Multiresolution Spatial and Temporal Coding in a Wireless Sensor Network for Long-Term Monitoring Applications

You-Chiun Wang, Yao-Yu Hsieh, and Yu-Chee Tseng

Abstract—In many WSN (wireless sensor network) applications, such as [1]–[3], the targets are to provide long-term monitoring of environments. In such applications, energy is a primary concern because sensor nodes have to regularly report data to the sink and need to continuously work for a very long time so that users may periodically request a rough overview of the monitored environment. On the other hand, users may occasionally query more in-depth data of certain areas to analyze abnormal events. These requirements motivate us to propose a *multiresolution compression and query (MRCQ) framework* to support in-network data compression and data storage in WSNs from both space and time domains. Our MRCQ framework can organize sensor nodes hierarchically and establish multiresolution summaries of sensing data inside the network, through spatial and temporal compressions. In the space domain, only lower resolution summaries are sent to the sink; the other higher resolution summaries are stored in the network and can be obtained via queries. In the time domain, historical data stored in sensor nodes exhibits a finer resolution for more recent data, and a coarser resolution for older data. Our methods consider the hardware limitations of sensor nodes. So, the result is expected to save sensors' energy significantly and thus can support long-term monitoring WSN applications. A prototyping system is developed to verify its feasibility. Simulation results also show the efficiency of MRCQ compared to existing work.

Index Terms—coding, data compression, sensor data aggregation, sensor data management, wireless sensor networks.

1 Introduction

Wireless sensor networks (WSNs) provide a new opportunity for pervasive and context-aware monitoring of physical environments. A WSN is composed of numerous *sensor nodes*, each being a tiny wireless device that can continuously collect environment information and report to a remote sink through a multi-hop ad hoc network [4]. A WSN is usually deployed in a region of interest to observe particular phenomena or track objects inside the region. Practical applications of WSNs include, for example, habitat monitoring, health care, smart home, and surveillance [5]–[7].

Because sensor nodes are typically operated by batteries and recharging is usually infeasible, it is a critical issue to extend the network lifetime by conserving their energy. In this paper, we consider WSNs with the following characteristics:

- These WSNs are deployed to support long-term monitoring of specified regions [1]–[3]. Since sensor nodes need to regularly report data to the sink, the communication overhead will dominate their energy consumption. In addition, due to the large amount of such regular reporting, sensor nodes closer to the sink will suffer from heavier traffic loads and thus rapier energy drain [8]. When these nodes exhaust their energy, the network would be broken. Thus, it is important to reduce the amount of transmission of regular reporting of sensor nodes.
- Sensing data often exhibit a certain degree of correlation. The readings of nearby sensor nodes may present high *spatial correlation* due to the similar environment. Besides, the sensing data collected by a single node may present high *temporal correlation* when its surrounding remains stable or changes slowly. Fig. 1 shows an experimental result to support the above argument, where four sensor nodes are used to collect the temperatures of a room in a day. We can observe that the temperatures reported by sensors 1, 2, and 4 are quite spatially correlated because they are close to each other, and the temperatures collected by each individual node during the periods [12:35 PM, 05:35 PM] and [08:05 PM, 01:05 AM] are quite temporally correlated. Therefore, there is large space to compress sensing reports to reduce transmission.
- Users could query different 'resolutions' of sensing data, both spatially and temporally, from the network [9], [10]. In the space domain, they may want to receive a periodical rough report containing an overview of the environment from the sink. Occasionally, they may query more in-depth data of certain areas where

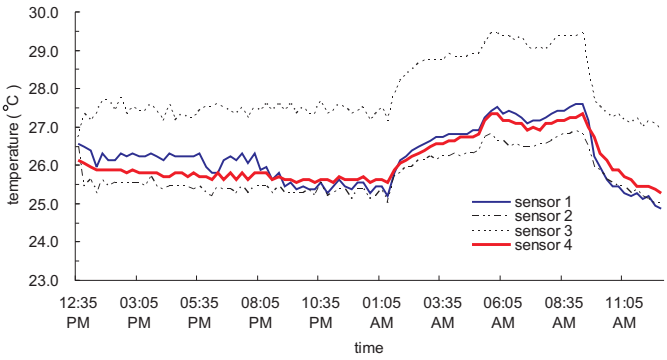


Fig. 1: Indoor temperatures collected by four sensor nodes during a day.

The authors are with the Department of Computer Science, National Chiao-Tung University, Hsin-Chu, 30010, Taiwan.
E-mail: {wangyc, shiehyy, yctseng}@cs.nctu.edu.tw

something abnormal may be happening. As a result, the WSN should be designed to allow periodical lower resolution reports to be sent to the sink as well as to keep higher resolution data inside the network for further queries. In the time domain, they may be more interested in recent sensing data than older ones.

- Sensor nodes are usually simple devices. They have a limited computation power and a smaller memory size. Thus, the proposed compression and storage schemes cannot be too complicated to fit into sensor nodes.

In this paper, we propose a *multiresolution compression and query (MRCQ) framework* to support in-network data compression and data storage in WSNs. The idea is to organize sensor nodes hierarchically and then establish multiresolution summaries of sensing data, via spatial and temporal coding techniques. To reduce communication cost, only lower resolution summaries are sent to the sink. The other higher resolution summaries will be stored at different layers of the network to be retrieved via further queries. The hierarchical architecture of sensor nodes is to support such layering. Data reported from lower layer nodes will be compressed by an upper layer node through a spatial coding technology. In this way, the amount of data transmission can be significantly reduced. Each node also compresses its historical data by a temporal coding technique. To keep historical summaries inside the network, we develop a *reverse-exponential storage algorithm*, where historical data also exhibits a multiresolution characteristic in the sense that finer resolutions are available for more recent data, while coarser resolutions are available for older data.

To summarize, major contributions of this paper are threefold. First, we propose a transmission and storage framework for WSNs to support multiresolution spatial and temporal coding of sensing data. To avoid sensor nodes wasting too much energy on regular reporting, they only report periodical lower resolution summaries to the sink. On the other hand, higher resolution data are kept in the network to allow occasional queries from users. This design can significantly extend a WSN's lifetime, especially in long-term monitoring applications in a somewhat stable or slowly changed environment. Second, we develop in-network spatial and temporal compression algorithms to help reduce data transmission in a WSN. Not only the network lifetime can be extended, but also the network congestion can be alleviated. Compression ratios are tunable, so users are allowed to trade data accuracy for energy consumption according to their requirements. In addition, we also design an efficient storage mechanism to help sensor nodes to maintain their historical sensing data in local memories. The historical data also have multiple resolutions depending on their seniority. Third, the proposed compression and storage algorithms consider the limitation of computation power and memory size of sensor nodes. We have developed a prototyping system to evaluate the feasibility of the MRCQ framework. Extensive simulations are also conducted to verify the efficiency of the proposed algorithms. The results show that MRCQ incurs a lower amount of data transmission and renders more accurate reports compared to DIMENSIONS [11].

The rest of this paper is organized as follows: Section 2 reviews some related work. Section 3 presents our MRCQ framework. Section 4 discusses our prototyping results. Section 5 gives simulation results. Section 6 concludes this paper.

2 Related Work

Data compression has been widely researched in various fields. Some of these concepts have also been applied to WSNs. Below we give a classification and a review.

Text-based compression: The *Lempel-Ziv-Welch (LZW) algorithm* [12] is a popular lossless compression scheme for text data. It is a dictionary-based algorithm that encodes new strings based on previously encountered strings. This concept is adopted by *S-LZW* [13] to reduce data transmission in a WSN. S-LZW treats sensing data as strings and divides them into fixed-size blocks, each being compressed by LZW. Although it is appropriate for sensor nodes, S-LZW does not use the spatial and temporal correlations of sensing data.

Wavelet-based compression: Wavelet-based compression such as *JPEG2000* [14] is designed for image compression. It divides an image into multiple small pixels and compresses them through wavelet transform and quantization [15]. *DIMENSIONS* [11] adopts this concept to support multiresolution storage in a WSN by organizing the network into multiple levels. The *three-dimensional discrete wavelet transform (3D-DWT)* [16] is adopted to generate spatiotemporal summarization of sensing data in each level. Users can obtain different resolutions of sensing reports from different levels via drill-down queries. Although DIMENSIONS meets our multiresolution requirement, it is too complicated for sensor nodes because wavelet-based compression would incur high computation and storage complexity. Also, such expensive wavelet compression and decompression operations are performed at each level of the DIMENSIONS hierarchy.

Distributed source coding: The *Slepian-Wolf theorem* [17] is the foundation of such coding. Given two correlated sources, each being encoded independently, and then decoded jointly at a receiver, the Slepian-Wolf theorem proves that it is possible to achieve lossless encoding of these two sources at a rate equal to their joint entropy, even though there is no negotiation between the two encoders. The theorem and its rate-distortion extension [18] provide a theoretical tool to characterize the amount of communications required for the distributed source coding in a network where nodes will generate highly correlated data [19]. Reference [20] adopts this property to provide distributed compression in a dense sensor network. However, inherited from the Slepian-Wolf theorem, [20] requires prior correlation knowledge of the data to be compressed, which limits its feasibility to be applied to real WSNs.

Compressed sensing: *Compressed sensing* (or *compressive sensing*) [21] is an emerging sampling theory that leverages compressibility without relying on any specific prior knowledge or assumption on signals. It indicates that any sufficiently compressible signal can be accurately recovered from a small number of nonadaptive, randomized linear projection samples. Specifically, given a set of sparse signal $\mathbf{x} = (x_{i,j})_{n\times 1}$, we can find a random projection matrix $\mathbf{A} = (A_{i,j})_{k\times n}$ with far fewer rows than columns (i.e., $k \ll n$) to obtain a small compressed data set $\mathbf{y} = (y_{i,j})_{k\times 1} = \mathbf{Ax}$. This concept is adopted in [22] to provide lossless compression in WSNs. Each of n sensors locally draws a vector with k elements by a random generator to form matrix $\mathbf{A}$, and then uses $\mathbf{A}$ to compress its sensing data. This scheme provides a decentralized compression in WSNs, but it cannot support multiresolution compression.

Data aggregation: In-network data aggregation for WSNs [23]–[27] focuses on reducing the message cost by fusing

TABLE 1: Comparison of prior work and our MRCQ framework.

work	compression technique	lossless compression	low complexity	multiresolution feature
S-LZW [13]	LZW	√	√	
DIMENSIONS [11]	wavelet compression			√
reference [20]	Slepian-Wolf theorem	√		
reference [22]	compressed sensing	√	√	
references [23]–[27]	data aggregation		√	
our MRCQ	DCT, differential, and reverse-exponential		√	√

similar sensing data into some representative values. For example, TAG [23] organizes a sensor network into a tree and proposes SQL-like semantics to aggregate streaming data into histograms. Nevertheless, such a compression provides only one level of resolution.

Our MRCQ framework provides multiresolution compressions in both space and time domains. Our spatial compression algorithm modifies the popular *DCT (discrete cosine transform)* method in the image processing field, and our temporal compression algorithm adopts a differential coding to transmit continuous data and a reverse-exponential concept to store historical data. Table 1 compares the features of prior works and our MRCQ framework. Since DIMENSIONS is the only work that possesses the multiresolution feature, we will compare with it numerically in Section 5.

3 Multiresolution Compression and Query (MRCQ) Framework

Fig. 2 illustrates the system architecture of our MRCQ framework. We assume that sensor nodes are homogeneous and they are arbitrarily deployed in the sensing field. The network is recursively divided into α ($\alpha > 1$) blocks and is organized into multiple *layers*, where a block in layer $i+1$ contains α blocks in layer i. In each layer, we select a node in each block as the *processing node (PN)* to collect and compress sensing reports from lower-layer blocks. The number of layers decides the resolutions and message sizes of sensing reports, and can be adjusted depending on application requirements.

In the lowest layer 1, the PN is responsible for compressing sensing reports from *leaf sensor nodes (LNs)*. The area handled by each PN is divided into $k \times k$ grids (called *pixels*), where k is a small integer. Ideally, each pixel should contain exact one LN and the sensing report of this LN is the pixel's value. Nevertheless, since LNs are randomly deployed, it is possible that some pixels contain no or multiple LNs. In a pixel with multiple LNs, its value is the LNs' average. In a pixel containing no LN, its value can be estimated by some interpolation scheme.

In MRCQ, sensing data is transmitted to the sink layer by layer. There are three algorithms. Data passing each layer will be compressed by its PN through a *spatial compression algorithm* (discussed in Section 3.1). LNs and layer-1 PNs will compress their data by a *temporal compression algorithm* (Section 3.2). Historical data will be stored by each LN and PN via a *reverse-exponential storage algorithm* (Section 3.3). Since the spatial and temporal compression algorithms may cause loss of precision, multiple resolutions can be supported. In particular, as we go deeper into the tree, a finer resolution can be obtained. When a query arrives at a PN, it can reply if its resolution satisfies the requirement of the query. Then, the content of the response will be decompressed at the sink. In this way, both computation and space complexities of PNs are greatly reduced.

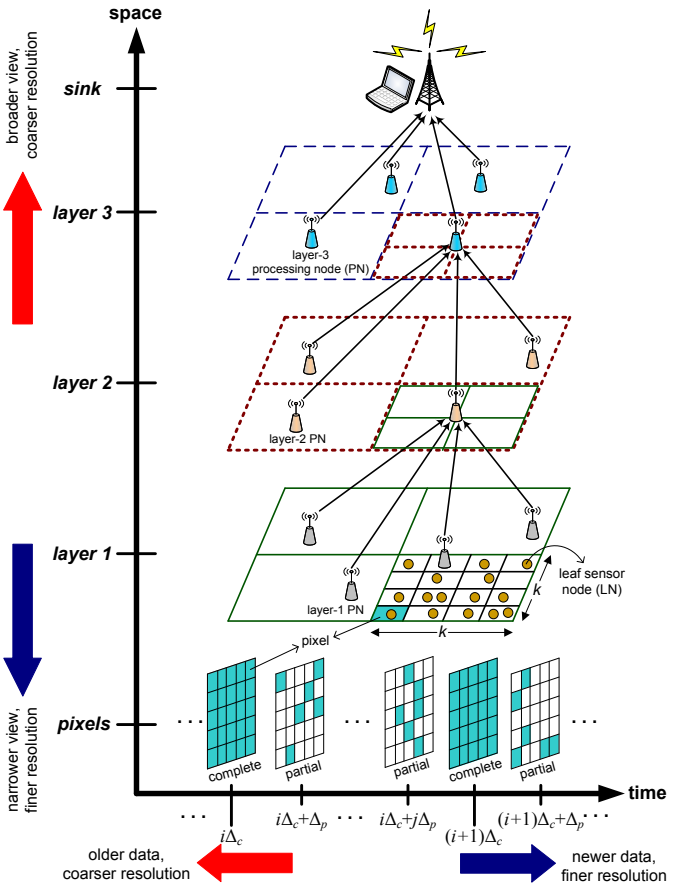


Fig. 2: System architecture of the MRCQ framework (with three layers and $\alpha = 4$).

3.1 Spatial Compression Algorithm

The spatial compression algorithm is performed by each PN to compress sensing data from its lower layer. A *compression ratio* γ ($0 < \gamma \leq 1$) can be specified, which is defined as the ratio of the size of compressed data to the size of uncompressed data. The spatial correlation of data is exploited in the compression. There are three components: *layer-1 compression*, *layer-i compression* ($i > 1$), and *decompression*.

3.1.1 Layer-1 Compression

A layer-1 PN collects the sensing data from its local LNs and stores them in a $k \times k$ matrix $\mathcal{M} = (s_{i,j})_{k \times k}$, where $s_{i,j}$ is the value of the local pixel (i, j). Then, we apply the *two-dimensional discrete cosine transform (2D-DCT)* [28] on $\mathcal{M}$ to generate a new matrix $\mathcal{M}' = (t_{i,j})_{k \times k}$, where

$$t_{i,j} = \frac{2}{k} C(i) C(j) \sum_{x=0}^{k-1} \sum_{y=0}^{k-1} s_{x,y} \cdot \cos(i\pi \frac{2x+1}{2k}) \cos(j\pi \frac{2y+1}{2k}), \quad (1)$$

where $C(i) = \frac{1}{\sqrt{2}}$ if $i = 0$ and $C(i) = 0$ otherwise. The 2D-DCT is widely used in image processing. It can transform an image from the spatial domain to the frequency domain and extract significant values of the image. In particular, those significant values will appear in the upper left part of matrix $\mathcal{M}'$, while insignificant values will appear in the opposite part. Therefore, we can preserve most characteristics of $\mathcal{M}$ by truncating the lower right part of $\mathcal{M}'$ for compression purpose.

The cosine operations in Eq. (1) might be too costly for sensor nodes. Fortunately, the variable k is a pre-defined system parameter. Since k is a small integer, we can maintain a small table in each PN to record the results of cosine operations for each (i, x) and (j, y) pair. Thus, the calculation of Eq. (1) can be reduced to simple addition and multiplication operations.

After calculating $\mathcal{M}'$, a *reduced zigzag scan (RZS)* is performed to translate $\mathcal{M}'$ into a one-dimensional array $\mathcal{D}$. RZS retrieves elements of $\mathcal{M}'$ from the upper left corner toward the lower right corner along the diagonal direction, as shown in Fig. 3, until $\lceil \gamma \cdot k^2 \rceil$ elements are scanned. Then, the array $\mathcal{D}$ is transmitted to its layer-2 PN. Due to the property of 2D-DCT, array $\mathcal{D}$ keeps most significant values of $\mathcal{M}$.

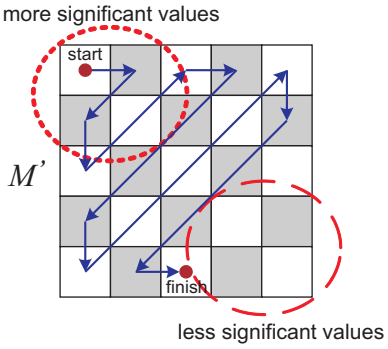


Fig. 3: An example of RZS with compression ratio $\gamma = \frac{20}{25} = 0.8$.

3.1.2 Layer-i Compression

A layer-i ($i > 1$) PN will further compress the data from its α child blocks in layer $i-1$. Intuitively, one possible solution is: 1) decompress the block from each lower-layer node, 2) combine all α blocks together to form a larger block, and 3) apply the 2D-DCT method again on the larger block. Nevertheless, this solution has two drawbacks. First, the 2D-DCT decompression and compression are too expensive. The situation becomes worse as we move to the higher layers. Second, the effectiveness of compression may degrade because in a larger area, the degree of spatial correlation of sensing data will decrease.

Here, we propose a simple layer-i compression scheme, as shown in Fig. 4. Specifically, for each layer-2 PN, it will collect reduced matrices $\mathcal{M}'$ from its α layer-1 PNs, each with $\lceil \gamma \cdot k^2 \rceil$ pixels. For each reduced layer-1 matrix $\mathcal{M}'$, we transmit the first $\lceil \gamma(\gamma \cdot k^2) \rceil$ significant pixels and discard the remaining $\lfloor \gamma \cdot k^2 - \gamma^2 \cdot k^2 \rfloor$ pixels[1], as shown in Fig. 4. So, only $\alpha \cdot \lceil \gamma^2 \cdot k^2 \rceil$ pixels will be sent to its layer-3 PN. Similarly, for each layer-i PN, it will collect α^{i-1} layer-1 matrices, and only preserve the first most significant $\lceil \gamma^i \cdot k^2 \rceil$ pixels of each layer-1 matrix to be sent to its layer-$(i+1)$ PN and discard the remaining $\lfloor \gamma^{i-1} \cdot k^2 - \gamma^i \cdot k^2 \rfloor$ pixels. The above scheme incurs quite low

1. These discarded pixels will be stored in the PN's local memory for further queries.

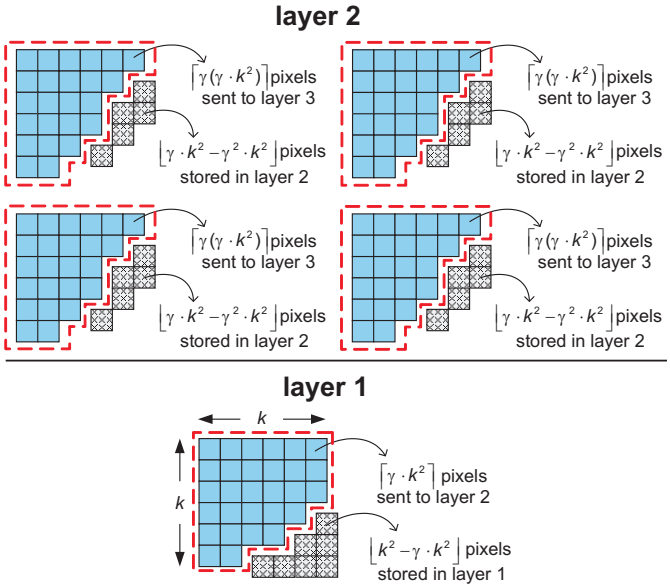


Fig. 4: Layer-i compression scheme.

computation cost. Besides, since entries of a layer-1 matrix are more spatially correlated, such a compression is more efficient.

Next, we analyze the size of packets transmitted in each layer. Suppose that the size of a pixel is l bits and each packet header is h bits. Given a compression ratio of γ, the size of each packet transmitted by a layer-1 PN is $h + \lceil \gamma \cdot k^2 l \rceil$ bits. For each layer-2 PN, after including a packet header, $h + \alpha \cdot \lceil \gamma^2 \cdot k^2 l \rceil$ bits will be sent to its parent. Similarly, the size of each packet transmitted by a layer-i PN is $h + \alpha^{i-1} \cdot \lceil \gamma^i \cdot k^2 l \rceil$ bits. In summary, excluding packet headers, only a γ^i ratio of the amount of original sensing data is transmitted by layer-i PNs. A smaller γ incurs less amount of data transmission and thus preserves more energy of PNs, but it also reduces data accuracy. We will discuss this tradeoff by simulations in Section 5.

3.1.3 Decompression

There are two cases where decompression may be taken. The first case is at the sink based on the α^{d-1} reduced layer-1 matrices collected from its children, each with $\lceil \gamma^d \cdot k^2 \rceil$ pixels, where d is the number of layers. Then, each reduced matrix will be expanded to a $k \times k$ matrix $\mathcal{M}' = (t_{i,j})_{k \times k}$ by appending sufficient 0's at the end. We then adopt the *inverse 2D-DCT* to transform $\mathcal{M}'$ to a matrix $\mathcal{M} = (s_{i,j})_{k \times k}$, where

$$s_{i,j} = \frac{2}{k} \sum_{x=0}^{k-1} \sum_{y=0}^{k-1} C(x)C(y) \cdot t_{x,y} \cdot \cos(x\pi \frac{2i+1}{2k}) \cos(y\pi \frac{2j+1}{2k}). \quad (2)$$

Since Eq. (2) is the inverse of Eq. (1), we can obtain an approximation of the original matrix of sensing data. The sink then puts all these recovered α^{d-1} matrices together to form a large matrix of approximate sensing data.

The second case happens when we query a certain layer i. A query will be flooded from the sink to all layer-i PNs. Then, each layer-i PN will send the discarded part ($\lfloor \gamma^{i-1} \cdot k^2 - \gamma^i \cdot k^2 \rfloor$ pixels, as shown in Fig. 4) to its layer-$(i+1)$ PN. Such operation is repeated until the sink receives all discarded pixels of all PNs from layers i to d. In this way, the sink can have the complete matrices seen by all layer-i PNs and then recover the sensing

data with a resolution of layer i. As can be seen, the above scheme requires each PN to transmit only few pixels.

3.2 Temporal Compression Algorithm

The aforementioned discussion assumes that LNs and PNs will periodically report data to their parents. In this section, we propose a temporal compression algorithm, which tries to reduce the amount of transmission by exploiting the similarity of data items that are generated at close times.

The concept of temporal compression is shown in Fig. 2. The compression is done only between LNs and layer-1 PNs and between layer-1 PNs and layer-2 PNs. The time axis is divided into *complete reporting intervals* of the same length Δ_c. Each complete reporting interval is further divided into smaller *partial reporting intervals* of length Δ_p, where Δ_c is a multiple of Δ_p. In the beginning of each complete reporting interval, LNs and PNs will report and compress data as we discussed earlier. During each complete reporting interval, differential compression will be conducted in the beginning of each partial reporting interval. Specifically, each LN will decide to report or not to report according to the variance of its current sensing data and its previous sensing data. If an LN does not report, its layer-1 PN will assume that its sensing data are unchanged. Similarly, a layer-1 PN will do the same thing to its layer-2 PN.

The compression between LNs and layer-1 PNs will be controlled by a small update threshold δ_L. An LN will not report if its current sensing data v_{current} differs from its previous reported data v_{rep} by an amount no more than δ_L, i.e., $|v_{\text{rep}} - v_{\text{current}}| \leq \delta_L$. If so, its layer-1 PN will use v_{rep} as its current sensing data. The compression between layer-1 PNs and layer-2 PNs will be controlled by a threshold δ_P. A layer-1 PN will not report if the difference between its current matrix $M_{\text{current}} = (t_{i,j})_{k\times k}$ and its previously report matrix $M_{\text{rep}} = (s_{i,j})_{k\times k}$ satisfies the inequality

$$\frac{1}{k^2}\sum_{i=1}^{k}\sum_{j=1}^{k}|s_{i,j} - t_{i,j}| \leq \delta_P.$$

If so, its layer-2 PN will use M_{rep} as its current sensing matrix.

Note that all layer-i PNs, $i \geq 2$, will not conduct temporal compression because the matrices seen by such nodes have already been compressed by the 2D-DCT method and computing the difference of two compressed matrices is time-consuming.

Remark 1. Since sensor nodes will regularly report their sensing data, we can set up timers at PNs and the sink, and apply a retransmission mechanism to handle the packet loss problem. The length of timeout can be set to the reporting interval Δ_c. When a node does not respond after a predefined number of retransmission requests, it is treated as failure. In the case of an LN failure, the layer-1 PN can use interpolation to estimate its value. In the case of a PN failure, a new PN can be elected to replace the old one.

3.3 Reverse-Exponential Storage Algorithm

The above compression and decompression algorithms only concern the current sensing data. In fact, sensing data is usually a streaming data. Thus, it is a challenging issue to store historical data in PNs and LNs under sensors' limited storages.

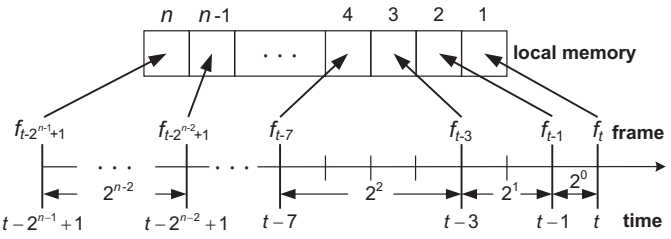


Fig. 5: Concept of the reverse-exponential storage algorithm.

In this section, we propose a *reverse-exponential storage algorithm* for this purpose. Thus, users can query different resolutions of sensing data on the time domain.

Let n_L and n_P be the maximum numbers of frames that an LN and a PN can store in its local memory, respectively, where a *frame* is the unit of sensing data for the node. Below, for simplicity, we will write both n_L and n_P as n (it will be clear from the context). For an LN, a frame is a piece of sensing data generated by itself. For a PN, a frame is a set of discarded pixels in the compression process (refer to Fig. 4). Let f_i be the frame stored by an LN/PN at timestamp i. Suppose that the current time is t. The objective of the reverse-exponential storage algorithm is to store historical frames at timestamps with intervals at an exponentially increasing order from t. Specifically, we would like to store frames $f_{t-2^{n-1}+1}$, $\cdots$, f_{t-3}, f_{t-1}, and f_t in the node, as shown in Fig. 5. In this way, users can query sensing data long time ago with different resolutions. When a query of a past frame f_i $(i \leq t)$ arrives at a node, two cases may happen:

Case A: The node is an LN. The response includes three possibilities:

- If f_i is stored in the node's local memory, it can directly reply f_i to the sink.
- If $t - 2^j + 1 < i < t - 2^{j-1} + 1$, it replies two frames f_{t-2^j+1} and $f_{t-2^{j-1}+1}$ to the sink. The sink then applies a *linear interpolation* to calculate f_i, that is,

$$\frac{f_i - f_{t-2^j+1}}{i-(t-2^i+1)} = \frac{f_{t-2^{j-1}+1} - f_{t-2^j+1}}{t-2^{j-1}+1-(t-2^j+1)}$$
$$\Rightarrow f_i = f_{t-2^j+1} + \frac{(f_{t-2^{j-1}+1} - f_{t-2^j+1}) \times (i-t+2^j-1)}{2^{j-1}}. \quad (3)$$

- If $i < t - 2^{n-1} + 1$, it replies a FAIL message to the sink since this information is too old.

Case B: The node is a PN. The query should specify a certain layer i. First, the sink will flood the query to all layer-i PNs. Then, each layer-i PN will send its frame(s) or a FAIL message according to the above three cases to its layer-$(i+1)$ PN. Note that since data stored in PNs are all compressed data, each layer-$(i+1)$ PN also needs to send the similar frame(s) to its parent. Such operation is repeated until the sink receives all frames from all layer-i to layer-d PNs. Then, the sink can combine these frames and recover the historical data via the inverse 2D-DCT method and a linear interpolation (as in Eq. (3)).

There are two properties in the reverse-exponential storage scheme. First, a long history of data can be stored with small buffers. Second, finer resolutions are available for more recent data, while coarser resolutions are available for older data.

The remaining problem is to maintain historical frames in a node's memory as time moves on. There are two cases to be discussed.

Case A: The node is an LN. Suppose that the current time is t. What are stored locally are frames $f_{t-2^{n-1}+1}, \cdots, f_{t-3}, f_{t-1}$, and f_t. As the time moves to $t+1$, each frame is aged by one. So, the place for f_t should be given to f_{t+1}, and the place for f_{t-1} should be given to f_t. For each remaining frame $f_{t-2^\beta+2}$, $\beta = 2, \cdots, n-1$, we can apply the linear interpolation in Eq. (3) on the frames $f_{t-2^\beta+1}$ and $f_{t-2^{\beta-1}+1}$ to approximate its value. For example, f_{t-2} can be interpolated from f_{t-3} and f_{t-1}.

Case B: The node is a PN. Recall that frames $f_{t-2^{n-1}+1}, \cdots, f_{t-3}, f_{t-1}$, and f_t are stored locally, where t is the current time. As the time moves to $t+1$, the place for f_t should be given to f_{t+1}, and the place for f_{t-1} should be given to f_t. For each remaining frame $f_{t-2^\beta+1}$, $\beta = 2, \cdots, n-1$, we need to select one frame that has the closest timestamp to it to represent this frame[2]. Specifically, for each $\beta = n-1, \cdots, 2$, the place for frame $f_{t-2^\beta+1}$ will be given to the frame whose original timestamp is closest to $t - 2^\beta + 1$ (to make this possible, we need to store each frame's original timestamp). One exception is frame $f_{t-2^{n-1}+1}$. We will not consider frames older than timestamp $t-2^{n-1}+1$ (because such frames are too old). Fig. 6 illustrates an example, where the number in each box represents the real age of a frame at each time instance. Suppose that at time t, we have historical frames with ages 1, 2, 4, 8, and 16 in locations $f_t, f_{t-1}, f_{t-3}, f_{t-7}$, and f_{t-15}, respectively. As the time moves to $t+1$, the age of each stored frame is increased by one. So, frames with ages 1, 2, 3, 5, and 9 are kept. The frame with age 17 is deleted according to our exceptional rule. Fig. 6 shows the buffering results from t to $t+10$. We can observe that the frames stored in the places f_t and f_{t-1} are always accurate. For each place $f_{t-2^\beta+1}$, $\beta = 2, \cdots, n-1$, the actual frame f_a stored in that place always satisfies a timestamp $|a - (t - 2^\beta + 1)| \leq 2^{\beta-1}$, because the original frame stored in the place $f_{t-2^{\beta-1}+1}$ will eventually move to the place $f_{t-2^\beta+1}$.

Finally, we comment on the values of L_N and L_P. Suppose that each node has a buffer space of m bits and the size of a pixel is l bits. Clearly, $L_N = \lfloor \frac{m}{l} \rfloor$. For L_P, recall Fig. 4. Each layer-i PN should keep $\alpha^{i-1} \cdot \lfloor (\gamma^{i-1} - \gamma^i) \cdot k^2 \rfloor$ pixels after each compression. Therefore, we have

$$L_P = \left\lfloor \frac{m}{\max_{i=1}^{d} \{\alpha^{i-1} \cdot \lfloor (\gamma^{i-1} - \gamma^i) \cdot k^2 \rfloor \cdot l\}} \right\rfloor.$$

4 Prototyping Experience

We use the MICAz Motes [29] to build a two-tier, 16-node WSN, as shown in Fig. 7. Each Mote's radio is a 2.4 GHz, IEEE 802.15.4-compatible module allowing low-power operations and offering a data rate of 250 Kbps. We set $\alpha = 4$ and $k = 2$, so there are four layer-1 PNs in our prototype, each being responsible for collecting and compressing 2×2 pieces of data. We use this network to collect the indoor temperatures during 45 hours. The complete reporting interval Δ_c is set to ten minutes. The compression ratio γ is set to 0.75 and the update thresholds δ_L and δ_P are set to 0.5 °C. For each LN, a sensing report is 15 bytes, which contains 11 bytes of header and trailer and 4 bytes of payload. The size of packets reported from a PN is 19 bytes with 8 bytes of payload.

2. Note that linear interpolation is infeasible here because these data are compressed data.

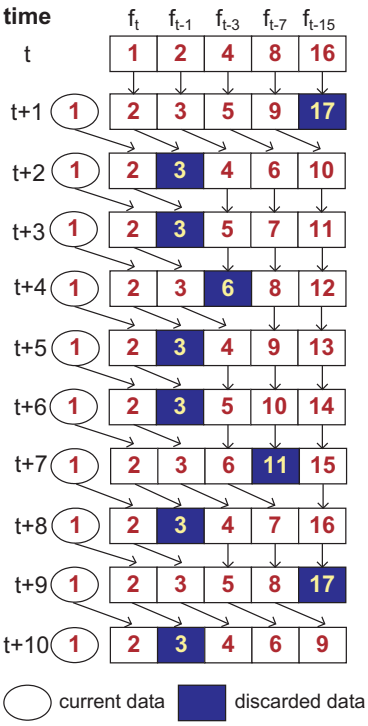


Fig. 6: An example of historical data buffering in a PN.

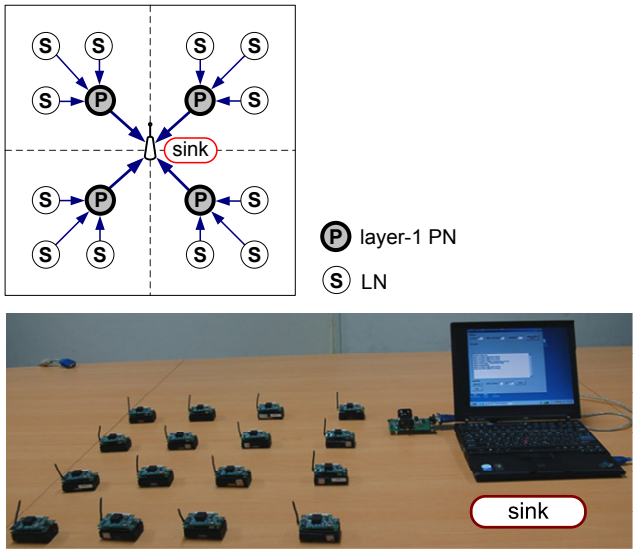
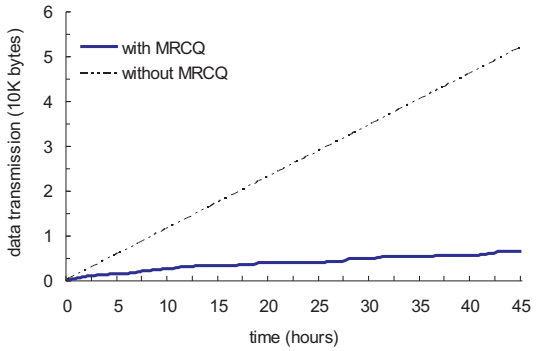


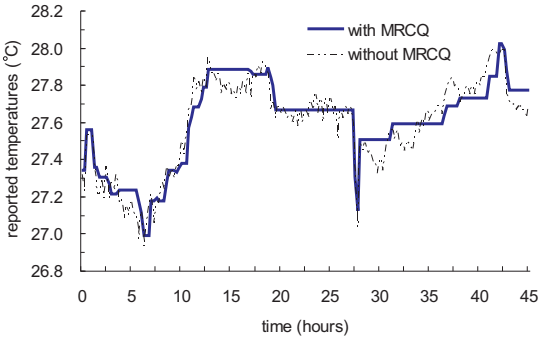
Fig. 7: A 16-mote prototype in our experiment.

Fig. 8(a) shows the total amount of data transmitted by these 16 nodes. We can observe that the amount of data transmission is greatly reduced by our MRCQ framework. This is because temperatures have high data correlations and thus can be compressed in both spatial and temporal domains. Fig. 8(b) illustrates the average temperatures being reported by the 16 nodes. The maximum error is 0.189 °C. This indicates that our MRCQ framework can preserve most characteristics of the sensing reports.

At the remote sink, we provide a user interface to monitor the network's status and to query data from sensor nodes, as shown in Fig. 9. It contains three major components: *monitoring*, *statistical*, and *querying* areas. The monitoring area shows the network topology and status. Inside each square, the corresponding sensor ID, coordinates, and current temperature



(a) amount of data transmission



(b) reported temperatures

Fig. 8: Experimental results from the prototyping system.

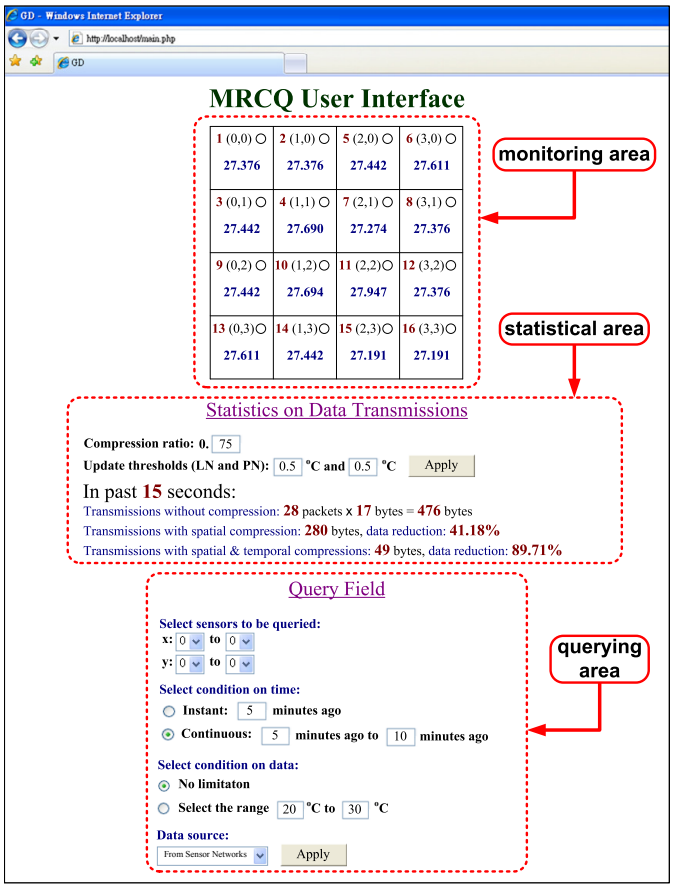


Fig. 9: User interface at the remote sink in our prototype.

are shown. The color of a circle means the corresponding sensor's status (black = inactive). The statistical area reports the total amount of data transmission when different compression algorithms are applied. The querying area allows users to obtain more in-depth sensing data from the network. Users can specify the sensor nodes to be queried and add conditions to restrict the queried data.

5 Simulation Studies

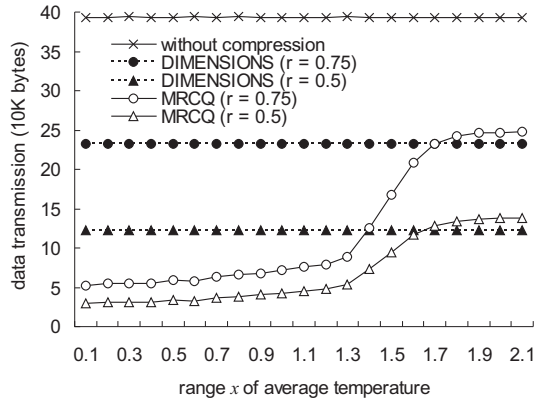
Since a large-scale deployment is difficult to realize, in this section, we develop a simulator to verify the efficiency of our MRCQ framework. We set up a 256×256 m^2 sensing field, on which 1000 sensor nodes are randomly deployed. We set $k = 8$ and $\alpha = 4$, and designate 20 and 4 nodes as layer-1 and layer-2 PNs, respectively. The transmission range of each sensor node is set to 30 m. The size of each packet transmitted by a PN is 18 bytes containing 16 bytes of payload, whereas the size of each sensing data reported by an LN is 6 bytes containing 4 bytes of payload. The sensing field is divided into 32×32 grids. The total simulation time is 100 minutes. During every minute, the temperature of each grid may be changed and a number of events will arbitrarily occur in the sensing field. We measure the total amount of data being transmitted by sensor nodes and the average errors caused by the compression algorithms. Here, we define the *average error* as $\frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K}|M_{\text{sink}}[i,j] - M_{\text{real}}[i,j]|$, where M_{sink} is a $K \times K$ matrix of sensing data seen by the sink and M_{real} is a matrix of the real temperatures seen by LNs. In these simulations, $K = 32$.

5.1 Comparison with DIMENSIONS

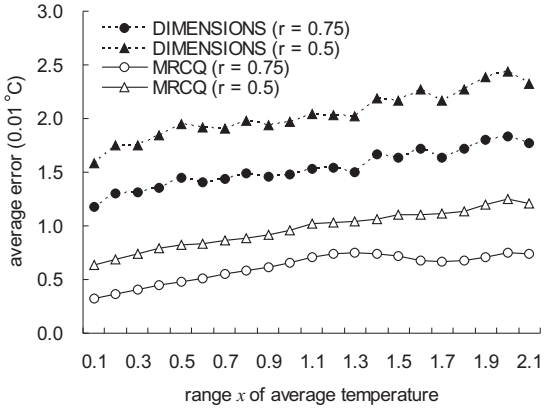
We compare our MRCQ framework against DIMENSIONS [11]. The compression ratio γ is set as 0.75 and 0.5 in both MRCQ and DIMENSIONS. For MRCQ, we set the update thresholds δ_L and δ_P as 0.5 °C in the temporal compression algorithm. Three different environmental scenarios are considered in this experiment.

In the first scenario, we observe the effect of the ranges of grid temperatures. Specifically, the average temperature of each grid is randomly picked from $[(25-x)°\text{C}, (25+x)°\text{C}]$, where the range x is selected from 0.1 to 2.1. In addition, there are 5 events arbitrarily occurring in the sensing field, each increasing $[1°\text{C}, 3°\text{C}]$ in its vicinity. Clearly, when the value of x is larger, it means that the environment changes more drastically. Fig. 10(a) shows the total amount of data transmission of MRCQ and DIMENSIONS. As can be seen, when the compression ratio γ becomes smaller, more data can be compressed. Our MRCQ framework can have less amount of data transmission compared with DIMENSIONS when $x \leq 1.6$. This means that MRCQ can compress more data when the environment is more stable. When the environment changes more drastically (e.g., $x \geq 1.7$), MRCQ will transmit more data to respond to the change. This will reflect on the average error of reporting data, as shown in Fig. 10(b). We can observe that MRCQ always has a lower average error compared with DIMENSIONS, even when the value of x exceeds 1.7.

In Fig. 10(a), we can observe that the amount of data transmission in DIMENSIONS is irrelevant to the range of grid temperatures, because it periodically compresses data in a constant manner. This will make DIMENSIONS difficult to respond to environmental changes, and causes more data inaccuracy when the environment is more unstable. On the other



(a) amount of data transmission

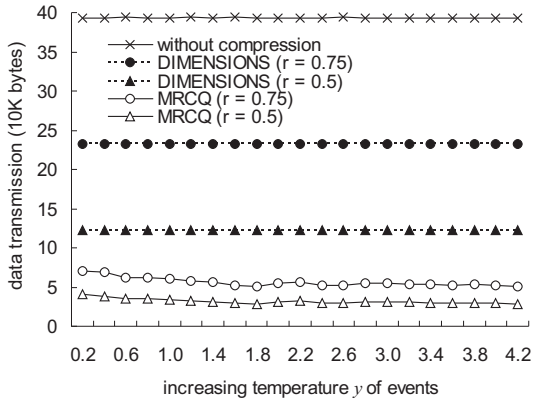


(b) average errors

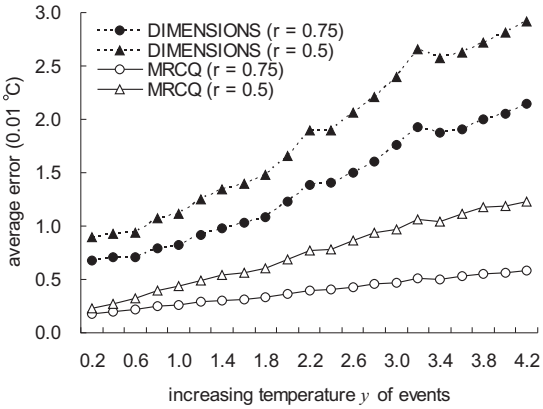
Fig. 10: Comparison of MRCQ and DIMENSIONS under different ranges of grid temperatures.

hand, the temporal compression algorithm can help MRCQ flexibly respond to environmental changes. In Fig. 10(a), when the environment is almost stable (e.g., $x \leq 1.3$), the temporal compression algorithm can help MRCQ reduce a large amount of data transmission. When $1.4 \leq x \leq 1.8$, since the difference between successive sensing data in each grid could often exceed the threshold δ_L, the effect of temporal compression decreases and thus the amount of data transmission in MRCQ grows. As $x \geq 1.9$, the temporal compression becomes almost of no effect so the data compression in MRCQ is dominated by the spatial compression. Thus, the amount of data transmission in MRCQ becomes constant. The above behavior can help MRCQ flexibly reduce more data when the environment is stable and render more accurate reports when the environment changes drastically.

In the second scenario, we observe the effect of the increasing temperatures of events. Specifically, there are 20 events arbitrarily appearing in the sensing field, and each event will cause an increase of $y°\mathrm{C}$ in its vicinity, where y is ranged from 0.2 to 4.2. The average temperature of each grid is randomly selected from $[24.8°\mathrm{C}, 25.2°\mathrm{C}]$. Clearly, a larger value of y means that the events will cause a larger difference in temperatures in their vicinity. Fig. 11 shows the amount of data transmission and average errors of MRCQ and DIMENSIONS. We can observe that MRCQ outperforms DIMENSIONS in this scenario. Since the temperature range in each grid is limited to $0.4°\mathrm{C}$ and the number of events is fixed, the effect of temporal compression in MRCQ will be almost constant. So, the amount of data transmission changes slowly in MRCQ, as shown in Fig. 11(a). The average errors of MRCQ and DIMENSIONS increase as y increases, because these events will introduce more difference in temperatures inside their neighboring regions.



(a) amount of data transmission



(b) average errors

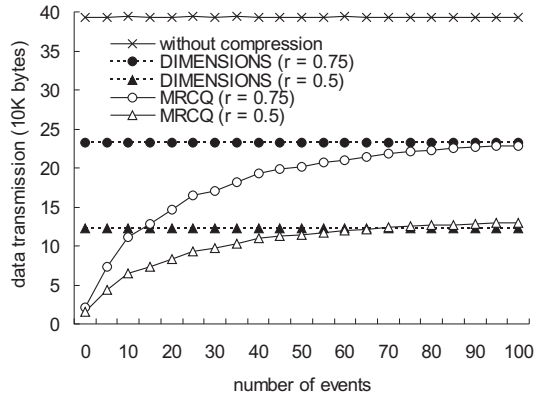
Fig. 11: Comparison of MRCQ and DIMENSIONS under different increasing temperatures of events.

In the third scenario, we observe the effect of the number of events. The average temperature of each grid is randomly set among $[24°\mathrm{C}, 26°\mathrm{C}]$. In addition, there are 0 to 100 events randomly occurring in the sensing field. When an event happens, an increase of $1°\mathrm{C}$ to $3°\mathrm{C}$ can be seen in its vicinity. Fig. 12(a) shows the total amount of data transmission in MRCQ and DIMENSIONS. We can observe that MRCQ has less amount of data transmission compared with DIMENSIONS when $\gamma = 0.75$ and when the number of events is smaller than 70 as $\gamma = 0.5$. Similar to the first scenario, the amount of data transmission grows in MRCQ as the number of events increases, because of the effect of the temporal compression algorithm. This flexibility can help MRCQ result in a smaller average error when there are more events, as shown in Fig. 12(b).

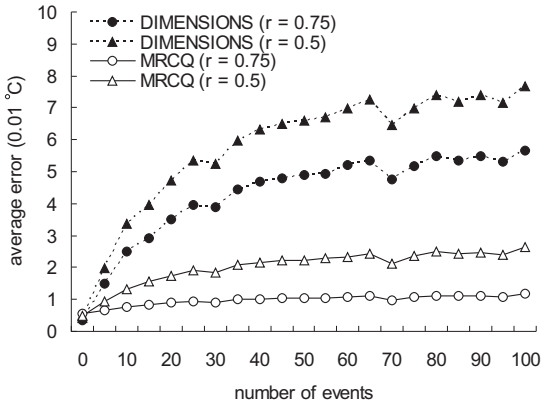
In summary, MRCQ can compresses more data when the environment is stable, and preserve more accuracy on reports by transmitting more data when the environment changes drastically. In the following experiments, we will measure the effects of spatial and temporal compression algorithms in MRCQ. The simulation settings are the same as that in the third scenario.

5.2 Effect of the Spatial Compression Algorithm

We then evaluate the effect of the spatial compression algorithm in MRCQ. We set the update thresholds δ_L and δ_P

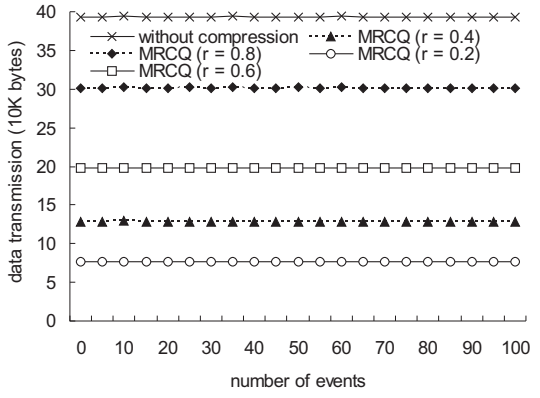


(a) amount of data transmission

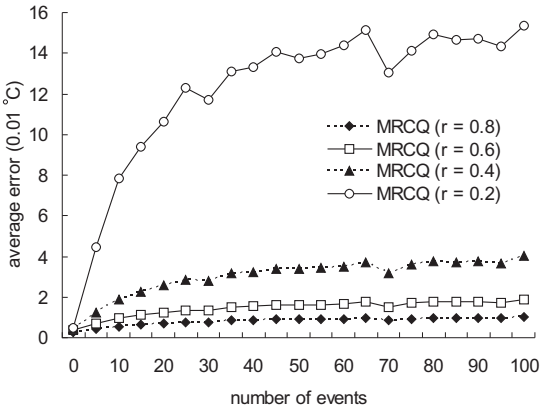


(b) average errors

Fig. 12: Comparison of MRCQ and DIMENSIONS under different numbers of events.



(a) amount of data transmission



(b) average errors

Fig. 13: Effect of the spatial compression algorithm.

as zero to eliminate the effect of the temporal compression algorithm. Fig. 13(a) shows the total amount of data transmission of MRCQ under different number of events and different compression ratios γ. Clearly, when γ becomes smaller, the spatial compression algorithm can reduce more data. The data transmission of the spatial compression algorithm is irrelevant to the number of events because it discards a constant number of pixels depending on the value of γ. However, this behavior will affect the average errors, as shown in Fig. 13(b). We can observe that when the value of γ becomes larger, the spatial compression algorithm can keep lower errors because PNs can transmit more data to reflect the changes of environment. However, when γ becomes too small (for example, $\gamma = 0.2$), the average error grows fast as the number of events increases. In this case, since each PN can only transmit a small portion of the compressed matrix, only a small number of significant values can be kept and thus the error increases.

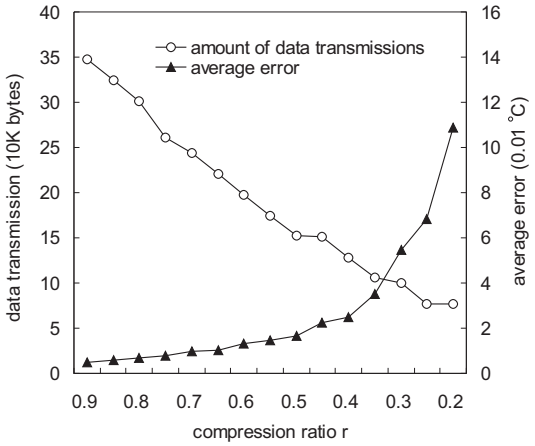
Fig. 14 shows the effect of compression ratio γ on the spatial compression algorithm when the number of events are 20 and 80. From Fig. 14, we can find that the suitable value of γ is around 0.35 to 0.4 since both the amount of data transmission and average error can be kept quite small.

5.3 Effect of the Temporal Compression Algorithm

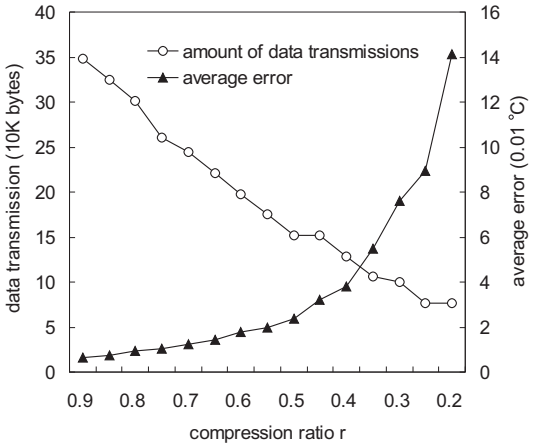
Finally, we evaluate the effect of the temporal compression algorithm in MRCQ. We set the compression ratio $\gamma = 0.8$ to fix the effect of the spatial compression algorithm and set $\delta_L = \delta_P = \delta$. Fig. 15(a) shows the total amount of data transmission of MRCQ under different number of events and different update thresholds δ. We can observe that when δ is very small (for example, $\delta = 0.1$), the effect of the temporal compression algorithm becomes insignificant and thus the amount of data transmission remains constant under different number of events. However, as δ becomes larger, the temporal compression algorithm can reduce more data when the number of events is smaller. When the number of events increases, the amount of data transmission also increases because the difference between two sequential sensing reports may often exceed the update threshold. Fig. 15(b) shows the average errors. We can observe that the errors can be kept quite small because in the temporal compression algorithm, LNs and layer-1 PNs will periodically report complete data to their upper PNs.

6 Conclusions

In this paper, we have proposed an MRCQ framework to provide multiresolution data compression and data storage in a WSN by spatial and temporal coding techniques. The proposed multiresolution idea can significantly extend a WSN's lifetime, especially in long-term monitoring applications with a slowly changed environment. Our in-network compression algorithms adopt the concepts of DCT and differential coding to reduce data redundancy. Our storage algorithm helps sensor nodes to store historical data in their small memories by a reverse-exponential solution. We have implemented a prototyping system on the MICAz platform to demonstrate the flexibility of MRCQ. Extensive simulation results have also been presented to verify the efficiency of MRCQ. They show that our MRCQ framework can flexibly adjust the amount of

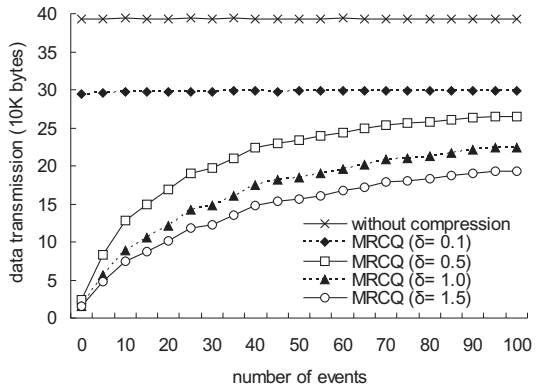


(a) event number: 20

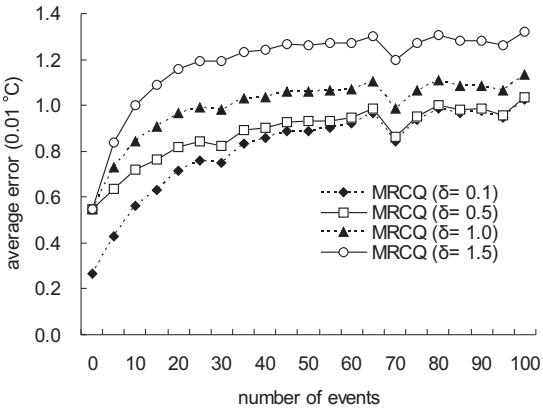


(b) event number: 80

Fig. 14: Effect of compression ratio γ on the spatial compression algorithm.



(a) amount of data transmission



(b) average errors

Fig. 15: Effect of the temporal compression algorithm.

data transmission according to the environmental stability and preserve important characteristics of sensing reports.

REFERENCES

[1] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Comm. of the ACM*, vol. 47, no. 6, pp. 34–40, 2004.

[2] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys '04)*, pp. 214–226, 2004.

[3] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "SensorScope: out-of-the-box environmental monitoring," *Proc. IEEE Int'l Conf. Information Processing in Sensor Networks (IPSN '08)*, pp. 332–343, 2008.

[4] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Comm. Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[5] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen, "The gator tech smart house: a programmable pervasive space," *IEEE Computer*, vol. 38, no. 3, pp. 50–60, 2005.

[6] C.C.Y. Poon, Y.T. Zhang, and S.D. Bao, "A novel biometrics method to secure wireless body area sensor networks for telemedicine and m-health," *IEEE Comm. Magazine*, vol. 44, no. 4, pp. 73–81, 2006.

[7] Y.C. Tseng, Y.C. Wang, K.Y. Cheng, and Y.Y. Hsieh, "iMouse: an integrated mobile surveillance and wireless sensor system," *IEEE Computer*, vol. 40, no. 6, pp. 60–66, 2007.

[8] C.T. Ee and R. Bajcsy, "Congestion control and fairness for many-to-one routing in sensor networks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys '04)*, pp. 148–161, 2004.

[9] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann, "An evaluation of multi-resolution storage for sensor networks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys '03)*, pp. 89–102, 2003.

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

[11] D. Ganesan, B. Greenstein, D. Estrin, J. Heidemann, and R. Govindan, "Multiresolution storage and search in sensor networks," *ACM Trans. Storage*, vol. 1, no. 3, pp. 277–315, 2005.

[12] T.A. Welch, "A technique for high-performance data compression," *IEEE Computer*, vol. 17, no. 6, pp. 8–19, 1984.

[13] C.M. Sadler and M. Martonosi, "Data compression algorithms for energy-constrained devices in delay tolerant networks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys '06)*, pp. 265–278, 2006.

[14] D.S. Taubman and M.W. Marcellin, *JPEG2000: fundamentals, standards and practice*, Kluwer Academic Publishers, 2002.

[15] R.M. Rao and A.S. Bopardikar, *Wavelet transforms: introduction to theory and applications*, Addison Wesley Publications, 1998.

[16] G. Davis, "Wavelet image compression construction kit," http://www.geoffdavis.net/dartmouth/wavelet/wavelet.html.

[17] D. Slepian and J.K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Information Theory*, vol. 19, no. 4, pp. 471–480, 1973.

[18] A.H. Kaspi and T. Berger, "Rate-distortion for correlated sources with partially separated encoders," *IEEE Trans. Information Theory*, vol. 28, no. 6, pp. 828–840, 1982.

[19] R. Cristescu, B. Beferull-Lozano, and M. Vetterli, "Networked Slepian-Wolf: theory, algorithms, and scaling laws," *IEEE Trans. Information Theory*, vol. 51, no. 12, pp. 4057–4073, 2005.

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

[21] D. Donoho, "Compressed sensing," *IEEE Trans. Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[22] J. Haupt, W.U. Bajwa, M. Rabbat, and R. Nowak, "Compressed sensing for networked data," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 92–101, 2008.

[23] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, "TAG: a tiny aggregation service for ad-hoc sensor networks," *ACM SIGOPS Operating Systems Review*, vol. 36, pp. 131–146, 2002.

[24] S. Lindsey, C. Raghavendra, and K.M. Sivalingam, "Data gathering algorithms in sensor networks using energy metrics," *IEEE Trans. Parallel and Distributed Systems*, vol. 13, no. 9, pp. 924–935, 2002.
[25] R. Kumar, M. Wolenetz, B. Agarwalla, J. Shin, P. Hutto, A. Paul, and U. Ramachandran, "DFuse: a framework for distributed data fusion," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys '03)*, pp. 114–125, 2003.
[26] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Networking*, vol. 11, no. 1, pp. 2–16, 2003.
[27] K.W. Fan, S. Liu, and P. Sinha, "Structure-free data aggregation in sensor networks," *IEEE Trans. Mobile Computing*, vol. 6, no. 8, pp. 929–942, 2007.
[28] N. Ahmed, T. Natarajan, and K.R. Rao, "Discrete cosine transfom," *IEEE Trans. Computers*, vol. 1, no. C-23, pp. 90–93, 1974.
[29] Crossbow, "MOTE-KIT2400 - MICAz Developer's Kit," http://www.xbow.com.