Data Compression by Temporal and Spatial Correlations in a Body-Area Sensor Network: A Case Study in Pilates Motion Recognition

Chun-Hao Wu and Yu-Chee Tseng, *Senior Member*, *IEEE*

Abstract—We consider a *body-area sensor network* (BSN) consisting of multiple small, wearable sensor nodes deployed on a human body to track body motions. Concerning that human bodies are relatively small and wireless packets are subject to more serious contention and collision, this paper addresses the data compression problem in a BSN. We observe that, when body parts move, although sensor nodes in vicinity may compete strongly with each other, the transmitted data usually exist some levels of redundancy and even strong temporal and spatial correlations. Unlike traditional data compression approaches for large-scale and multihop sensor networks, our scheme is specifically designed for BSNs, where nodes are likely fully connected and overhearing among sensor nodes is possible. In our scheme, an offline phase is conducted in advance to learn the temporal and spatial correlations of sensing data. Then, a partial ordering of sensor nodes is determined to represent their transmission priorities so as to facilitate data compression during the online phase. We present algorithms to determine such partial ordering and discuss the design of the underlying MAC protocol to support our compression model. An experimental case study in Pilates exercises for patient rehabilitation is reported. The results show that our schemes reduce more than 70 percent of overall transmitted data compared with previous approaches.

Index Terms—Body-area sensor network, data compression, inertial sensor, pervasive computing, wireless sensor network.

1 Introduction

The advance of Microelectro Mechanical Systems (MEMS) and wireless technology has boosted *body-area sensor networks (BSNs)*, in which sensor nodes are deployed on a human body to monitor various physical and biological signs, such as motions and heartbeat rates. Its applications include sports training [1], medical care [2], [3], pervasive games [4], [5], affective computing [4], and human-computer interface [6].

This paper considers applying BSNs to physical rehabilitation, where patients have to frequently practice certain fixed-pattern exercises repeatedly [3]. Fig. 1 shows a scenario, where a BSN is deployed on a patient to capture her body motions. Via wireless links, the sink is able to collect sensing data, the data analyzer is able to analyze and visualize her motions, and the application system is able to give comments and send feedbacks to therapists for further diagnosis. Such applications may require real-time and high-resolution sensing data for various purposes, such as visualizing motions, recognizing body conditions, and diagnosing diseases. These requirements imply shorter transmission delays and higher sampling rates for the BSN. In addition, due to the size of a human body, nodes are typically dense and almost fully connected, leading to severe contention and collision among their transmissions [7], [8]. Since wireless bandwidths are limited, how to efficiently utilize them is very important.

• *The authors are with the Department of Computer Science, National Chiao-Tung University, 1001 University Road, Hsin-Chu 30010, Taiwan. E-mail: {wchunhao, yctseng}@cs.nctu.edu.tw.*

Manuscript received 11 Mar. 2010; revised 7 Oct. 2010; accepted 19 Oct. 2010; published online 17 Dec. 2010.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2010-03-0116. Digital Object Identifier no. 10.1109/TMC.2010.264.

We attack these issues by exploiting data compression by taking advantage of the special features of sensing data when sensors are deployed on a human body and the possibility that sensors can overhear each others' transmissions in a BSN. We observe that when human body parts move, it usually exhibits strong temporal and spatial correlations among sensing data. By *temporal correlation*, body signs sensed by a single node typically change smoothly and slowly. By *spatial correlation*, body signs measured on different nodes are typically correlated because body components are connected and they normally move with certain rhymes. On the other hand, since nodes in a BSN are typically fully connected, spatial correlations can be efficiently exploited through overhearing, a nature of wireless communication, among nodes to minimize the total amount of transmissions. Although data compression schemes that exploit temporal and spatial correlations in wireless sensor networks (WSNs) have been widely discussed [9], [10], such schemes are usually designed for large-scale networks with multihop communications and are thus not suitable for BSNs, which are likely fully connected, providing a room for customized data compression of body motion data.

Since wearable devices are typically lightweight and have limited computation power and resources, the corresponding sensing data processing, compression, and MAC protocol should be lightweight, too. In particular, the following guidelines are important in designing data compression schemes for BSNs. First, the general form to model the temporal-spatial correlations of human body parts should not be too complicated, especially when considering the correlation among multiple sensors on different nodes. Second, we have to take the data gathering process and the distributed nature of a BSN into consideration. For example,

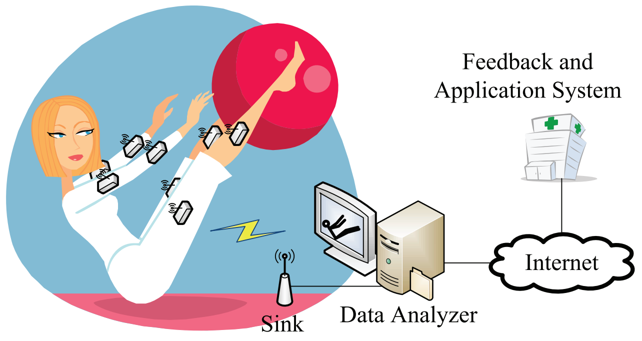


Fig. 1. A BSN architecture for motion recognition.

the rich connectivity of a BSN normally incurs higher contention and collision among transmissions.

In this paper, we propose a novel data compression method for BSNs based on overhearing. We exploit temporal and spatial correlations among sensing data to facilitate data compression and transmissions. Each node samples its data, overhears others' transmissions, compresses its data, and then transmits. Temporal and spatial correlations are modeled by differential coding and linear regression, respectively. Note that our data compression is lossless. In our method, an offline phase is conducted to learn these correlations and various system parameters. Then, nodes are sorted into a partial-ordering tree, which specifies the preferred transmission order of nodes (level-1 nodes transmit first, followed by level-2 nodes, etc.). During the online phase, each node overhears others' transmissions, decodes some of them, encodes its own sample according to the learned correlations, and transmits the encoded result. It is to be noted that under our modeling, the compression ratio is indeed affected by nodes' transmission order. Our scheme can also tolerate that the actual transmission deviates from the ideally planned order, at the cost of a higher compression ratio. We show how to construct an optimal partial-ordering tree, called *Minimum-Cost Tree (MT)*, such that the total data transmitted on the air are minimal statistically. The MT scheme, however, may need to decode and encode a long sequence of overheard data. An alternative is a near-optimal *Minimum-Cost, Depth-Bounded Tree (MDT)*, which limits the number of transmissions to be overheard by each node. The MDT scheme is more practical, but unfortunately, it has been proven that optimizing tree cost given any bound on its depth is NP-hard.

To summarize, major contributions of this paper are threefold. First, a distributed data compression method suitable for BSNs is proposed to exploit temporal and spatial correlations of sensing data. Our method considers the correlation among multiple sensors on different nodes. The computation complexity and communication overhead are kept as low as possible. Second and more importantly, we utilize overhearing, a nature of wireless communication, and investigate the prediction directions in spatial correlations. We show how to construct an optimal partial-ordering tree, based on which nodes can sent their transmission sequences and overhear others' transmissions in a cooperative way so as to minimize the total amount of transmissions. To the best of our knowledge, applying overhearing in data compression is never explored before. We also propose a more practical minimum-cost, depth-bounded tree, which turns out to be NP-hard. These techniques may be applied to other distributed source coding, too. Third, to verify our results, an experimental case study in Pilates exercises for patient rehabilitation is reported.

The rest of paper is organized as follows: Section 2 reviews some related works. Section 3 describes our BSN system architecture. Section 4 gives our data compression models and algorithms, including some notes on practical issues. Experimental results and our prototypes are shown in Section 5. Section 6 concludes this paper.

2 Related Works

BSN is an emerging technique. Its bandwidth utilization issues have started to receive attention. Assuming a fully connected BSN, a QoS framework similar to IEEE 802.16 is proposed for BSNs in [11] to grant connections and allocate bandwidths based on applications' requests. Several lossy data compression methods have been proposed. In [12], sensors with higher priorities are allowed to transmit their sensing data with higher precision. If the data sources are sparse, e.g., in the case of PPG signals, compressed sensing techniques can be applied to reduce the required sampling rates [13]. For motion capturing, Cheng et al. [14] have tested the performance of several existing lossless and lossy compression algorithms on motions that exhibit repetitive patterns, such as running. However, the compression is performed by each individual sensor and does not consider the correlations among sensors. We extend their work to consider correlations among sensors (both on the same node and on different nodes). Although our data compression model is lossless, any lossy transform (e.g., FFT and wavelet transform mentioned in [14]) can be applied to the data source before passing through our data compression model. If the application is interested only in certain events, rather than the sensing values, distributed motion classification/annotation would be most efficient [15], [16].

On the other hand, for large-scale WSNs, data compression has been intensively studied. We classify these solutions as follows:

- **Cross-layer solutions.** The impact of joint routing and compression is studied in [17]. ElBatt [18] studies the feasibility of sublinear scaling laws under different cooperation schemes for spatially correlated data.
- **Data aggregation.** Data compression may be performed by relay nodes or cluster heads via data aggregation. TAG [19] organizes a network into a tree and proposes SQL-like semantics to aggregate streaming data into histograms for low-power aggregation. Multiresolution spatial and temporal coding is addressed in [20].
- **Data acquisition.** As acquiring sensing data from a large-scale WSN to a user's hand-held device may involve long delays, nonuniform energy consumptions, and network impairments, model-driven data acquisition models are proposed in [9], [10], and [21]. Although these methods are suitable for infrequent queries and slow-changing environments, they are

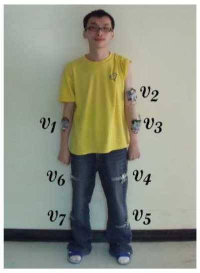


Fig. 2. An example of placing sensor nodes on a human body.

not practical for motion capturing, as motion data usually change more rapidly.

3 BSN System Architecture

We consider a BSN deployed on a human body for motion recognition. An example is the Pilates exercises for patient rehabilitation. The system architecture is shown in Fig. 1. Wearable sensor nodes are placed at main movable body parts. These nodes will periodically report their readings to a sink node, which is connected to the data analyzer. The data analyzer will try to understand how well the person conducts some motions. The application server can further send feedbacks to the person and reconstruct, store, and replay these motions.

As an example, to recognize Pilates exercises for lower back pain relief [22], [23], nodes can be placed at main body parts as shown in Fig. 2. Some Pilates exercises are shown in Fig. 3. In the upper leg lifting exercise (b), one has to lie on his side and put legs and feet together in parallel. Then, lift the top leg up from 0 to about 45 degrees, stay for about 5 seconds, and then return to the start. The exercise needs to be repeated several times.

Due to the size of a human body, nodes in a BSN are typically fully connected. Therefore, when a node is transmitting, the other nodes can overhear its transmission. In addition, their data could be highly correlated. By utilizing the overheard information, one may conduct compression based on sensor data's spatial and temporal correlations. We show that by properly arranging nodes' transmission order, not only the competition and interference among nodes can be relieved, but also significant compression can be done. In this work, we place a triaxial accelerometer in each sensor node. The sampling rate on each axis is set to 20 Hz, a common value for body motion measurement [24]. (However, the applicability of our work is not limited to these assumptions.) Sensor readings are calibrated and compressed locally, but the computation should be kept as simple as possible.

4 Design of Data Compression by Overhearing

We first present our basic idea in Section 4.1. Then, we present our data compression model and two algorithms to sort nodes' overhearing sequences in Sections 4.2 and 4.3, respectively. Although our model does not require to change the underlying MAC protocol, we discuss the corresponding design issues in Section 4.4. Finally, in Section 4.5, we discuss how to retrain the system where

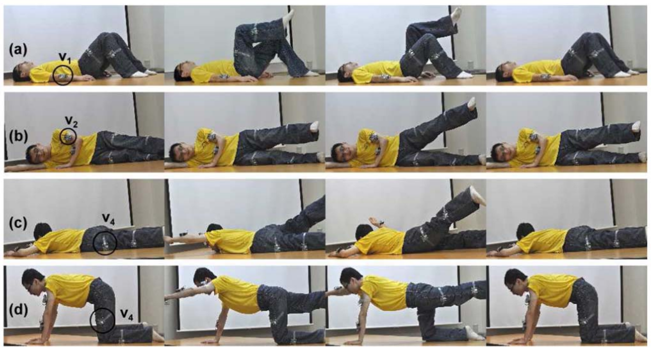


Fig. 3. Examples of Pilates exercises.

there are deviations between the training parameters and the actual behaviors.

4.1 Basic Idea

We consider a BSN with n nodes $v_1, v_2, \ldots, v_n$, each equipped with m sensors. Note that in the case of triaxial accelerometer, each axis is regarded as one separate sensor. We will design a compression scheme to exploit the *temporal* and *spatial correlations* of sensor readings. The goal is to automatically learn these correlations to facilitate compression. Our system has an *offline phase* and an *online phase*. During the offline phase, we will collect the raw outputs of the jth sensor of v_i into a column vector $R_i^{(j)}$, where the tth sampling result is stored at the tth entry in the vector and is written as $R_i^{(j)}[t]$. Note that since the same Pilates exercise should be repeated multiple times by the same person to make the training results more representative, $R_i^{(j)}$ should contain the result of several repetitions of the same Pilates exercise. Then, $R_i^{(j)}$ is calibrated into a column vector $X_i^{(j)}$ of the same size carrying meaningful data. Note that we do not specify the size of vector $R_i^{(j)}$. A larger vector means more statistics, leading to higher accuracy in our prediction of correlations.

Temporal correlation hereby means the similarity of sensing values over the time axis, i.e., $\Delta X_i^{(j)}[t] = X_i^{(j)}[t] - X_i^{(j)}[t-1]$ is very likely to fall within a small range relative to that of the original value $X_i^{(j)}[t]$. Previous researches [14], [24] also show that body movements are normally centered at very slow frequencies and do not change rapidly in nearby samples.

Spatial correlation hereby means that human motions in nearby body parts show inherent rhythm, making it a good source for data compression. Specifically, one may easily predict $\Delta X_i^{(1)}, \Delta X_i^{(2)}, \ldots, \Delta X_i^{(m)}$ given $\Delta X_k^{(1)}, \Delta X_k^{(2)}, \ldots, \Delta X_k^{(m)}$, if nodes v_i and v_k are attached to correlated body parts. To achieve low-complexity compression, we would investigate the possibility of establishing the following linear correlation:

$$\begin{aligned} \Delta X_i^{(j)} &= \alpha_{(i,j)|k}\mathbf{1} + \beta_{(i,j)|(k,1)}\Delta X_k^{(1)} \\ &\quad + \beta_{(i,j)|(k,2)}\Delta X_k^{(2)} + \cdots + \beta_{(i,j)|(k,m)}\Delta X_k^{(m)} \\ &\quad + \epsilon_{(i,j)|k}, \end{aligned} \tag{1}$$

where $\alpha_{(i,j)|k}$ and $\beta_{(i,j)|(k,1)}, \ldots, \beta_{(i,j)|(k,m)}$ are scalars, $\mathbf{1}$ is a column vector containing all 1s, and $\epsilon_{(i,j)|k}$ is a column vector to compensate the prediction errors. If the outputs of some sensor of v_k have little correlation with $\Delta X_i^{(j)}$, its corresponding β should approach zero. With properly selected coefficients α and β, vector $\epsilon_{(i,j)|k}$ would contain all very small values, making it much less costly to represent $\epsilon_{(i,j)|k}$ than $\Delta X_i^{(j)}$. The number of bits required to represent $\epsilon_{(i,1)|k}, \epsilon_{(i,2)|k}, \ldots, \epsilon_{(i,m)|k}$ is used as the cost to compress sensing data of v_i when the data of v_k are known. Intuitively, we try to recover v_i's data from v_k's data. In subsequent sections, we will show how to find the optimal predictor v_k of each node v_i (by building a partial-ordering tree) so as to minimize the overall cost.

On the other hand, sensors of the same node v_i (such as $\Delta X_i^{(j)}$ and $\Delta X_i^{(k)}$) may also exhibit strong spatial correlations, especially among the three axes of a triaxial accelerometer. We call such spatial correlations *intranode spatial correlations*, and those among sensors of different nodes *internode spatial correlations*. So, (1) is generalized as follows to include intranode spatial correlations:

$$\begin{aligned}\Delta X_i^{(j)} = \alpha_{(i,j)|k}\mathbf{1} + \sum_{p=1}^{m} \beta_{(i,j)|(k,p)} \Delta X_k^{(p)} \\ + \sum_{q \in L_{(i,j)|k}} \beta_{(i,j)|(i,q)} \Delta X_i^{(q)} + \epsilon_{(i,j)|k},\end{aligned} \tag{2}$$

where we use all sensors of v_k and a set $L_{(i,j)|k}$ of sensors of v_i to predict $\Delta X_i^{(j)}$. Note that to conduct data compression, we must avoid circular dependency among sensors of the same v_i. $L_{(i,j)|k}$ is to specify the set of sensors of v_i whose sensing data are encoded before that of sensor j. Later on, we will show how to determine the set $L_{(i,j)|k}$. By learning the intranode spatial correlations in (2), the resulting error vector $\epsilon_{(i,j)|k}$ of (2) is expected to be even smaller than that of (1), making compressing the former more efficient than compressing the latter.

Our experiences show that the above modeling is quite effective in recognizing Pilates exercises. Consider the sensor deployment in Fig. 2 and the exercise in Fig. 3a and assume that each node has only one x-axis accelerometer. Fig. 4a shows the calibrated data $X_4^{(1)}, \ldots, X_7^{(1)}$. We see that $X_4^{(1)}$ and $X_5^{(1)}$ exhibit strong spatial correlations, and so are $X_6^{(1)}$ and $X_7^{(1)}$. By taking the difference operation on $X_i^{(1)}$, Fig. 4b shows that $\Delta X_i^{(1)}$ has a much smaller range, and the aforementioned spatial correlations still exist. So, we can apply (2) to predict $\Delta X_4^{(1)}$ by $\Delta X_5^{(1)}$, as shown in Fig. 4c. Fig. 4d shows the corresponding probability distribution of the contents of the error vector $\epsilon_{(4,1)|5}$. As can be seen, the distribution concentrates around 0 mg.

By overhearing among sensor nodes, we will investigate the feasibility of learning these correlations in the offline phase and exploring real-time data compression in a BSN during the online phase. We will also address the issue of the supporting MAC protocol, especially the design of packet collision, in Section 4.4.

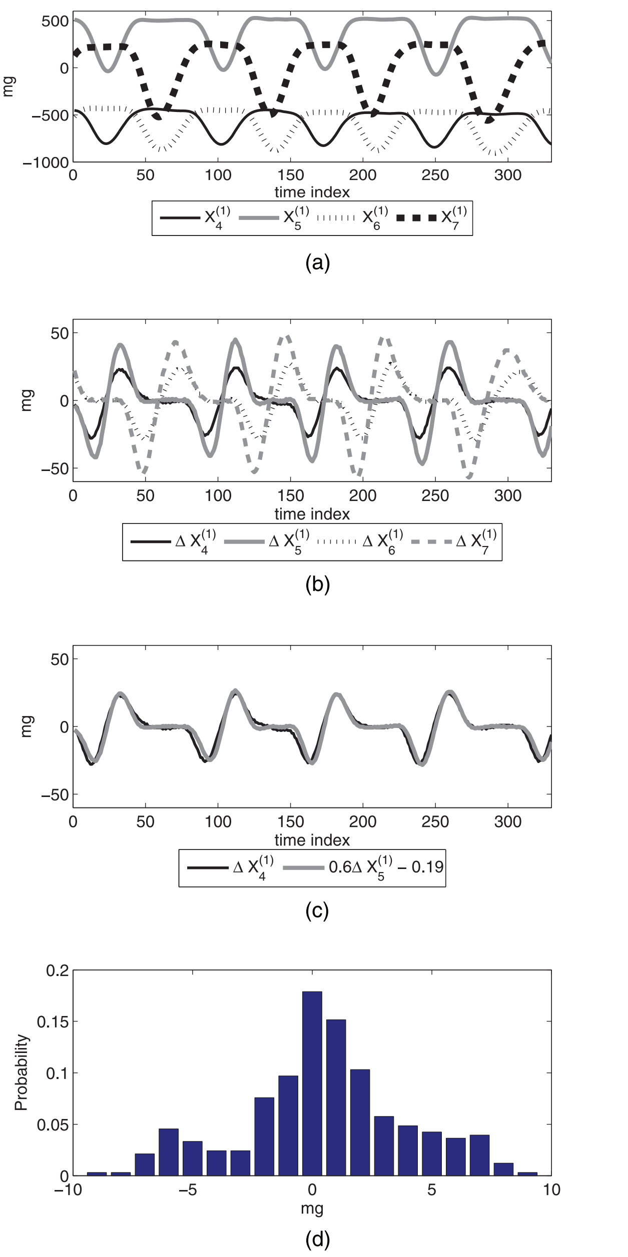


Fig. 4. Validation of our temporal and spatial correlation modeling. (a) Calibrated data $X_i^{(1)}$ of exercise Fig. 3a. (b) The difference vector $\Delta X_i^{(1)}$. (c) Predicting $\Delta X_4^{(1)}$ by $\Delta X_5^{(1)}$. (d) Probability distribution of the prediction error vector $\epsilon_{(4,1)|5}$.

4.2 Data Compression Model

Data flows of our offline and online phases are shown in Fig. 5. Temporal correlation will be exploited on individual sensors by a simple differential coding, while spatial correlation will be exploited across sensors and will be learned from the offline phase. The learned results consist of some coefficients and some transmission and encoding orders, which are sent to the sensor nodes for them to conduct online data compression. The *transmission order*, which is represented by a spanning tree, specifies the partial ordering of nodes' transmissions and the target nodes that a node should overhear. To simplify the notations, we will abbreviate $R_i^{(1)}, \ldots, R_i^{(m)}$ by $R_i^{(1:m)}, \Delta X_i^{(1)}, \ldots, \Delta X_i^{(m)}$ by

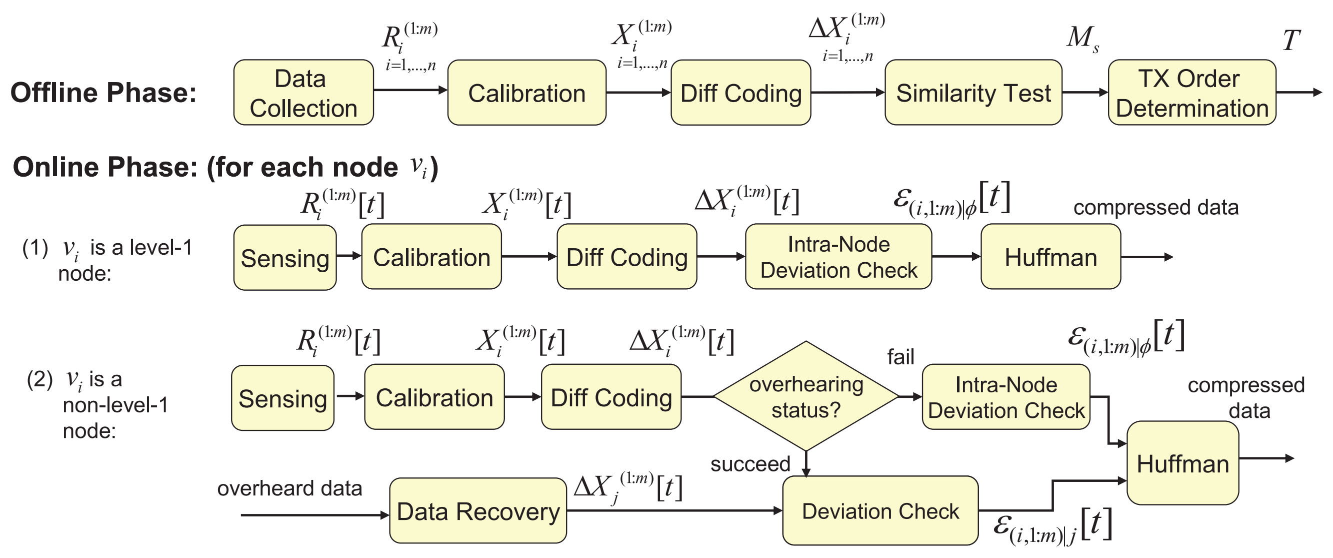


Fig. 5. Offline and online phases of our data compression model.

$\Delta X_i^{(1:m)}, \beta_{(j,q)|(i,1)}, \ldots, \beta_{(j,q)|(i,m)}$ by $\beta_{(j,q)|(i,1:m)}, \epsilon_{(i,1)|j}, \ldots, \epsilon_{(i,m)|j}$ by $\epsilon_{(i,1:m)|j}, \beta_{(j,q)|(j,l)}$ for all $l \in L_{(j,q)|i}$ by $\beta_{(j,q)|(j,L_{(j,q)|i})}$, and so on.

Offline phase. In the offline phase, for each type of motion, training data are collected from the BSN in the "Data Collection" block and then sent to an external data analyzer, where the rest of the blocks will be executed.

Data collection. We will collect from nodes $v_1, v_2, \ldots, v_n$ column vectors $R_1^{(1:m)}, R_2^{(1:m)}, \ldots, R_n^{(1:m)}$. In the collection process, each node v_i puts the tth sampling result of its qth sensor into $R_i^{(q)}[t]$. $R_i^{(1:m)}$ should be stored in v_i and then forwarded to the sink in a reliable way to avoid the impact of network impairments. The same Pilates exercise should be repeated multiple times by the same person to make the training results more representative, i.e., $R_i^{(j)}$ should contain the results of several repetitions of the same Pilates exercise. To make the learned results effective, it is assumed that the same Pilates exercise is performed by the same person in the online phase.

Calibration. Each $R_i^{(j)}, i = 1 \ldots n, j = 1 \ldots m$, is converted into a meaningful vector $X_i^{(j)}$. For example, the raw outputs of an accelerometer may be converted from voltages into physical units (e.g., Guass). Then, we may apply various data filtering techniques to refine the data. For human motion, we propose to use a low-pass filter to remove noise. We adopt the FIR filter based on Kaiser window function [25]. A low-pass filter will zero out the signal components whose frequencies are higher than a predefined stopband frequency. As the energy of the filtered signal components becomes larger, the resulting signal waveform will become smoother. To automatically determine the stopband, we apply a fast Fourier transform to $R_i^{(j)}$ such that the removed energy is equal to γ percent of the total energy. We will test different values of γ in Section 5. The filtered data are rounded to the nearest integers and recorded in $X_i^{(j)}$.

Diff coding. To exploit temporal correlation, each $X_i^{(j)}$ is converted into a differential column vector $\Delta X_i^{(j)}$ such that $\Delta X_i^{(j)}[t] = X_i^{(j)}[t] - X_i^{(j)}[t-1]$.

Similarity test. Our goal is to identify the correlation between the outputs of each pair of v_i and v_j. Specifically, an $n \times n$ similarity matrix M_s will be constructed. $M_s[i][j]$ is to measure how well we can predict the data of v_j given the data of v_i. For ease of explanation, for now we assume that $L_{(j,1:m)|i}$ is given (we will explain how to find it later). To compute $M_s[i][j]$, we apply (2) and try to build the following equality for each $q = 1, 2, \ldots, m$, where $t = 1, 2, \ldots$:

$$\Delta X_j^{(q)}[t] = \alpha_{(j,q)|i} + \sum_{p=1}^{m} \beta_{(j,q)|(i,p)} \Delta X_i^{(p)}[t] + \sum_{l \in L_{(j,q)|i}} \beta_{(j,q)|(j,l)} \Delta X_j^{(l)}[t].$$

Then, we can compute $\alpha_{(j,q)|i}$, $\beta_{(j,q)|(i,1:m)}$, and $\beta_{(j,q)|(j,L_{(j,q)|i})}$ for each q by linear regression analysis [26]. Note that linear regression only tries to use the coefficients on the right-hand side to approximate $\Delta X_j^{(q)}[t]$, so we let the difference

$$\epsilon_{(j,q)|i}[t] = \Delta X_j^{(q)}[t] - \alpha_{(j,q)|i} - \sum_{p=1}^{m} \beta_{(j,q)|(i,p)} \Delta X_i^{(p)}[t] - \sum_{l \in L_{(j,q)|i}} \beta_{(j,q)|(j,l)} \Delta X_j^{(l)}[t].$$

Intuitively, if $\Delta X_i^{(1:m)}$ is obtained by overhearing, we only need to transmit the difference vector $\epsilon_{(j,q)|i}$ to obtain $\Delta X_j^{(q)}$. Here, we use the notation $\epsilon_{(j,q)|\phi}$ to represent that v_i's transmission is not based on any overhearing. Let the offline and online distributions of $\epsilon_{(j,q)|i}$ be $f_{(j,q)|i}$ and $g_{(j,q)|i}$, respectively. Assuming that the Huffman encoding [27] is applied, we will use the distribution of $\epsilon_{(j,q)|i}$ measured during the offline phase to transmit the data measured during the online phase. So, we let $M_s[i][j]$ be the total of entropies, $\sum_{q=1}^{m} H(f_{(j,q)|i})$. To compute this term, we also need to determine the encoding sequence of the m sensors of v_j, i.e., $L_{(j,q)|i}, q = 1 \ldots m$. We develop a greedy heuristic as follows: we work in the backward direction starting from the last sensor. Specifically, for $q = 1 \ldots m$, we compute $H(f_{(j,q)|i})$, assuming that $L_{(j,q)|i}$ contains all other sensors $\neq q$. Then, the sensor q with the minimum entropy is selected as the last one in the sequence. We then exclude this sensor and repeat the same process for the rest of the $m-1$ sensors. After determining the encoding sequence, $M_s[i][j]$ can be obtained.

Note that when $i=j$, $M_s[i][i]$ is a special case, which means that v_i is the first node to transmit and cannot overhear any information. In this case, we will ignore the factor of internode correlation in (2) and only consider intranode correlation. So, we reduce (2) to the following:

$$\Delta X_i^{(q)} = \alpha_{(i,q)}\mathbf{1} + \sum_{l\in L_{(i,q)}} \beta_{(i,q)|(i,l)} \Delta X_i^{(l)} + \epsilon_{(i,q)|\phi}. \tag{3}$$

With a similar greedy process, we can compute $M_s[i][i]$. So, the whole matrix M_s is obtained.

TX order determination. Based on M_s, the main goal of this block is to find a proper order of nodes' transmission time based on the correlations of their data to achieve the best compression ratio. The ordering instructs a node v_i to sequentially overhear the transmissions of some earlier nodes, so as to compute v_i's error vector $\epsilon_{(i,1:m)|j}$ with respect to v_j's data. In the offline phase, assuming ideal error-free transmissions, we model the ordering problem as one of finding a spanning tree T along a directed weighted complete graph $G=(\{u\}\cup V, E)$, where u is a virtual node, $V=\{v_1, v_2, \ldots, v_n\}$, and E contains all possible edges between nodes. The directed edge $\langle v_i, v_j\rangle \in E$ means "predicting $\Delta X_j^{(1:m)}$ from $\Delta X_i^{(1:m)}$" and the directed edge $\langle u, v_i\rangle \in E$ means "sending $\Delta X_i^{(1:m)}$ to the sink directly without overhearing." So, the weight of $\langle v_i, v_j\rangle$ is $w(\langle v_i, v_j\rangle) = M_s[i][j]$ and the weight of $\langle u, v_i\rangle$ is $w(\langle u, v_i\rangle) = M_s[i][i]$. Since virtual node u will not overhear any node, we let $w(\langle v_i, u\rangle)=\infty$ for all v_i. Note that T is an outgoing directed tree and is always rooted at u. In Section 4.3, we propose two solutions to construct T.

Online phase. The online phase is performed by each node in a distributed manner (refer to Fig. 5). Consider the transmission of the tth sampling result of v_i. There are two cases: v_i is a level-1 node and v_i is a non-level-1 node. In both cases, it will go through the "Sensing" block to collect the raw data and the "Calibration" and the "Diff Coding" blocks to obtain $\Delta X_i^{(1:m)}[t]$. If v_i is a level-1 node in T, $\Delta X_i^{(1:m)}[t]$ will go through the "Intranode Deviation Check" block and the resulting $\epsilon_{(i,1:m)|\phi}[t]$ will be compressed by the "Huffman" block and then transmitted without overhearing. If v_i is a non-level-1 node, then it has to overhear all precedent nodes along the path from the root u to itself in T. If all these data are overheard successfully, the sequence of blocks "Data Recovery," "Deviation Check," and "Huffman" will be activated, as shown in the lower path of Fig. 5. However, it is possible that v_i may miss some expected overhearing targets for some reasons. In this case, v_i should transmit without going through overhearing; it activates the "Intranode Deviation Check" block and the "Huffman" block, as shown in the upper path of Fig. 5.

Intranode deviation check. In the above discussion, v_i has to transmit without overhearing. From $\Delta X_i^{(1:m)}[t]$, v_i follows (3) to calculate

$$\epsilon_{(i,q)|\phi}[t] = \Delta X_i^{(q)}[t] - \alpha_{(i,q)} - \sum_{l\in L_{(i,q)}} \beta_{(i,q)|(i,l)} \Delta X_i^{(l)}[t]$$

for $q=1,\ldots,m$ to obtain $\epsilon_{(i,1:m)|\phi}[t]$.

Data recovery. Let $u \rightarrow v_{j_1} \rightarrow v_{j_2} \rightarrow \cdots \rightarrow v_{j_p} \rightarrow v_i$ be the path in tree T from the root u to v_i. Node v_i has to overhear the transmissions by $v_{j_1}, v_{j_2}, \ldots, v_{j_p}$ and recover the original $\epsilon_{(j_1,1:m)|\phi}[t]$, $\epsilon_{(j_2,1:m)|j_1}[t]$, $\epsilon_{(j_3,1:m)|j_2}[t], \ldots, \epsilon_{(j_p,1:m)|j_{p-1}}[t]$ through Huffman decoding. If all the necessary data are overheard successfully, the lower path of Fig. 5 is executed. Otherwise, it follows the upper path of Fig. 5 (i.e., this block and the subsequent "Deviation Check" block are ignored). In the former case, v_i first applies

$$\Delta X_{j_1}^{(q)} = \alpha_{(j_1,q)}\mathbf{1} + \sum_{l\in L_{(j_1,q)}} \beta_{(j_1,q)|(j_1,l)} \Delta X_{j_1}^{(l)} + \epsilon_{(j_1,q)|\phi}$$

for $q=1,\ldots,m$ to obtain $\Delta X_{j_1}^{(1:m)}$. Then, it follows the encoding sequence and repeatedly applies

$$\begin{aligned}\Delta X_{j_{k+1}}^{(q)}[t] &= \alpha_{(j_{k+1},q)|j_k} + \sum_{r=1}^{m} \beta_{(j_{k+1},q)|(j_k,r)} \Delta X_{j_k}^{(r)}[t] \\ &\quad + \sum_{l\in L_{(j_{k+1},q)}} \beta_{(j_{k+1},q)|(j_{k+1},l)} \Delta X_{j_{k+1}}^{(l)}[t] + \epsilon_{(j_{k+1},q)|j_k}[t]\end{aligned}$$

for each of its own sensor q, and for $k=1,\ldots,p-1$, until $\Delta X_{j_p}^{(1:m)}[t]$ are obtained.

Deviation check. From $\Delta X_i^{(1:m)}[t]$ and $\Delta X_{j_p}^{(1:m)}[t]$ (obtained by the "Data Recovery" block), v_i calculates

$$\begin{aligned}\epsilon_{(i,q)|j_p}[t] &= \Delta X_i^{(q)}[t] - \alpha_{(i,q)|j_p} - \sum_{r=1}^{m} \beta_{(i,q)|(j_p,r)} \Delta X_{j_p}^{(r)}[t] \\ &\quad - \sum_{l\in L_{(i,q)|j_p}} \beta_{(i,q)|(i,l)} \Delta X_i^{(l)}[t]\end{aligned}$$

for $q=1,\ldots,m$ to obtain $\epsilon_{(i,1:m)|j_p}[t]$.

Huffman. If the lower path of Fig. 5 is taken, v_i follows its encoding sequence to encode $\epsilon_{i^{(q)}|j_p}[t]$ for each $q=1,\ldots,m$ by Huffman encoding, puts the encoding results together, and transmits the compressed data. Note that the average compressed data size should be pretty close to the total entropies of $\epsilon_{(i,1:m)|j_p}$ computed in the offline phase if the exercises are conducted similarly to the offline phase. If the upper path of Fig. 5 is taken, v_i also encodes $\Delta X_i^{(q)}[t]$ for each $q=1,\ldots,m$ by Huffman encoding, puts the encoding results together, and transmits the compressed data.

4.3 Algorithms for Constructing the TX Order Tree T

Recall that in the "TX Order Determination" block in the offline phase, we are given a directed weighted complete graph G, in which each edge represents the corresponding message volume when a node's transmission is based on overhearing another node's transmission. Our goal is to form a TX order tree T from G for the best transmission efficiency. We will propose two solutions. The first one, called *Minimum-Cost Tree*, can achieve optimal compression ratio, but may require a node to overhear and decompress the transmissions of multiple nodes before it can transmit its own data. The second one, called *Minimum-Cost, Depth-Bounded Tree*, allows a node to bound the number of nodes that it has to overhear and is thus more practical for wireless environments with non-negligible transmission errors. Note that given a T and a path $u \rightarrow v_i \rightarrow v_j \rightarrow v_k \rightarrow \ldots$ in T, the underlying meanings are: "v_i transmits to the sink directly," "v_j overhears v_i's transmission, recovers v_i's original data, and compresses its own data," "v_k overhears v_i's transmission, recovers v_i's original data, overhears v_j's transmission, recovers v_j's original data, and compresses its own data," etc. For v_k, overhearing only v_j's transmission is not sufficient because v_j's data are compressed data and can be decompressed only if v_i's original data are known. This chain effect is not preferred, especially for BSNs.

The MT scheme works by finding an outgoing tree T rooted at u from G with the smallest total edge weight. We adopt an efficient polynomial-time algorithm [28], [29] to find T, which works similarly to Kruskal's algorithm for undirected graphs. The algorithm greedily selects incoming edges with minimum weights and breaks cycles, if any.

1. We will pick a set S of edges from G to form the edges of T. Edges entering u are first removed from G. For all incoming edges of each v_i, select the one with the smallest weight into S. Now, S contains n edges. For each $v_i \in V$, we denote by $\text{prev}(v_i)$ the (only) node such that $\text{prev}\langle (v_i), v_i\rangle \in S$.
2. If S contains no cycles, then $T = (\{u\} \cup V, S)$ is an optimal tree and the algorithm terminates. Note that T is connected because it is acyclic and every node except the root u has an incoming edge. Otherwise, continue to the following steps to break the cycles. (Note that there are many polynomial-time algorithms to identify cycles in a graph.)
3. Find any cycle C in S. For each edge $\langle v_i, v_j\rangle \in E$ such that $v_i \notin C$ and $v_j \in C$, we reweight it as follows:

   $$w'(\langle v_i, v_j\rangle) = w(\langle v_i, v_j\rangle) - w(\langle \text{prev}(v_j), v_j\rangle).$$

   Note that $w'(\langle v_i, v_j\rangle) \geq 0$ because $\langle \text{prev}(v_j), v_j\rangle$ has the smallest weight among all v_j's incoming edges in E (see step 1).
4. From all reweighted edges in step 3, let $\langle v_i, v_j\rangle$ be the one with the smallest weight $w'(\langle v_i, v_j\rangle)$. Break the cycle by deleting $\langle \text{prev}(v_j), v_j\rangle$ from S and adding $\langle v_i, v_j\rangle$ to S.
5. After breaking the cycle, edges in C do not need to be considered any more. So, we adjust G by contracting the nodes in C into a pseudonode v_k, by removing all nodes and edges in C from G and adding a v_k to G. For each $v_i \notin C$, we let weight $w(\langle v_k, v_i\rangle) = \min\{w(\langle v_j, v_i\rangle), v_j \in C\}$ and weight

   $$w(\langle v_i, v_k\rangle) = \min\{w'(\langle v_i, v_j\rangle), v_j \in C\},$$

   where w' means the new weight in step 3.
6. Go to step 2 with the contracted graph G.

It is proved in [28], [29] that the T found above is the minimum outgoing tree. For example, consider the network in Fig. 6a. The solid arrows are selected into S in step 2. There is one cycle containing nodes v_1, v_2, and v_3. The cycle is contracted into a pseudonode v_5 in Fig. 6b. We delete $\langle v_1, v_2\rangle$ from S and add $\langle u, v_2\rangle$ to S. Weights of edges to and from v_5 are recalculated. After one iteration, tree T is found.

The second MDT scheme enforces that the depth of T be bounded by a constant δ. For example, if we set $\delta = 2$, then a level-1 node can transmit anytime and a level-2 node only needs to overhear its previous node's transmission. Limiting T's depth would be more practical for BSNs. However, it turns out that optimizing this "bicriteria lowest-cost, bounded-depth" problem is NP-hard [30]. It has been proved that approximating this problem cannot be achieved within a log factor and an approximation algorithm is proposed in [30], which can bound the cost within a factor of $O(logn)$ of the optimal solution. Several heuristics for finding bounded-diameter minimum spanning trees in an undirected graph have been proposed (refer to [31] and the references therein). Although their goals are similar to our MDT problem in that bounding the depth of the tree to δ can be enforced by bounding its diameter to 2δ, they are only applicable to undirected graphs. So, we will adopt the solution in [30] in our simulation studies. It remains an open issue to find better heuristics for the MDT problem.

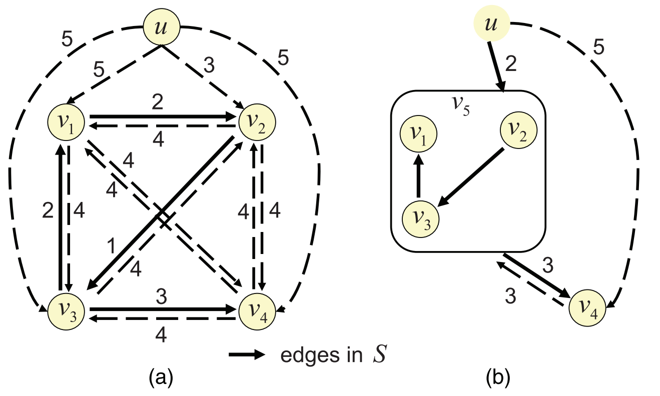


Fig. 6. An example of the MT method.

4.4 Design Issues of the Underlying MAC Protocol

The above overhearing and transmitting activities can rely on any general wireless MAC protocols to achieve our goal of data compression. The only requirement would be to support assigning priorities to nodes so as to utilize the partial-ordering tree T. For example, one may use a polling-based protocol, a prioritized CSMA, or even a polling-based protocol on top of an existing underlying MAC protocol. However, if modifying the underlying MAC is possible, better performance may be achieved. Below, we comment on the desired changes on existing MAC protocols to facilitate our overhearing behavior. Our discussions will consider both polling-based and CSMA-based protocols.

- *Backoff window.* Recall the partial-ordering tree T. Nodes should transmit their sensing data according to their levels in T. If a polling-based MAC is adopted, no change is needed; the sink node can simply poll nodes level by level. If a CSMA-based MAC is adopted, then we can assign nodes' backoff timers based on their levels. A lower level node should adopt a smaller backoff window, while a higher level one should adopt a larger backoff window. (This is similar to the design of IEEE 802.11e, which uses differential backoff windows.)
- *Packet loss behavior.* Next, we consider the packet loss issue. Packet loss may happen to the sink as well as a node which intends to overhear the packet. For loss at the sink node, if a polling-based MAC is used, it

can simply repoll the sender; if a CSMA-based MAC is used, the sender will automatically retransmit following the protocol's definition. For loss at a node intending to overhear the packet, no matter which kind of MAC is adopted, when the opportunity for it to transmit appears before it already overhears all needed packets that it intends to overhear, it runs into a dilemma of whether to transmit or not. However, a node missing the packet that it intends to overhear should not request for retransmission because this would further complicate the problem (e.g., Keally et al. [32] also suggest avoiding retransmitting lost packets because doing so may violate the delay requirement). There are two options. One way is to repeat the backoff process, expecting that all needed packets that it intends to overhear would arrive correctly later on. The other way is to still transmit but assume that it is a level-1 node (i.e., without compression by overhearing). We would recommend using the second option because the former approach may encounter a cascaded effect and even a dilemma where a packet may have been correctly received by the sink but simply missed by the node. Also note that our data collection is lossless. Missing intended overhearing packets at a node only increases the compression ratio, but the accuracy of collected data is unaffected.

- *Data aggregation.* In the above discussion, to keep our presentation simple, we have focused on each piece of data individually and discussed its compression. To improve bandwidth utilization, it is better to pack several (compressed) sensing data into one packet instead of sending multiple small packets. This would depend on the maximum payload size agreed on the underlying MAC protocol. Since our overhearing scheme can further reduce the size of transmitted data, any data aggregation scheme can directly benefit from our scheme. (Note that since we adopt Huffman coding, which is a prefix-free coding, no delimiter character needs to be inserted between two pieces of encoded data.)

4.5 Retraining the System

So far, we have assumed that the training data in the offline phases can precisely reflect the behaviors during the online phase. In practice, there may exist some degree of mismatch in the predicted correlations, thus causing inefficiency. Specifically, there could be two major sources of such mismatch. One is that the trained Huffman codes may not match the online probability distribution of the ϵ functions. The other is that the trained αs and βs may drift from the actual coefficients during the online phase. Fortunately, these problems are solvable because our data compression model is lossless, implying that the online user motions are always recoverable, as long as the communication channel remains working. So long as the user's motions are repetitive, retraining the system by the data analyzer is always possible.

When the learned probability distributions of the ϵ functions do not match with the online distributions, the amount of additional bits in the online transmission is known as the relative entropy [27]. We may use adaptive Huffman coding [33] to measure current distributions, adjust the Huffman trees accordingly, and forward the results to the sensor nodes.

Sometimes, the online spatial correlations (αs and βs) may not match with the learned results for many reasons. For example, the user may simply perform a different motion, or the system is worn by a different person. (Note that this is less serious in the case of Pilates exercises because they are a set of standard and repetitive motions.) To make our scheme adaptive, the data analyzer can keep a window of past sensing data in its database, periodically compute new αs and βs, and update them to sensor nodes whenever mismatch is detected.

5 Experiment Results: A Case Study in Pilates Exercises

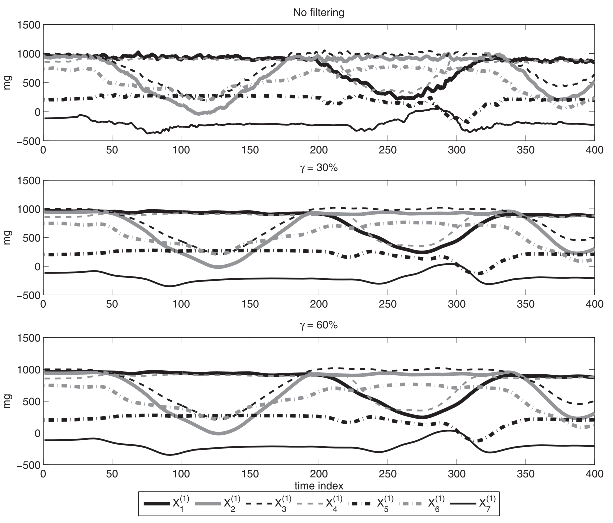
To verify the effectiveness of the proposed compression method, we have conducted some experiments on Pilates exercises. We collect sensing data, split them into a training data set and a test data set, build the offline data compression model by the training data set, and then analyze performance by the test data set.

5.1 Experiment Setup and Data Collection

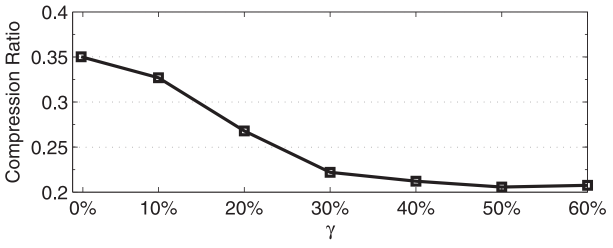
We conduct experiments on the four Pilates exercises as shown in Fig. 3. We place $n = 7$ nodes on appropriate body parts to monitor human motions (refer to Fig. 2). Each node is a Jennic JN5139 single-chip microprocessor [34] with a 16-MIPS RISC CPU, 192 KB ROM, 96 KB RAM, a ZigBee-compliant module, and an OS5000 triaxial accelerometer [35]. Note that we regard a triaxial accelerometer as $m = 3$ sensors. We index the x-, y-, and z-axes accelerometers as sensors 1, 2, and 3, respectively. Each piece of raw data in each axis is 16 bits, and the sampling rate is set to 20 Hz. To fairly compare different methods, we collect sensing data from the BSN, calibrate it on a PC, and also test different methods on the same set of calibrated data on the PC. Each exercise is repeated 20 times by the same person, and the sensing results, which consist of thousands of sensing values, are stored in nodes and then forwarded to the sink in a reliable way to avoid the impact of network impairments. Two-third of the calibrated sensing data are used as the training data set, and the rest one-third as the test data set. We use *compression ratio* (size of the compressed data divided by that of the uncompressed data) as our main performance metric.

For each exercise, we compare three data compression methods.

- **Zlib.** This lossless method is used in [14] to compress motion data. It first accumulates pieces of sensing data in a fix-sized buffer, compresses the buffer by the zlib compression algorithm once it is full, and transmits the compressed buffer directly to the sink. So, it compresses data in a block-by-block fashion. Note that it has no sense of overhearing in its design.
- **MT.** This is our MT method.



(a)



(b)

Fig. 7. The effect of γ. (a) Waveforms of $X_i^{(1)}$ before and after the low-pass filter. (b) Compression ratio versus γ.

- **MDT.** This is our MDT method with depth bounded by $\delta = 2$. The MDT tree is found by the approximation method in [30].
- **Diff.** To see the improvement made by exploiting spatial correlations, this method utilizes only differential coding, intranode correlations among sensors, and Huffman encoding. Technically, it is our MDT method with depth bounded by $\delta = 1$.

To make comparison, we also implement a method with no data compression, called *uncompressed* method. Note that we realize the "Huffman" block based on the implementations recommended by [36] and [37]. The other blocks, from our experiences, only take a few multiplications and additions to complete. This is also true for the FIR filter in the "Calibration" block. So, the feasibility of our scheme has been verified by our prototyping. The performance part done on a PC is mainly for comparing data compression ratio and energy consumption of different methods (since we do not have the implementations of other methods on our sensor platform). This PC-based simulation can correctly capture the intended metrics, except network behaviors (which will be modeled by some network simulations).

5.2 Effects of γ

In the "Calibration" block, the parameter γ of the low-pass filter tries to remove noises while retaining information of interest. Note that its purpose is to smooth out the raw data; it does not change the size of a piece of sensing data. Fig. 7a shows the low-pass-filtered $X_i^{(1)}$ in the x-axis of exercise Fig. 3d for $\gamma = 0$, 30, and 60 percent (the results for y-axis and z-axis are similar and are not shown here). Fig. 7b shows the compression ratios achieved by different γs when the MT method is applied. Considering both accuracy and compression ratio, we will set $\gamma = 30$ percent in the rest of our experiments.

5.3 Compression Ratio under Ideal Channel Condition

First, we consider an ideal wireless channel. That is, the effect of network impairments, such as transmission errors, is ignored here. For MT, MDT, and Diff methods, we build the data compression models by the training data set and measure their online compression ratios by the test data set. For each method, the offline phase outputs a transmission scheduling tree T, the corresponding coefficients αs and βs, and the Huffman codewords for each sensor. Note that all nodes in the Diff method are level-1 nodes, so there is no overhearing. These results are applied to the online phase to compress the test data set. The "Huffman" block adopts the adaptive Huffman coding. For the Zlib method, we buffer the test data in every four seconds in a block and then conduct compression (note that in Zlib, the block size is a trade-off between compression ratio and delay [14]). Note that this is not an issue for our methods because our methods can perform compression on each piece of sensing data $\Delta X_i^{(1:m)}[t]$.

Fig. 8 shows the compression ratios achieved by individual nodes. Fig. 9a compares the overall compression ratios after summing up the data transmitted by all nodes. The corresponding partial-ordering trees are shown in Fig. 9b. Our MT, MDT, and Diff outperform Zlib significantly in all nodes and all exercises. Note that nodes that move more frequently exhibit higher compression ratios. Also note that Diff can be regarded as a special case of our scheme which only tries to exploit the correlation of intranode data. So, these performance curves show the importance of exploiting the correlation of internode data. For most nodes, MT outperforms MDT by small gaps (for very few nodes, MDT may outperform MT because MT tries to minimize the average of compressed data sizes). From Fig. 9b, we also see that the number of level-1 nodes of a partial-ordering tree has major impact on the compression ratio of a scheme (Diff can be regarded as consisting of only level-1 nodes). This observation again proves the importance of exploiting spatial correlations. We believe that this is even more important when the network becomes larger. While the overall performance of MT is always better than MDT, MDT is much simpler and is more robust against network impairments, as to be shown later on.

5.4 Compression Ratio under Nonideal Channel Condition

Since MT and MDT rely on overhearing, we intend to evaluate how packet loss may degrade the performance of our methods (note that when a node fails to overhear the packets that it expects to receive, it will simply compress its data based on temporal and intranode spatial correlations). The packet loss probabilities of wireless links typically vary

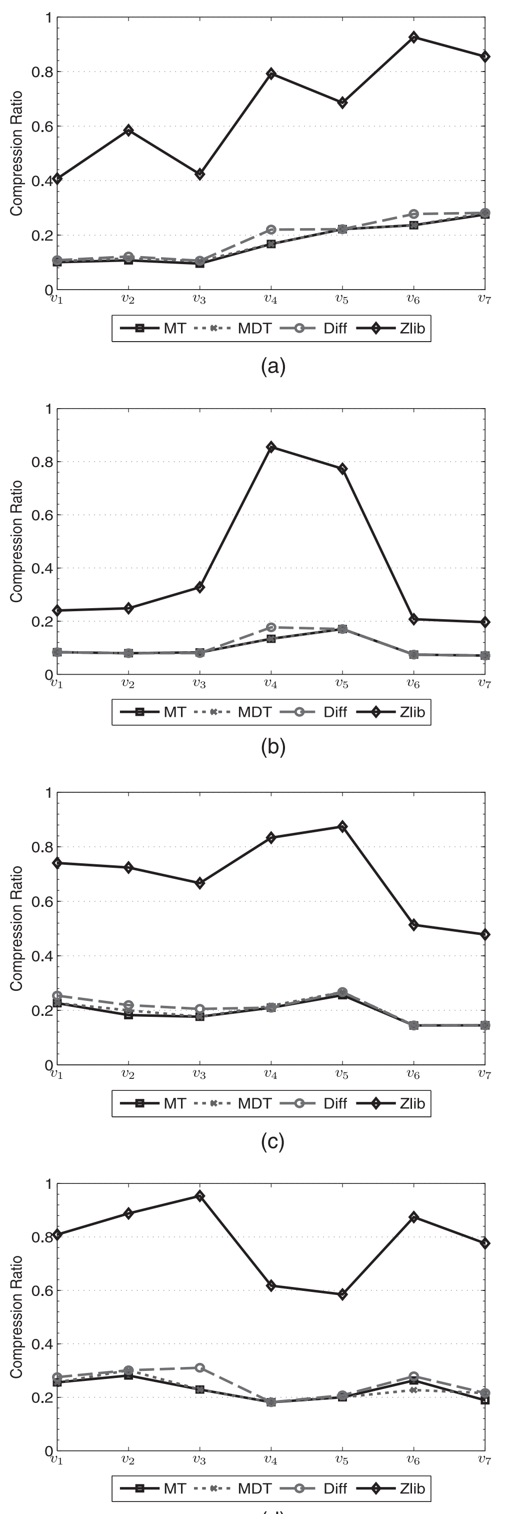
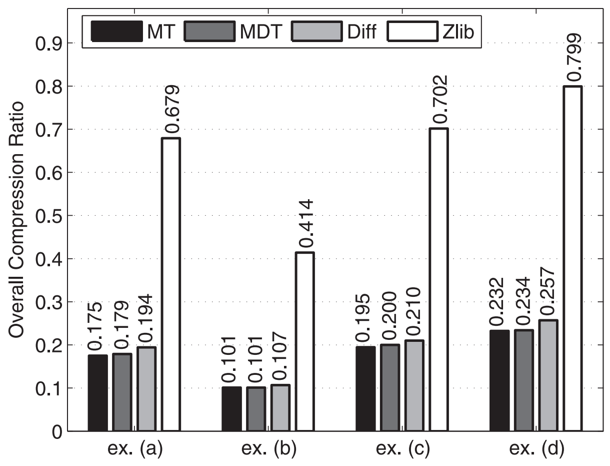
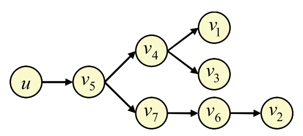
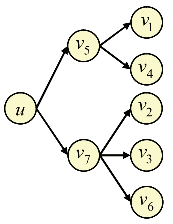
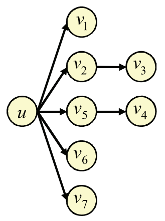
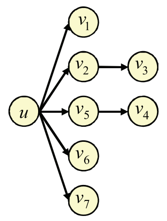
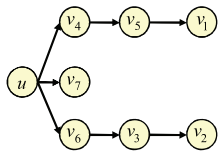
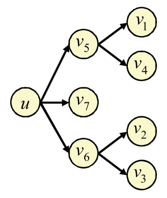
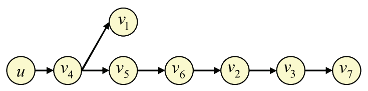
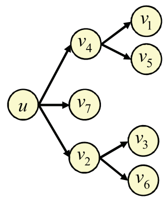


Fig. 8. Comparison of compression ratios of individual nodes ($v_1, v_2, \ldots, v_7$) under an ideal wireless channel. (a) Exercise Fig. 3a. (b) Exercise Fig. 3b. (c) Exercise Fig. 3c. (d) Exercise Fig. 3d.



(a)

exer cise	MT	MDT
(a)		
(b)		
(c)		
(d)		

(b)

Fig. 9. Comparison of overall compression ratios and the partial-ordering trees being used. (a) Overall compression ratio. (b) Partial-ordering trees.

over time and space, depending on the environment [38]. We simulate the network behavior by MATLAB because the probability of packet loss in a real testbed is not controllable. In our simulation, we set a packet loss probability p_l for each wireless link (i.e., the network is fully connected, but each link has a loss probability of p_l) and vary p_l for the whole network to observe the compression performance. We believe that using the same p_l for all links suffices because our BSN is not large (with eight nodes) and varying p_l for individual links may bias the observed results.

We use a simple polling MAC, which works as follows:

1. The sink node sorts nodes of the partial-ordering scheduling tree T (shown in Fig. 9b) by a BFS traversal. The result is the polling order.

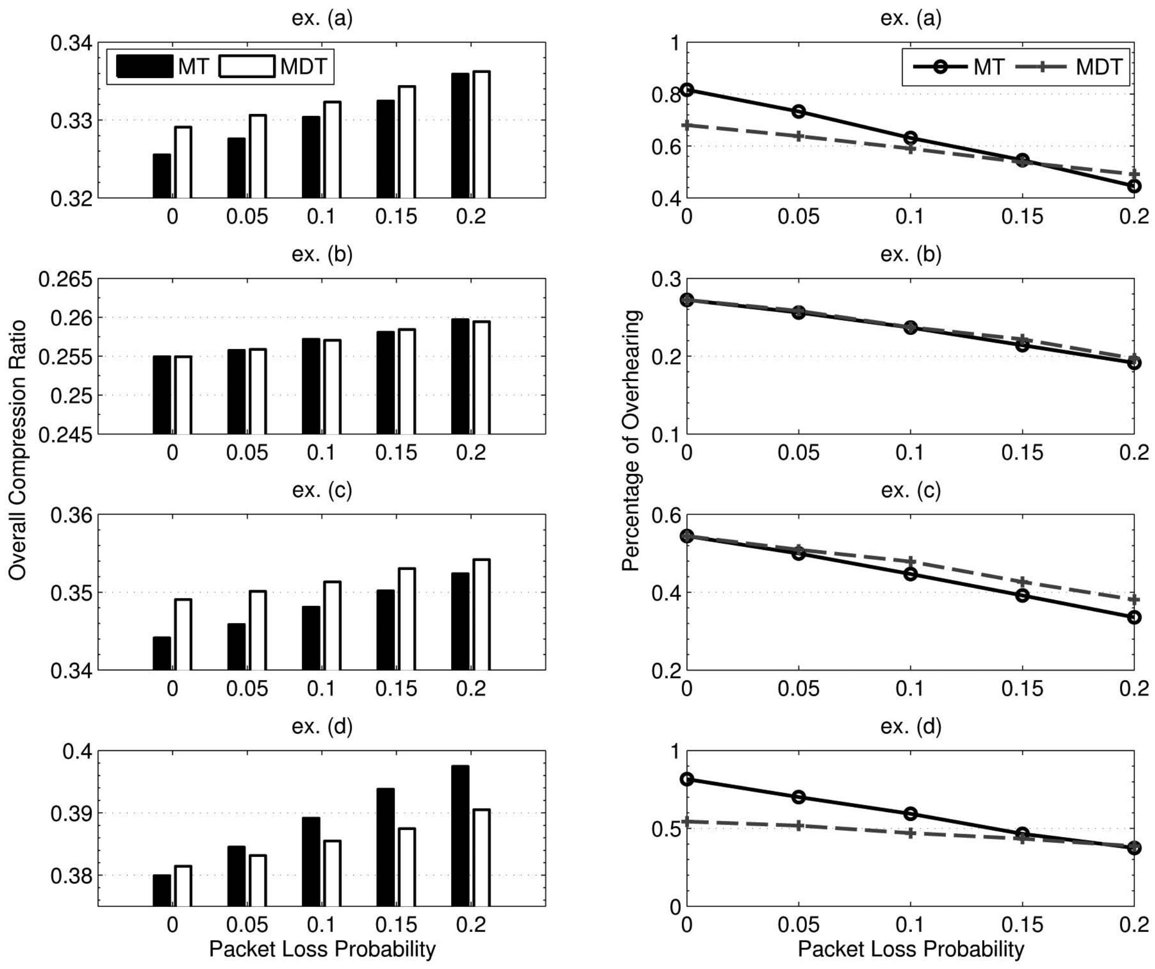


Fig. 10. The effect of wireless packet loss.

2. In each round, the sink node polls nodes according to the above polling order. It asks each polled node v_i to either report $\epsilon_{(i,1:m)|\phi}$ or $\epsilon_{(i,1:m)|j_p}$ depending on whether the sink node could recover the received data.
3. Upon receiving the polling request, the polled node broadcasts its reply packet, which contains 10 consecutive pieces of sensing data. All other nodes, including the sink node, have a probability of $(1-p_l)$ to correctly receive the packet.
4. The sink node retransmits its polling request at most twice in case of packet loss, and then polls the next node.

Note that in the above MAC protocol, the event of packet loss is solely based on each node's own observation of success/failure (and thus has the same loss probability of p_l). We run the simulation 1,000 times and calculate the overall compression ratio over all nodes. The polling message sent by the sink node is a 1-byte packet consisting of the node ID of the polled node and the expected type of reply ($\epsilon_{(i,1:m)|\phi}$ or $\epsilon_{(i,1:m)|j_p}$). The reply packet (for MT and MDT) contains a 9-byte MAC header and a variable-size payload (say, $\epsilon_{(i,1:m)|j_p}[t], \epsilon_{(i,1:m)|j_p}[t+1], \ldots, \epsilon_{(i,1:m)|j_p}[t+9]$). To eliminate the impact of packet loss, the MAC header always includes the first piece of the original sensing data, i.e., $X_i^{(1:m)}[t]$. For the uncompressed method, the reply packet contains a 2-byte MAC header and a fix-size payload (say, $X_i^{(1:m)}[t], X_i^{(1:m)}[t+1], \ldots, X_i^{(1:m)}[t+9]$).

Fig. 10 shows the overall compression ratios and the corresponding overhearing percentages for each exercise under different methods, where the overhearing percentage is defined as the number of reply packets that send $\epsilon_{(i,1:m)|j_p}$ divided by the total number of reply packets. Intuitively, a higher percentage of overhearing will help reduce the overall compression ratio. As we can see, as p_l increases, both MT's and MDT's compression ratios will increase, but MDT will gradually outperform MT as p_l increases.

5.5 Energy Cost

As energy consumption is an important factor in BSNs, we conduct experiments to observe how our methods perform with respect to this. We adopt the polling MAC protocol in Section 5.4. Table 1 summarizes the parameters used in this evaluation. The round length of the polling MAC will depend on the sampling rate. In the beginning of each round, each node has to wake up. The sink node then polls nodes according to the polling order. In the case of packet loss from the polled node, the sink can repoll at most two more times. Loss is determined by the polling timeout parameter. Each node has to keep on listening to the channel until it receives a polling request. After being polled and transmitting its data, the node can turn off its transceiver until the beginning of the next round. Note that

TABLE 1
Energy-Related Parameters Used in Our Evaluation

Parameter	Value
Round length	250 ms
Polling timeout	10 ms
Polling retry limit	2
Overhead of uncompressed MAC	2 bytes
MAC overhead of MT (MDT)	9 bytes
PHY overhead	16 bytes
RF reception power	35.5 mW
RF transmission power	35.5 mW
RF idle power	10 mW
Bit time	4 μs
RX/TX switching time	10 μs

after transmission, a node has to wait for an additional polling timeout interval before turning off its transceiver to make sure that the sink will not repoll it.

The power consumption of a sensor node mainly comes from its computing module, sensor module, and wireless module. Here, we are mainly interested in the power consumption of the wireless module because the other factors are irrelevant to our work. The parameters in Table 1 are mainly based on those in [39] and [40], which consider IEEE 802.15.4-compatible transceivers operating in 2.4 GHz with a data rate of 250 kbps. A node can switch among three modes: reception, transmission, and idle modes. When being turned off, a wireless module consumes no power. The bit time is the reverse of the data rate. The RX/TX switching time specifies the time interval required for a reception followed by an immediate transmission to account the nodal processing time.

We mainly observe the average energy consumption per-node per-polling round. Assuming the same initial battery energies for all nodes, it follows that the node that exhausts the maximal average energy per round dies first. In Fig. 11, we run our simulation 1,000 rounds for each exercise and show the maximal per-node, per-round energy consumption among all nodes under different packet loss probabilities. As we can see, both MT and MDT save over 40 percent of energy as opposed to the uncompressed case.

5.6 Scalability Issue

In addition, given a fixed channel bandwidth, the overall compression ratio also reflects how many sensor nodes may coexist in a BSN. So, this also implies the scalability of a method. For example, in Fig. 10, the compression ratios of both MT and MDT are upper bounded by 0.4. This implies that our methods can roughly tolerate a BSN that is 2.5 times that of the uncompressed method. Of course, the impact of contention and collision among nodes is not taken into consideration. Since our work is not targeted at MAC design, we will leave this issue for future study.

5.7 Effect of System Retraining

In the previous experiments, we have assumed that the training and testing data are collected from the same person. In reality, a set of training data may be used to serve a different person who performs exercises in a different fashion. In this case, the system needs to be retrained, as discussed in Section 4.5, when we find deviations between the training parameters and the actual behaviors. Fortunately, our data compression is lossless, which means that any deviation can be accurately captured. The cost is a higher compression ratio before the retraining is done. We evaluate this problem below.

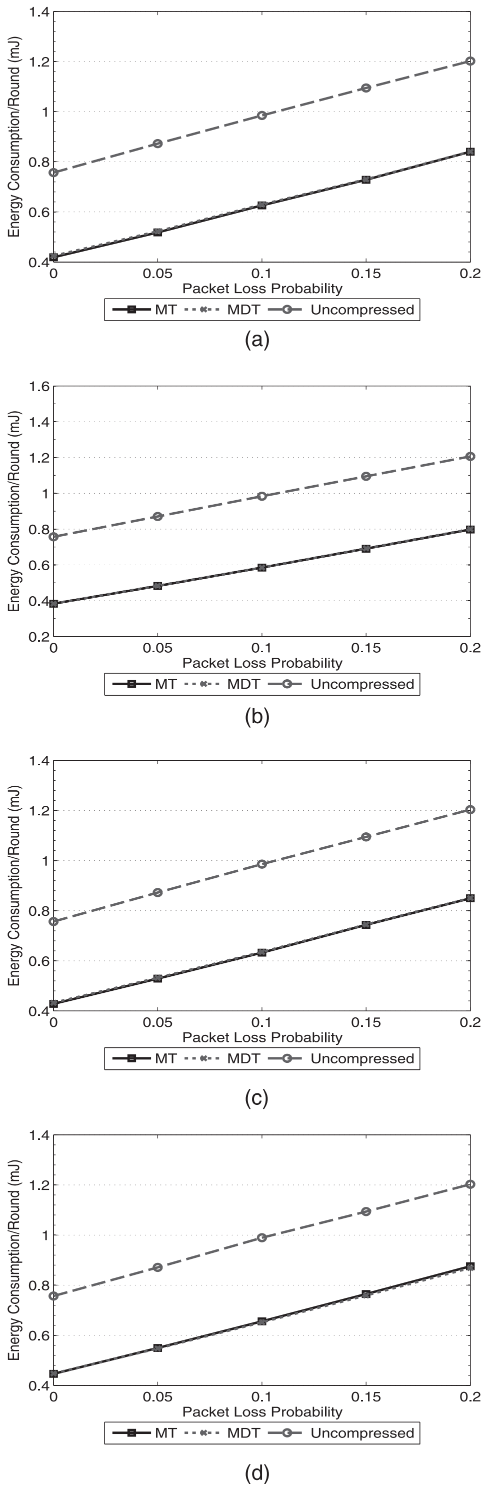


Fig. 11. Comparison of the maximal average per-node energy consumption among all nodes for each exercise. (a) Exercise Fig. 3a. (b) Exercise Fig. 3b. (c) Exercise Fig. 3c. (d) Exercise Fig. 3d.

We conduct this experiment with two users. A window-based method is adopted to keep the most recent sensing data. Before the network starts, the initial data compression model is built by the training data of one user. Recall that a model includes coefficients αs and βs in (2), Huffman

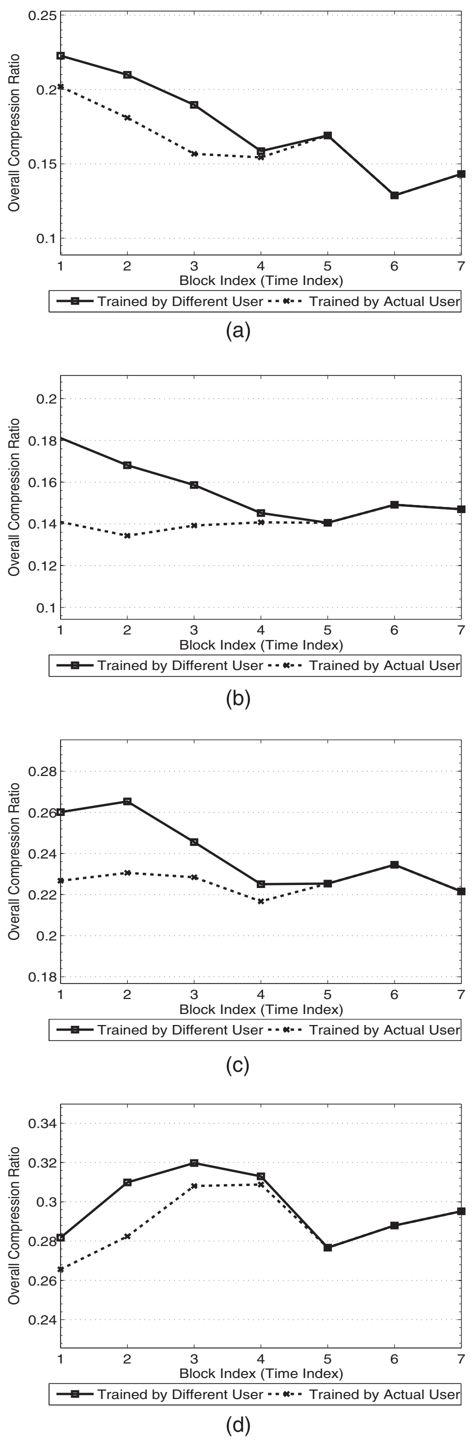


Fig. 12. The effect of system retraining on overall compression ratio. (a) Exercise Fig. 3a. (b) Exercise Fig. 3b. (c) Exercise Fig. 3c. (d) Exercise Fig. 3d.

codewords, and partial-ordering trees. Then, the second user conducts exercises on the system. Upon receiving new sensing data, the data analyzer inserts it at the end of the sliding window and removes the same amount of old data from the window. In this evaluation, a sliding window which can keep 20 seconds of sensing data is used. Originally, it contains sensing data from the first person. Testing data are divided into blocks of length of 5 seconds. These blocks will gradually replace all data of the first user. In Fig. 12, we evaluate the overall compression ratio under ideal channels before the ith block is moved into the sliding window. To make comparison, we also show overall compression ratios where the initial data compression models are trained by the actual user who performs the online testing (i.e., the second person). We can see that there is a higher compression ratio before retraining starts. The gap diminishes as the second person's data gradually replace the first person's in the sliding window. After the fifth block, the sliding window is completely filled with the actual user's data, so there is no deviation between the training parameters and the actual behaviors. We can see that our data compression model can be accurately retrained within a short delay.

6 Conclusions

In this paper, we have introduced a novel data compression framework to attack the multisensor, multinode compression problem in BSNs. We exploit the temporal and spatial correlations among sensing data from multiple sensor nodes by differential coding and overhearing among sensor nodes via linear regression. We also formulated the transmission-ordering problem to determine nodes' overhearing sequence to better exploit spatial correlations and thus minimize the total amount of transmissions. We proposed two scheduling methods, which differ in the longest sequence of nodes that a node needs to overhear. Our experiments on capturing Pilates exercises demonstrated the effectiveness of our approach. Our approach only requires a sensor node to overhear at most one other node. One future direction that deserves further research is to exploit the correlation between a node and a set of other nodes.

Acknowledgments

Yu-Chee Tseng's research was cosponsored by the MoE ATU Plan; by NSC grants 97-3114-E-009-001, 97-2221-E-009-142-MY3, 98-2219-E-009-019, and 98-2219-E-009-005; by MOEA 98-EC-17-A-02-S2-0048 and 98-EC-17-A-19-S2-0052; by ITRI, Taiwan; by III, Taiwan; and by Intel.

References

[1] D.T.W. Fong, J.C.Y. Wong, A.H.F. Lam, R.H.W. Lam, and W.J. Li, "A Wireless Motion Sensing System Using ADXL MEMS Accelerometers for Sports Science Applications," *Proc. World Congress on Intelligent Control and Automation,* 2004.

[2] J. Brutovsky and D. Novak, "Low-Cost Motivated Rehabilitation System for Post-Operation Exercises," *Proc. Int'l Conf. IEEE Eng. in Medicine and Biology Soc. (EMBS '06),* 2006.

[3] J.M. Winters, Y. Wang, and J.M. Winters, "Wearable Sensors and Telerehabilitation: Integrating Intelligent Telerehabilitation Assistants with a Model for Optimizing Home Therapy," *IEEE Eng. in Medicine and Biology Magazine,* vol. 22, no. 3, pp. 56-65, May/June 2003.

[4] S. Kang, J. Lee, H. Jang, Y. Lee, S. Park, and J. Song, "A Scalable and Energy-Efficient Context Monitoring Framework for Mobile Personal Sensor Networks," *IEEE Trans. Mobile Computing,* vol. 9, no. 5, pp. 686-702, May 2010.

[5] C.-H. Wu, Y.-T. Chang, and Y.-C. Tseng, "Multi-Screen Cyber-Physical Video Game: An Integration with Body-Area Inertial Sensor Networks," *Proc. IEEE Int'l Conf. Pervasive Computing and Comm. Workshops (PERCOM Workshops),* 2010.

[6] D. Vlasic, R. Adelsberger, G. Vannucci, J. Barnwell, M. Gross, W. Matusik, and J. Popović, "Practical Motion Capture in Everyday Surroundings," *ACM Trans. Graphics,* vol. 26, no. 3, p. 35, 2007.

[7] D. Cavalcanti, R. Schmitt, and A. Soomro, "Performance Analysis of 802.15.4 and 802.11e for Body Sensor Network Applications," *Proc. Int'l Workshop Wearable and Implantable Body Sensor Networks (BSN '07),* 2007.

[8] B. de Silva, A. Natarajan, and M. Motani, "Inter-User Interference in Body Sensor Networks: Preliminary Investigation and an Infrastructure-Based Solution," *Proc. Int'l Workshop Wearable and Implantable Body Sensor Networks (BSN '09),* 2009.

[9] A. Deshpande, C. Guestrin, S.R. Madden, J.M. Hellerstein, and W. Hong, "Model-Driven Data Acquisition in Sensor Networks," *Proc. Int'l Conf. Very Large Data Bases (VLDB '04),* 2004.

[10] D. Chu, A. Deshpande, J.M. Hellerstein, and W. Hong, "Approximate Data Collection in Sensor Networks Using Probabilistic Models," *Proc. Int'l Conf. Data Eng. (ICDE '06),* 2006.

[11] G. Zhou, J. Lu, C.-Y. Wan, M. Yarvis, and J. Stankovic, "BodyQoS: Adaptive and Radio-Agnostic QoS for Body Sensor Networks," *Proc. IEEE INFOCOM,* 2008.

[12] D. Jea, W. Wu, W.J. Kaiser, and M.B. Srivastava, "Approximate Data Collection Using Resolution Control Based on Context," *Proc. Int'l Workshop Wearable and Implantable Body Sensor Networks (BSN '07),* 2007.

[13] P.K. Baheti and H. Garudadri, "An Ultra Low Power Pulse Oximeter Sensor Based on Compressed Sensing," *Proc. Int'l Workshop Wearable and Implantable Body Sensor Networks (BSN '07),* 2009.

[14] L. Cheng, S. Hailes, Z. Cheng, F.-Y. Fan, D. Hang, and Y. Yang, "Compressing Inertial Motion Data in Wireless Sensing Systems - An Initial Experiment," *Proc. Int'l Workshop Wearable and Implantable Body Sensor Networks (BSN '08),* 2008.

[15] H. Ghasemzadeh, E. Guenterberg, and R. Jafari, "Energy-Efficient Information-Driven Coverage for Physical Movement Monitoring in Body Sensor Networks," *IEEE J. Selected Areas in Comm.,* vol. 27, no. 1, pp. 58-69, Jan. 2009.

[16] E. Guenterberg, H. Ghasemzadeh, and R. Jafari, "A Distributed Hidden Markov Model for Fine-Grained Annotation in Body Sensor Networks," *Proc. Int'l Workshop Wearable and Implantable Body Sensor Networks (BSN '09),* 2009.

[17] A. Scaglione and S.D. Servetto, "On the Interdependence of Routing and Data Compression in Multi-Hop Sensor Networks," *Proc. ACM MobiCom,* 2002.

[18] T. ElBatt, "On the Trade-Offs of Cooperative Data Compression in Wireless Sensor Networks with Spatial Correlations," *IEEE Trans. Wireless Comm.,* vol. 8, no. 5, pp. 2546-2557, May 2009.

[19] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks," *ACM SIGOPS Operating Systems Rev.,* vol. 36, pp. 131-146, 2002.

[20] Y.-C. Wang, Y.-Y. Hsieh, and Y.-C. Tseng, "Multiresolution Spatial and Temporal Coding in a Wireless Sensor Network for Long-Term Monitoring Applications," *IEEE Trans. Computers,* vol. 58, no. 6, pp. 827-838, June 2009.

[21] A. Silberstein, G. Puggioni, A. Gelfand, K. Munagala, and J. Yang, "Suppression and Failures in Sensor Networks: A Bayesian Approach," *Proc. Int'l Conf. Very Large Data Bases (VLDB '07),* 2007.

[22] R. Rydeard, A. Leger, and D. Smith, "Pilates-Based Therapeutic Exercise: Effect on Subjects with Nonspecific Chronic Low Back Pain and Functional Disability: A Randomized Controlled Trial," *J. Orthopaedic and Sports Physical Therapy,* vol. 36, no. 7, pp. 472-484, 2006.

[23] I. Garciá, S. de Barros, and M. Saldanha, "Isokinetic Evaluation of the Musculature Involved in Trunk Flexion and Extension: Pilates Method Effect," *Revista Brasileira de Medicina do Esporte,* vol. 10, no. 6, pp. 491-493, 2004.

[24] C.V.C. Bouten, K.T.M. Koekkoek, M. Verduin, R. Kodde, and J.D. Janssen, "A Triaxial Accelerometer and Portable Data Processing Unit for the Assessment of Daily Physical Activity," *IEEE Trans. Biomedical Eng.,* vol. 44, no. 3, pp. 136-147, Mar. 1997.

[25] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing.* Prentice-Hall, 1989.

[26] D.C. Montgomery, E.A. Peck, and G.G. Vining, *Introduction to Linear Regression Analysis,* fourth ed. Wiley-Interscience, 2006.

[27] T.M. Cover and J.A. Thomas, *Elements of Information Theory.* Wiley-Interscience, 1991.

[28] Y.J. Chu and T.H. Liu, "On the Shortest Arborescence of a Directed Graph," *Science Sinica,* vol. 14, pp. 1396-1400, 1965.

[29] J. Edmonds, "Optimum Branchings," *J. Research of the Nat'l Bureau of Standards,* 1967.

[30] J. Naor and B. Schieber, "Improved Approximations for Shallow-Light Spanning Trees," *Proc. IEEE Symp. Foundations of Computer Science (SFCS '97),* 1997.

[31] T.T.H. Binh, R.I. McKay, N.X. Hoai, and N.D. Nghia, "New Heuristic and Hybrid Genetic Algorithm for Solving the Bounded Diameter Minimum Spanning Tree Problem," *Proc. ACM Conf. Genetic and Evolutionary Computation (GECCO '09),* 2009.

[32] M. Keally, G. Zhou, and G. Xing, "Watchdog: Confident Event Detection in Heterogeneous Sensor Networks," *Proc. IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS '10),* 2010.

[33] D.E. Knuth, "Dynamic Huffman Coding," *J. Algorithms,* vol. 6, no. 2, pp. 163-180, 1985.

[34] Jennic, JN5139 Datasheet, http://www.jennic.com/support/datasheets/jn5139_module_datasheet, 2011.

[35] OceanServer, "OS5000 Family - Triaxial Accelerometer," http://www.ocean-server.com, 2008.

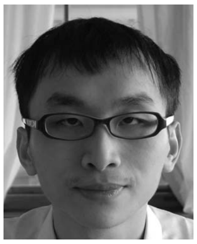
[36] H.-A. Pham, V.-H. Bui, and A.-V. Dinh-Duc, "An Adaptive, Memory-Efficient and Fast Algorithm for Huffman Decoding and Its Implementation," *Proc. Int'l Conf. Interaction Sciences (ICIS '09),* 2009.

[37] R. Hashemian, "Design and Hardware Implementation of a Memory Efficient Huffman Decoding," *IEEE Trans. Consumer Electronics,* vol. 40, no. 3, pp. 345-352, Aug. 1994.

[38] A. Natarajan, B.d. Silva, K.-K. Yap, and M. Motani, "Link Layer Behavior of Body Area Networks at 2.4 GHz," *Proc. ACM MobiCom,* 2009.

[39] J. Ammer and J. Rabacy, "The Energy-per-Useful-Bit Metric for Evaluating and Optimizing Sensor Network Physical Layers," *Proc. IEEE Sensor and Ad Hoc Comm. and Networks Conf. (SECON '06),* 2006.

[40] A.Y. Wang and C.G. Sodini, "A Simple Energy Model for Wireless Microsensor Transceivers," *Proc. IEEE Global Telecomm. Conf. (GlobeCom '04),* 2004.



Chun-Hao Wu received the BS degree in computer science from the National Chiao-Tung University, Taiwan, in 2007. He is currently working toward the PhD degree in the Department of Computer Science, National Chiao-Tung University. His research interests include mobile computing, wireless communication, wireless body-area sensor network, and parallel and distributed computing.

Yu-Chee Tseng received the PhD degree in computer and information science from the Ohio State University in January 1994. He is a professor (2000-present), the chairman (2005-2009), and an associate dean (2007-present) in the Department of Computer Science, National Chiao-Tung University, Taiwan. He was the adjunct chair professor at the Chung Yuan Christian University (2006-2010). His research interests include mobile computing, wireless communication, and parallel and distributed computing. He received the Outstanding Research Award from the National Science Council of China in 2001, 2003, and 2009, the Best Paper Award from the International Conference on Parallel Processing in 2003, the Elite I.T. Award in 2004, and the Distinguished Alumnus Award from the Ohio State University in 2005. He serves/served on the editorial boards for *Telecommunication Systems* (2005-present), *IEEE Transactions on Vehicular Technology* (2005-2009), *IEEE Transactions on Mobile Computing* (2006-present), and *IEEE Transactions on Parallel and Distributed Systems* (2008-present). He is a senior member of the IEEE.