. Surprisingly, compression with our sorting scheme is very close to the optimal scheme. Figure 13(b) shows the DCT coefficients on a random set of 8 samples using a logical map created by sorting the samples and an optimal logical map, which we find using exhaustive search. At this stage, we cannot prove for an optimal bound for logical mapping using sorting. However, from the empirical results, we believe that compression with logical mapping using sorting will perform reasonably well in practice. In many

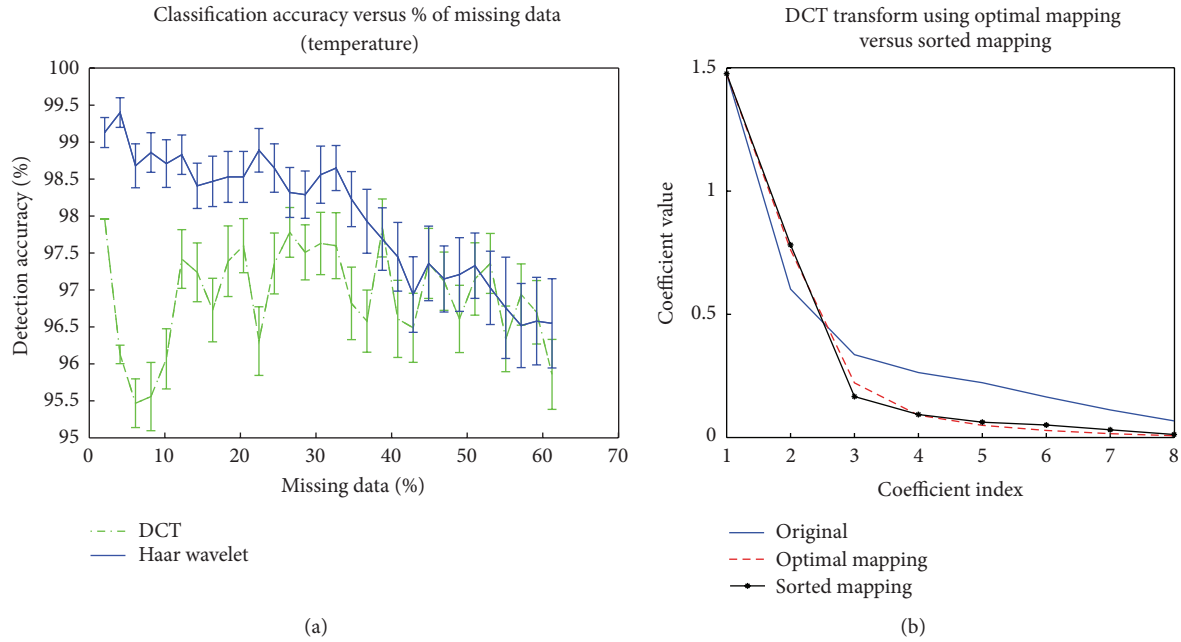

FIGURE 13: (a) Classification accuracy. When half the network data is missing, RIDA can still classify missing data correctly more than 90% of the time using either DCT or DWT. (b) DCT coefficients using optimal mapping versus sorted mapping. Coefficients using DCT transform on sorted data follow closely with those using an optimal mapping.

cases, we found that the optimal logical map is exactly the same as the logical map generated from the sorted samples.

8.3. Quantization. One key part in data compression is quantization, which decides how to drop coefficients that contain the least information about sensor data. The quantization step weights the transformed data using a quantization matrix or a quantization vector. A compression algorithm with a good quantization scheme can compress data well with a small error. However, there is not a systematic method to find the quantization matrix or quantization vector. We often construct a quantization matrix or a quantization vector based on empirical study. For example, the JPEG quantization matrix is based on the smoothness and continuities of natural objects in images. Sensor data is different from images of natural objects. Therefore, we should develop a new quantization scheme for sensor data compression. However, we need to study empirically a large number of sensor data sources to make sure the quantization scheme is representative of sensor data. We would like to investigate a quantization scheme for sensor data compression in the future.

8.4. Impact of Packet Loss. In our experiments, we do not consider packet loss and its impact on decompression error and energy consumption. A simple method assumes sensors retransmit compressed data if there is an error. Using this method, packet loss does not affect decompression error because we transmit all compressed data to the sink. However, this method increases retransmissions and requires acknowledgment mechanisms in routing protocols. We have not yet analyzed the effect of packet loss on energy consumption due to insufficient knowledge about routing protocols.

8.5. Possible Extensions. RIDA has been designed in a completely distributed manner. The only assumption about the network is that it has a cluster topology. Currently, RIDA allows each node to calculate its own coefficient. The total computation complexity is $O(N^2)$. However, if nodes within the cluster can periodically elect a cluster head to perform all the transformation and transmission of the coefficients, the total computation complexity is only $O(N \log(N))$. RIDA can still be applied within the cluster head. This approach requires the cluster to have a protocol to periodically elect the cluster head. Furthermore, if the network is hierarchical where the cluster heads have more computational power than sensor nodes, we can simply apply RIDA at the clusters to minimize the communication cost. Finally, one drawback of organizing sensors into clusters is that RIDA cannot benefit from long-range correlations, where nodes from different clusters might report correlated data. Extending RIDA to exploit long-range correlation is a possible direction for future work.

9. Conclusion

We have designed and evaluated the robust information-driven data compression architecture for energy conservation in irregular wireless sensor networks (RIDA). It uses a logical mapping to rearrange sensor data to improve compression performance. We have successfully adapted two popular data compression methods, DCT and DWT, for our architecture. RIDA is suitable for sensor networks that are dense, have

a cluster topology, and have slowly changing environmental conditions. Our experiments show that using RIDA can double the compression performance of a state-of-the-art distributed data compression method, Wagner-DWT, and eliminate 80% of the energy consumption compared to a sensor network not using RIDA. In addition, RIDA can perform well even when half the sensor data is missing.

Acknowledgments

The authors would like to thank Jonathan Walpole, Andrew Black, and the anonymous reviewers for their suggestions that helped improve the quality of this paper. The research described in this paper was supported by the National Science Foundation Grants NSF 0424602, 05-14818, 07-47442, 07-22063, and 01-21475. Any opinions, findings, conclusions, or recommendations it contains are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] T. He, S. Krishnamurthy, L. Luo et al., "VigilNet: an integrated sensor network system for energy-efficient surveillance," *ACM Transactions on Sensor Networks*, vol. 2, no. 1, pp. 1–38, 2006.

[2] D. B. Hoang and N. Kamyabpour, "An energy driven architecture for wireless sensor networks," *CoRR*, abstract 1206.2005, 2012.

[3] T. B. Matos, A. Brayner, and J. E. B. Maia, "Towards in-network data prediction in wireless sensor networks," in *Proceedings of the 25th Annual ACM Symposium on Applied Computing (SAC '10)*, pp. 592–596, New York, NY, USA, March 2010.

[4] D. Ganesan, D. Estrin, and J. Heidemann, "Dimensions: why do we need a new data handling architecture for sensor networks," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 143–148, 2003.

[5] P. Bonnet, J. Gehrke, and P. Seshadri, "Towards sensor database systems," in *Proceedings of the 2nd International Conference on Mobile Data Management*, pp. 3–14, Hong Kong, January 2001.

[6] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki, "Practical data compression in wireless sensor networks: a survey," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 37–59, 2012.

[7] K. Akkaya, M. Demirbas, and R. S. Aygun, "The impact of data aggregation on the performance of wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 8, no. 2, pp. 171–193, 2008.

[8] X. Xu, S. Wang, X. Mao, S. Tang, and X. Li, "An improved approximation algorithm for data aggregation in multi-hop wireless sensor networks," in *Proceedings of the 2nd ACM International Workshop on Foundations of Wireless Ad Hoc and Sensor Networking and Computing (FOWANC '09)*, pp. 47–56, ACM, New York, NY, USA, May 2009.

[9] Y. Zhu, R. Vedantham, S. J. Park, and R. Sivakumar, "A scalable correlation aware aggregation strategy for wireless sensor networks," *Information Fusion*, vol. 9, no. 3, pp. 354–369, 2008.

[10] S. Madden, M. J. Franklin, J. Hellerstein, and W. Hong, "Tinydb: an acquisitional query processing system for sensor networks," *ACM Transaction on Database System*, vol. 30, no. 1, pp. 122–173, 2005.

[11] C. Cappiello and F. A. Schreiber, "Quality- and energy-aware data compression by aggregation in WSN data streams," in *Proceedings of the 7th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom '09)*, Washington, DC, USA, March 2009.

[12] N. Kimura and S. Latifi, "A survey on data compression in wireless sensor networks," in *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC '05)*, vol. 02, pp. 8–13, IEEE Computer Society, Washington, DC, USA, 2005.

[13] N. Gehrig and P. L. Dragotti, "Distributed compression in camera sensor networks," in *Proceedings of the IEEE 6th Workshop on Multimedia Signal Processing*, pp. 311–314, Siena, Italy, September 2004.

[14] S. S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense micro-sensor network," *IEEE Signal Processing*, vol. 19, no. 2, pp. 51–60, 2002.

[15] A. Ciancio and A. Ortega, "A distributed wavelet compression algorithm for wireless multihop sensor networks using lifting," in *Proceedings of the 30th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '05)*, pp. IV825–IV828, Philadelphia, Pa, USA, March 2005.

[16] C. M. Sadler and M. Martonosi, "Data compression algorithms for energy-constrained devices in delay tolerant networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)*, pp. 265–278, Boulder, Colo, USA, November 2006.

[17] T. Dang, N. Bulusu, and W. Feng, "Rida: a robust informationdriven data compression aorchitecture for irregular wireless sensor networks," in *Proceedings of the 4th the European Conference on Wireless Sensor Networks (EWSN '07)*, pp. 13–149, Delft, The Netherlands, January 2007.

[18] R. Wagner, *Distributed multi-scale data processing for sensor networks [Ph.D. thesis]*, Rice University, Houston, Tex, USA, 2007.

[19] S. Madden, "Intel Research Lab at Berkeley," November 2006, http://db.lcs.mit.edu/labdata/labdata.html.

[20] SensorScope, "Sensorscope wireless distributed sensing sysem for environmental monitoring," April 2008, http://sensorscope.epfl.ch/index.php.

[21] S. K. Mitra, *Digital Signal Processing: A Computer-Based Approach*, Mc Graw Hill, New York, NY, USA, 2006.

[22] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, New York, NY, USA, 1991.

[23] M. Vetterli and J. Kovacevic, *Wavelets and Subband Codding*, Prentice Hall, Upper Saddle River, NJ, USA, 1995.

[24] G. Shen, S. K. Narang, and A. Ortega, "Adaptive distributed transforms for irregularly sampled wireless sensor networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '09)*, pp. 2225–2228, IEEE Computer Society, Washington, DC, USA, April 2009.

[25] G. Shen and A. Ortega, "Joint routing and 2D transform optimization for irregular sensor network grids using wavelet lifting," in *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN '08)*, pp. 183–194, IEEE Computer Society, Washington, DC, USA, April 2008.

[26] C. Luo, F. Wu, J. Sun, and C. W. Chen, "Compressive data gathering for large-scale wireless sensor networks," in *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking (MobiCom '09)*, pp. 145–156, ACM, New York, NY, USA, 2009.

[27] F. Oldewurtel, J. Ansari, and P. Mähönen, "Crosslayer design for distributed source coding in wireless sensor networks," in *Proceedings of the 2nd International Conference on Sensor Technologies and Applications (SENSORCOMM '08)*, pp. 435–443, IEEE Computer Society, Washington, DC, USA, 2008.

[28] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[29] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: universal encoding strategies?" *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

[30] M. F. Duarte, S. Sarvotham, D. Baron, M. B. Wakin, and R. G. Baraniuk, "Distributed compressed sensing of jointly sparse signals," in *Proceedings of the 39th Asilomar Conference on Signals, Systems and Computers*, pp. 1537–1541, Pacific grove, Calif, USA, November 2005.

[31] M. F. Duarte, M. B. Wakin, D. Baron, and R. G. Baraniuk, "Universal distributed sensing via random projections," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN '06)*, pp. 177–185, Nashville, Tenn, USA, April 2006.

[32] M. Rabbat, J. Haupt, A. Singh, and R. Nowak, "Decentralized compression and predistribution via randomized gossiping," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN '06)*, pp. 51–59, Nashville, Tenn, USA, April 2006.

[33] W. Wang, M. Garofalakis, and K. Ramchandran, "Distributed sparse random projections for refinable approximation," in *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN '07)*, pp. 331–339, Cambridge, Mass, USA, April 2007.

[34] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Compressive wireless sensing," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN '06)*, pp. 134–142, April 2006.

[35] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Joint source-channel communication for distributed estimation in sensor networks," *IEEE Transactions on Information Theory*, vol. 53, no. 10, pp. 3629–3653, 2007.

[36] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Siences (HICSS '00)*, vol. 2, p. 223, January 2000.

[37] A. Krause, A. Gupta, C. Guestrin, and J. Kleinberg, "Near-optimal sensor placements: maximizing information while minimizing communication cost," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN '06)*, pp. 2–10, ACM, New York, NY, USA, April 2006.

[38] N. Patwari and A. O. Hero, "Manifold learning algorithms for localization in wireless sensor networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. III857–III860, May 2004.

[39] V. Shnayder, M. Hempstead, B. R. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 188–200, Baltimore, Md, USA, November 2004.

[40] CrossBow, "Crossbow mica2 product webpage," April 2008, http://www.xbow.com/Products/productdetails.aspx?sid=257.

[41] TinyOS-1.x, "Tinyos-1.x official website," April 2008, http://www.tinyos.net/tinyos-1.x/.