Compression and Storage Schemes in a Sensor Network with Spatial and Temporal Coding Techniques

You-Chiun Wang, Yao-Yu Hsieh, and Yu-Chee Tseng
Department of Computer Science
National Chiao-Tung University, Hsin-Chu, 30010, Taiwan
Email: {wangyc, shiehyy, yctseng}@cs.nctu.edu.tw

***Abstract*—Wireless sensor networks provide a convenient manner to monitor the physical environments. How to extend the network lifetime by reducing the amount of message transmissions is a critical issue. In this paper, we propose a *multi-resolution compression and storage (MCS) framework* to compress and preserve sensing data in a wireless sensor network. Our MCS framework adopts spatial and temporal compression schemes to reduce the amount of message transmissions, so the network lifetime can be prolonged and the network congestion can be alleviated. In addition, we also develop a storage mechanism to maintain sensing data in sensor nodes, so that users can query more detailed data when necessary. Our proposed methods consider the hardware limitations of sensor nodes. We also implement a prototyping system on the MICAz Mote platform.**

***Index Terms*—data correlation, message compression, pervasive computing, sensor data management, wireless sensor networks.**

I. Introduction

Wireless sensor networks provide a new opportunity for pervasive and context-aware monitoring of physical environments. Such a network consists of a large amount of *sensor nodes*, where each node is a tiny wireless device that can continuously collect information from its surrounding environment and report to a remote sink through a multi-hop ad hoc manner. A wireless sensor network is usually deployed in a region of interest to observe abnormal phenomena or track objects inside that region. Wireless sensor networks can enrich our daily life through many applications, such as environment monitoring, smart home, and surveillance [1]–[3].

Because sensor nodes are usually operated by small batteries and it is infeasible to recharge them or deploy new nodes in many scenarios, how to extend the network lifetime becomes an important issue. In this paper, we consider wireless sensor networks that possess the following characteristics. First, these wireless sensor networks are deployed in some particular regions to provide long-term monitoring of the regions. In this case, because sensor nodes should continuously report what they sense to the sink, the communication overhead will dominate their energy consumption. Moreover, sensor nodes close to the sink will suffer from heavy loads of message transmissions. This will lead to network congestion [4] and rapidly consume the energy of sensor nodes around the sink. As these nodes exhaust their energy, the network may be broken. Therefore, how to reduce the amount of message transmissions of sensor nodes plays a leading role in extending the network lifetime. Second, sensing data reported from sensor nodes often exhibit a certain degree of data correlation. In particular, the sensing readings of neighboring sensor nodes may present high *spatial correlation* because they collect data from the same environment. These sensor nodes may detect either the same phenomenon or nothing from the environment. In addition, the sensing data collected by an individual sensor node may present high *temporal correlation* when its surrounding environment remains stable. With this observation, we can properly compress the sensing reports from sensor nodes to reduce the amount of message transmissions. Third, people may query different resolutions of sensing data from a wireless sensor network [5]. They may periodically request a rough report from the sink to obtain an overview of the monitoring environment. Sometimes, they could have interest to query more detailed information from a subset of sensor nodes. With this requirement, sensor nodes should not only simply report what they sense to the sink, but also have to maintain the sensing data in their local memories for further queries.

In this paper, we propose a *multi-resolution compression and storage (MCS) framework* to provide message compression and multiple resolutions of sensing data in wireless sensor networks. The basic concept of our MCS framework is to organize the network into a hierarchical architecture and then establish multi-resolution summaries of sensing data by spatial and temporal compressions. These summaries of sensing data will be maintained in the network for further query. Specifically, we organize the sensor nodes into multiple layers. Messages transmitted by nodes in a lower layer will be compressed by a certain node in the upper layer through the spatial coding technology. Meanwhile, each node can also compress its sensing report by temporal coding method. In this way, the amount of message transmissions can be significantly reduced and thus the network lifetime can be prolonged. Moreover, in the MCS framework, nodes in each layer will maintain a copy of their historical sensing data. Therefore, users can query different resolutions and views of sensing

data from different layers. In particular, they can obtain a coarser resolution but broader view of sensing data from a higher layer, and a narrower view but finer resolution from a lower layer. Our proposed compression and storage schemes consider the hardware limitations of sensor nodes. We also implement a prototyping system on the MICAz Mote platform [6] to evaluate the system performance.

In the literature, data compression for textual content has been well studied. *Huffman coding* [7] and *Lempel-Ziv-Welch (LZW) scheme* [8] are two well-known text-coding compression algorithms. They can provide lossless text compression, in the sense that data can be completely recovered after decompression. However, the results of these text-coding compression algorithms cannot be directly applied to the message compression in wireless sensor networks. This is because the textual data are composed of a finite set of alphabets whereas the sensing readings of a wireless sensor network are usually continuous values. These algorithms cannot define the alphabets of sensing readings and thus fail.

In-network data aggregation [9]–[11] also discusses how to reduce the amount of message transmissions in a wireless sensor network with data similarity. These data aggregation methods attempt to fuse a set of similar sensing reports and generate one representative value to stand for these reports. Nevertheless, unlike data compression, this fusion operation is irreversible in the sense that the original sensing reports cannot be recovered. Therefore, the subtle difference between sensing reports cannot be reflected because they have been fused together.

In [5], a storage architecture called *DIMENSIONS* is proposed to support multi-resolution storage in a wireless sensor network. DIMENSIONS organizes the network into multiple levels and adopts a wavelet compression method in each level to generate spatiotemporal summarization of data. Users can obtain multiple resolutions of sensing reports from different levels via drill-down queries. However, such a wavelet compression requires a large cost of computation power and memory size. In DIMENSIONS, these high-cost wavelet compression and decompression operations are performed in each level, so sensor nodes may suffer from higher computation and space complexity.

The rest of this paper is organized as follows. Section II proposes our MCS framework. Section III presents the compression and storage schemes. Section IV reports our prototyping experiences and some experimental results. Section V concludes this paper.

II. Multi-resolution Compression and Storage (MCS) Framework

The system architecture of our MCS framework is illustrated in Fig. 1. We consider that sensor nodes are homogeneous and they are arbitrarily deployed in the region of interest. In MCS, we recursively divide the network into α blocks. In this way, the network will be 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* to collect and compress sensing reports from the corresponding α blocks in its lower layer. The number of layers decides the resolutions and message sizes of sensing reports, and it can be adjusted by users depending on their application requirements.

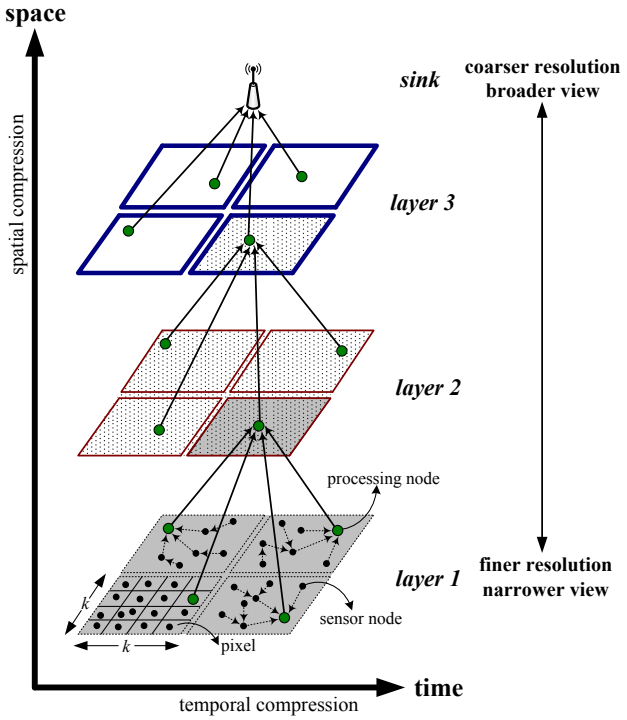

Fig. 1. System architecture of the multi-resolution compression and storage (MCS) framework. There are three layers and $\alpha = 4$.

In the lowest layer (i.e., layer 1), the processing node is responsible for compressing sensing reports from sensor nodes. In this layer, we further divide each block into $k \times k$ grids (called *pixels*), where k is a small integer. Ideally, each pixel should contain exact one sensor node and we use the sensing report of this node as the pixel's value. However, because sensor nodes are arbitrarily deployed in the region of interest, a pixel may possibly contain two or more sensor nodes. In this case, we take the average of sensing reports from these sensor nodes as the pixel's value. One the other hand, when a pixel contains no node, we can use the average of values from the neighboring pixels as this pixel's value.

In MCS, sensing data reported from sensor nodes are transmitted to the sink layer by layer. Messages passed through each layer will be compressed by the corresponding processing node in each block, through the *spatial compression scheme*. Sensor nodes and layer-1 processing nodes will also compress their data along the time axis by the *temporal compression scheme*. To help users query past data from the network, each node will store the historical data in its local memory by the *storage scheme*.

Since the spatial and temporal compression schemes will cause some loss of data precision (depending on the compression ratio), processing nodes in different layers will provide different resolutions of sensing data. In particular, users can

obtain a coarser resolution (but broader view) by querying processing nodes in a higher layer. When users have interest in accessing more in-depth data, they can query the processing nodes in a lower layer. Note that each processing node stores and reports compressed data and these data are decompressed only at the sink. In this way, both the computation and space complexity of processing nodes can be greatly reduced.

Next, we present our compression and storage schemes in detail.

III. Compression and Storage Schemes

A. Spatial Compression Scheme

The spatial compression scheme is performed by each processing node to compress sensing data from its lower layer. Users can specify a *compression ratio* γ, $0 \leq \gamma < 1$, which is defined as the reduction in size relative to the uncompressed size through each layer, that is,

$$\gamma = 1 - \frac{\text{compressed size}}{\text{uncompressed size}}.$$

Then a processing node will compress the sensing data based on their spatial correlation. According to the layer number, the spatial compression scheme contains three procedures: *layer-1 compression*, *layer-i ($i > 1$) compression*, and *decompression* (at the sink).

1) Layer-1 Compression Procedure: A layer-1 processing node collects the sensing data from the sensor nodes in its block and will store them in a $k \times k$ matrix $\mathcal{M} = (s_{i,j})_{k\times k}$, where each element $s_{i,j}$ records the value of a pixel (i, j) specified in Fig. 1. Then, the processing node can apply the *two-dimensional discrete cosine transform (2D-DCT) method* [12] on $\mathcal{M}$ to generate a new matrix $\mathcal{M}' = (t_{i,j})_{k\times k}$. In particular, for each element $t_{i,j} \in \mathcal{M}'$, we have

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

where $c(i) = \frac{1}{\sqrt{2}}$ if $i = 0$ and $c(i) = 1$ otherwise. The 2D-DCT method is widely used in the field of image processing. It can transform an image from the spatial domain to the frequency domain and extract significant values of the image. In particular, the 2D-DCT method will compact those significant values in the upper-left part of the transformed matrix $\mathcal{M}'$, while leaving other insignificant values in the opposite part. In this way, we can still maintain most significant characteristics of the original matrix $\mathcal{M}$ by preserving only the upper-left part of the transformed matrix $\mathcal{M}'$ and thus achieve message compression. However, the cosine operations in Eq. (1) are too complicated for sensor nodes to calculate. Fortunately, the variable k in the cosine operation is a system constant (i.e., the length of matrix $\mathcal{M}$) while the other four variables i, j, x, and y are non-negative integers no larger than k. Since k is a small integer, we can maintain a small table in each processing node to record the results of cosine operations with different pairs of inputs (i, x) and (j, y). Therefore, the calculation of Eq. (1) can be simplified as operations of additions and multiplications.

After calculating the matrix $\mathcal{M}'$, a *reduced zigzag scan (RZS) method* is applied to translate $\mathcal{M}'$ into an one-dimensional array $\mathcal{D}$ to compress data. In particular, the RZS method begins at the upper-left corner of $\mathcal{M}'$ and sequentially scan the diagonals of $\mathcal{M}'$, as shown in Fig. 2. The RZS method stops when it has scanned $\lceil k^2 \cdot \lambda \rceil$ elements of $\mathcal{M}'$, where $\lambda = 1 - \gamma$ is the ratio of elements in $\mathcal{M}'$ to be preserved. Then, we transmit the array $\mathcal{D}$ to the layer-2 processing node. With the property of 2D-DCT, the RZS method is guaranteed to maintain most significant values of the matrix $\mathcal{M}$ in the array $\mathcal{D}$. Note that the compression ratio γ affects the data precision after decompressing $\mathcal{D}$ to the original matrix $\mathcal{M}$. Specifically, we can maintain more information in the array $\mathcal{D}$ as γ becomes smaller. Therefore, the data precision can be increased after decompressing $\mathcal{D}$.

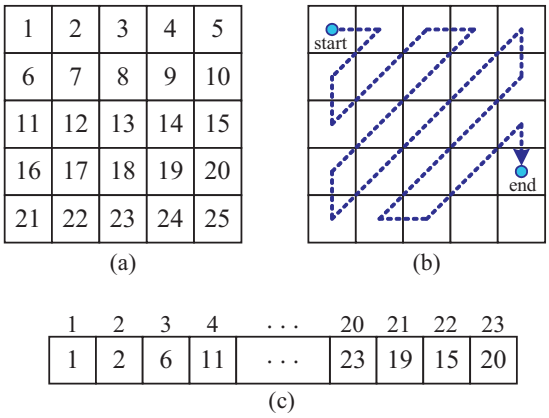

Fig. 2. An example of the RZS method, where the compression ratio $\gamma = 0.1$. (a) Indices of matrix $\mathcal{M}'$. (b) Reduced zigzag scan. (c) The result of one-dimensional array.

2) Layer-i ($i > 1$) Compression Procedure: A layer-i processing node further compresses the data from its corresponding α blocks in the lower layer. Intuitively, one possible method is to first decompress the data from the lower layer and then apply the 2D-DCT method again on these decompressed data to recompress them. Unfortunately, this intuitive method has two drawbacks. First, because the 2D-DCT method and its inversion are two expensive operations, the processing node will suffer from high computation complexity. The situation becomes worse in a higher layer because the processing node needs to handle a large amount of data from many nodes. Second, this intuitive method may not help compress more data because in a large range of sensing field, the degree of spatial correlation of sensing data will degrade.

Therefore, in layer-i compression, we reduce the length of array $\mathcal{D}$ (passed from the layer $i-1$) to $\lceil \lambda^i \cdot k^2 \rceil$ elements by discarding the last $\lfloor \lambda^{i-1} \cdot k^2 - \lambda^i \cdot k^2 \rfloor$ elements of $\mathcal{D}$. Recall that the array $\mathcal{D}$ stores data in an decreasing order according to the data importance. Therefore, we can reduce the size of sensing data with a ratio of λ^i in layer i and still maintain the significant values of these data.

3) Decompression Procedure: To reduce the computation overhead of processing nodes, the decompression procedure is performed only at the sink. In particular, once the sink has collected the reports from the processing nodes in the highest layer, it first recovers the spatial locations of blocks in each layer (as shown in Fig. 1). Then, for each layer-1 block, the sink recovers the corresponding array $\mathcal{D}$ to a two-dimensional matrix $\mathcal{M}' = (t_{i,j})_{k\times k}$. Since the array $\mathcal{D}$ contains only $\lceil \lambda^d \cdot k^2 \rceil$ elements of compressed data, where d is the number of layers in the network, we should fill the remaining $\lfloor k^2 - \lambda^d \cdot k^2 \rfloor$ elements with zeros in the matrix $\mathcal{M}'$. Finally, we adopt the inverse 2D-DCT method to transform $\mathcal{M}'$ to a new matrix $\mathcal{M}'' = (s_{i,j})_{k\times k}$. In particular, for each element $s_{i,j} \in \mathcal{M}''$, we have

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

Note that since the matrix $\mathcal{M}'$ is incomplete, the transformed matrix $\mathcal{M}''$ by the inverse 2D-DCT method may not be necessarily equal to the original matrix $\mathcal{M}$ (that is, lose some data precision). The compression ratio γ decides both the size and precision of sensing reports, which is a tradeoff between each other.

B. Temporal Compression Scheme

The temporal compression scheme is performed by each sensor node and layer-1 processing node. Users can specify a small *update threshold* δ to determine whether a node should transmit its data or not. Specifically, for each sensor node, every time it transmits the sensing report to the layer-1 processing node, the sensor node keeps the value of report $v_{\text{reference}}$ as a reference value. Then, when the sensor node generates another sensing report with the value v_{current}, it checks whether $|v_{\text{reference}} - v_{\text{current}}| < \delta$. If so, it means that the difference between the current report and the previous report is insignificant. Therefore, the sensor node should not transmit the new generated report to preserve its energy. Otherwise, the sensor node has to transmit this report and replace $v_{\text{reference}}$ by v_{current} for the following reference.

For each layer-1 processing node, it should check a two-dimensional matrix rather than a single value. In particular, let $M_{\text{reference}} = (s_{i,j})_{k\times k}$ be the matrix previously transmitted to the layer-2 processing node. Once a layer-1 processing node generates a new matrix $M_{\text{current}} = (t_{i,j})_{k\times k}$, it first calculates the average difference of pixels between $M_{\text{reference}}$ and M_{current}:

$$\beta = \frac{1}{k^2}\sum_{i=0}^{k-1}\sum_{j=0}^{k-1}|s_{i,j} - t_{i,j}|.$$

If $\beta < \delta$, it means that the difference between $M_{\text{reference}}$ and M_{current} is insignificant and thus the processing node does not need to transmit the new generated matrix M_{current} to the upper layer. Otherwise, the processing node should report M_{current} to the layer-2 processing node and replaces $M_{\text{reference}}$ by M_{current}.

Note that a layer-i ($i > 1$) processing node will not conduct the temporal compression scheme because it stores only the matrix that has been transformed by the 2D-DCT method and thus it is very difficult to calculate the difference between two compressed matrices.

C. Storage Scheme

The aforementioned compression schemes will cause some data imprecision when passing sensing reports through each layer, so the data obtained from the sink would be a rough overview of the environment. However, we can maintain the historical data in processing nodes and sensor nodes, so users can query multiple resolutions of sensing data from the nodes in different layers. In this section, we propose a storage scheme to help nodes to store historical data in their limited memories.

Let n_i be the maximum number of frames that a node i can store in its local memory, where a *frame* is the unit of sensing data to be stored by node i. In particular, for a sensor node, a frame is the generated sensing report. For a processing node, a frame is the compressed matrix. Let us denote by f_t as the frame generated by a node at time $t \in \mathbb{N}$. Our storage scheme will maintain frames in an exponentially increasing order from t. Specifically, for a node i, we will store frames f_t, f_{t-1}, f_{t-3}, f_{t-7}, $\cdots$, and $f_{t-2^{n_i-1}+1}$. In this way, when the sink queries a past frame f_j ($j \in \mathbb{N}, j \le t$) from node i, three cases will be considered:

1. If f_j has been stored in node i's local memory, node i directly replies f_j to the sink.
2. If $t - 2^l + 1 < j < t - 2^{l-1} + 1$, where $l \in \mathbb{N}$ and $2 \le l < n_i - 1$, node i replies two frames $f_{t-2^{l-1}+1}$ and f_{t-2^l+1} to the sink.
3. If $j < t - 2^{n_i-1} + 1$, node i replies a fail message to the sink because f_j is too old to be stored in node i.

When the queried node is a processing node, the sink will adopt the inverse 2D-DCT method to decompress the received matrices. Note that in the above case 2, the sink should adopt a *linear interpolation* to calculate the queried frame f_j, that is,

$$f_j = f_{t-2^l+1} + \frac{(f_{t-2^{l-1}+1} - f_{t-2^l+1}) \times (j - t + 2^l - 1)}{2^l - 2^{l-1}}.$$

There are two advantages in the above storage scheme. First, a node can keep as old data as possible. Second, recent data can be more precise compared with those very old data.

IV. Implementation and Experimental Results

We use the MICAz Motes [6] as sensor nodes and processing nodes. A MICAz Mote is a 2.4 GHz, IEEE 802.15.4-compliant module that can support a data rate of 250 Kbps. In our current prototype, we deploy 16 Motes and implement an one-layer architecture, as shown in Fig. 3. We set the system parameters $\alpha = 4$ and $k = 2$. In this way, there are four layer-1 processing nodes, each being responsible for collecting and compressing data from its neighboring three sensor nodes. We apply the temporal compression scheme in the twelve sensor nodes and apply both the spatial and temporal compression

schemes in the four processing nodes. We use this prototype to collect indoor temperatures during 25 hours. The reporting interval of each node is set to ten minutes. The compression ratio γ is set to 0.25 in the spatial compression scheme and the update threshold δ is set to 0.2 °C in the temporal compression scheme. For each sensor node, the packet size of sensing report is set to 15 bytes, which contains 11 bytes of header and trailer and 4 bytes of payload. For each processing node, it will compress the sensing data from itself and its three corresponding sensor nodes. The size of packet reported from a processing node is set to 19 bytes, which includes a payload size of 8 bytes.

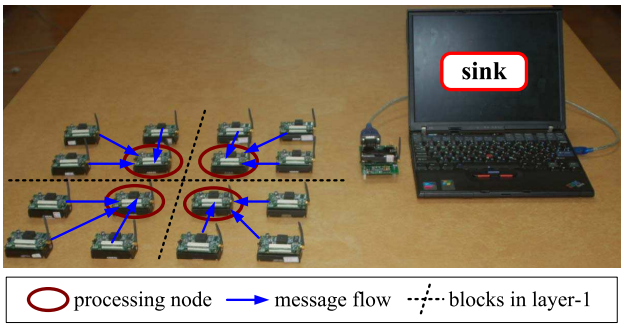

Fig. 3. A 16-mote prototype in our experiment.

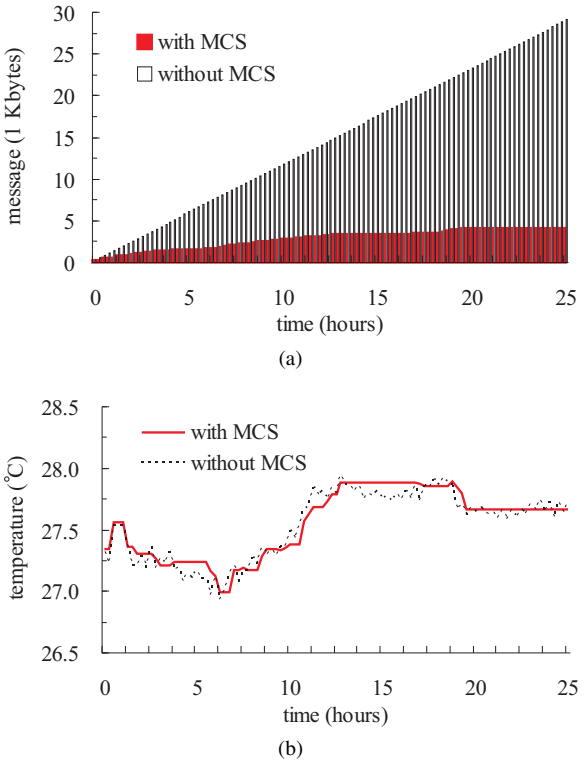

Fig. 4. Experimental results: (a) the total amount of message transmissions and (b) the average temperatures reported by the 16 nodes.

Fig. 4(a) shows the total amount of messages transmitted by the 16 Motes. As can be seen, the amount of message transmissions can be significantly reduced when our MCS framework is adopted. This is because the sensing reports with high data correlations can be compressed by the proposed spatial and temporal compression schemes. On the other hand, from Fig. 4(b), we can observe that the most significant characteristics of sensing reports can be preserved when the MCS framework is adopted. The maximum error is limited to 0.2 °C because of the update threshold in the temporal compression scheme.

V. Conclusions

In this paper, we have proposed the MCS framework to provide multi-resolution data compression and storage in a wireless sensor network. Our compression schemes can effectively reduce message transmissions of sensor nodes so that the network lifetime can be extended. Our storage scheme can help sensor nodes to store as much data as possible in their small memories. We have also implemented a prototyping system on the MICAz platform. It is verified that our MCS framework not only significantly reduces the message transmissions but also preserves important characteristics of sensing reports.

Acknowledgement

Y. C. Tseng's research is co-sponsored by Taiwan MoE ATU Program, by NSC grants 93-2752-E-007-001-PAE, 96-2623-7-009-002-ET, 95-2221-E-009-058-MY3, 95-2221-E-009-060-MY3, 95-2219-E-009-007, 95-2218-E-009-209, and 94-2219-E-007-009, by Realtek Semiconductor Corp., by MOEA under grant number 94-EC-17-A-04-S1-044, by ITRI, Taiwan, by Microsoft Corp., and by Intel Corp.

References

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

[2] 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.

[3] 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.

[4] C. T. Ee and R. Bajcsy, "Congestion control and fairness for many-to-one routing in sensor networks," in *ACM International Conference on Embedded Networked Sensor Systems*, 2004, pp. 148–161.

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

[6] Crossbow, "MOTE-KIT2400," http://www.xbow.com.

[7] M. Nelson and J. L. Gailly, *The data compression book (2nd edition)*. MIS:Press, New York, USA, 1996.

[8] J. A. Storer, *Data compression: methods and theory*. Computer Science Press, New York, USA, 1988.

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *ACM International Conference on Mobile Computing and Networking*, 2000, pp. 56–67.

[10] S. Lindsey, C. Raghavendra, and K. M. Sivalingam, "Data gathering algorithms in sensor networks using energy metrics," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 9, pp. 924–935, 2002.

[11] K. W. Fan, S. Liu, and P. Sinha, "Structure-free data aggregation in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, pp. 929–942, 2007.

[12] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transfom," *IEEE Transactions on Computers*, vol. 1, no. C-23, pp. 90–93, 1974.