Distributed Compressive Data Aggregation in Large-Scale Wireless Sensor Networks

Tsung-Yi Tsai, Wei-Chi Lan, Chunlei Liu, and Min-Te Sun

***Abstract*—As wireless sensor networks are used extensively in environment and habitat monitoring, the large volume of data transmission can increase the workload of the sensor nodes and reduce their useful lifetime. The compressive sampling techniques have been proposed to reduce the volume of data transmission when the data is sparse in certain domain. While finding the optimal routing path that minimizes data traffic is an NP-complete problem, a near-optimal routing protocol in the literature requires omniscient knowledge of the entire network and thus incurs extensive message exchanges in real applications. In this paper, we propose a distributed algorithm that uses local minimization to dynamically construct a routing path to reduce the data traffic for compressive sampling based aggregation. This algorithm does not require the omniscient knowledge of the global network topology and incurs much lower overhead than the near optimal solution, and therefore, is more suitable for practical applications.**

***Index Terms*—Compressive sensing, data aggregation, distributed algorithm, routing.**

I. Introduction

In recent years, wireless sensor networks (WSNs) have been used extensively in environment and habitat monitoring [1]-[3]. In large-scale WSNs, hundreds or thousands of sensor nodes are deployed to sense and to transmit data to the sink through multi-hop routes. However, due to their limited battery power, the sensor nodes may quickly run out of power because of the large amount of data they have to relay for other nodes in large-scale WSNs.

The compressive sampling (CS) techniques in [4]-[7] provide a universal sampling scheme without spatial correlation information and promise a full recovery of sensory data if the data are sparse in some domain (e.g. DCT [8]). By incorporating routing and CS, these techniques aggregate data from other nodes to reduce the data traffic. However, it has been proved that finding the optimal routing path that minimizes data traffic in compressive data aggregation is an NP-complete problem [9]. In the literature, there have been some works [9], [10] that provide the near-optimal solutions through linear programming techniques. However, these solutions usually require omniscient information of the entire network topology and assume that the network environment is static. In practical applications, such as ocean monitoring [11], the sensor nodes are mobile during the process of data aggregation. In addition, the sink of wireless sensor networks is usually configured to be mobile to avoid the hot spot problem [12]. When the location of the sink changes, the routing paths derived from these centralized algorithms have to be reconstructed, thus leading to high overhead.

In this paper, we propose a distributed algorithm to efficiently construct a routing path suitable for CS-based aggregation in large-scale WSNs. We aim at minimizing the data traffic during data collection without complete network topology information. The idea of our algorithm is to first construct a data aggregation path by a distributed algorithm, and then apply a local minimization algorithm at each node to reduce the local data traffic. Two distributed algorithms to construct the compressive data aggregation path, namely Shortest Path Tree (SPT) and Minimum Relay Tree (MRT), are considered. SPT transmits data along the shortest path from the each node to the sink, while MRT delivers data using the minimum number of relay nodes. After the aggregation paths are constructed, the local minimization algorithm redirects the data flow at each node to reduce the data traffic.

There are two major contributions in our research. First, our algorithm is a distributed scheme, so no global network topology information is required to find the routing path for compressive data aggregation. Second, our schemes has much lower overhead than the near optimal solution, so it is more suitable for WSNs in practical applications.

The rest of this paper is organized as follows. The existing CS based data aggregation protocols are reviewed in Section II. The concepts of CS theory are introduced in Section III. In Section IV, the SPT, MRT and local minimization algorithm are described in detail. Simulation results are demonstrated and analyzed in Section V. Conclusion and the future work are provided in Section VI.

II. Literature Review

Based on the compressive sensing theory, sensor nodes can compress sensory data without additional prior knowledge and recover these data with a relatively small number of measurements. These features make it appropriate for data aggregation in WSNs and may result in lower transmission cost, shorter acquisition time and less power consumption. These approaches can be classified into three categories: CS technique improvements, CS-based aggregation path optimizations, and applications in WSNs.

Manuscript received April 22, 2013; revised July 25, 2013.

Tsung-Yi Tsai is with the Quanta, Taiwan (e-mail: emilwings@gmail.com).

Wei-Chi Lan and Min-Te Sun are with the Department of Computer Science and Information Engineering, National Central University, Taiwan (e-mail: weichilan@gmail.com, msun@csie.ncu.edu.tw).

Chunlei Liu is with the Department of Mathematics and Computer Science, Valdosta State University, GA 31698 USA (e-mail: cliu@valdosta.edu).

DOI: 10.7763/JACN.2013.V1.59

A. CS Technique Improvements

In [13], a universal compressive wireless sensing (CWS) scheme is proposed for single hop data gathering. The sensor nodes deliver the linear projections of readings to the fusion center through synchronized analog transmissions. Due to the difficulty to apply analog synchronization in large-scale sensor network, CWS is not applicable for multi-hop data gathering. Reference [14] presents the first complete CS-based data gathering scheme in large-scale WSNs. In [14], every sensor node compresses incoming data by summing M coded vector without prior knowledge. Therefore, the traffic load on the aggregation path remains the same regardless of the number of nodes on the path.

Most CS-based data gathering schemes assume that the data transmissions are reliable, but in actual environment the transmissions are usually unreliable. Reference [15] presents an oversampled CS source coding scheme to handle transmission errors. Reference [16] proposes an adaptive data gathering scheme to deal with abnormal readings. In [16], the number of measurements can be adjusted according to the variation of the sensed data. However, neither [15] nor [16] discusses how to reduce the global communications.

B. CS-Based Aggregation Path Optimizations

To reduce unnecessary transmissions at the early stage by directly applying compressed sensing, Reference [17] proposes a hybrid-CS scheme to improve the performance, in which CS-based collection scheme is used only at nodes whose traffic is larger than a threshold. Reference [9] aims at minimizing the energy consumption of the hybrid-CS data gathering scheme. It proves that the optimization problem is NP-complete and gives a greedy algorithm, called MECDA, to grow routing paths from the sink with polynomial time complexity by examining all candidate neighboring nodes one by one at each round. However, when the network topology changes, this solution requires heavy computation to reconstruct the aggregation tree. Reference [10] focuses on finding a routing path that can maximize the lifetime of WSNs and minimize the delay of data gathering. Nevertheless, it also needs global network information and complex computation to create the tree structure.

C. CS Applications in WSNs

A growing number of researches aim to apply CS in other applications in WSNs. Reference [18]–[20] investigate a CS-based approach for target detection and positioning in WSNs. Reference [21] and [22] propose a CS-based data persistence scheme in which sensed data are stored in the network until a mobile sink visits or a data collector query for the readings.

III. PRELIMINARY

In this section, we introduce the main concepts of compression sensing theory and describe how to combine CS with routing in WSNs.

A. Compressive Sampling

Suppose that the measured data from all the network sensors at a given time form an N-dimensional vector $d = [d_1, d_2, \cdots, d_N]^T$. d is considered to be K-sparse if there exists a particular domain Ψ in which d can be represented by K ($K \ll N$) non-zero coefficients:

$$d = \Psi x = \sum_{i=1}^{N} x_i \varphi_i \tag{1}$$

where $\Psi = [\varphi_1, \varphi_2, \ldots, \varphi_N]$ is the representation basis with column vector $\{\varphi_i\}$ and $x = [x_1, x_2, \cdots, x_N]^T$ is the corresponding coefficients.

According to the CS theory, if the sensory data is K-sparse in some domain such as Discrete Cosine Transform (DCT) [8], then it can be reconstructed by solving a programming optimization problem from a small number of measurements. In practical implementation, a sensor node transforms its sensory data by an M -dimensional vector that contains a series of random numbers, and then sends out the summation of this result and the measurements it receives from other nodes. Mathematically, the last measurements obtained by the sink can be represented as

$$y = \Phi d = \sum_{i=1}^{N} \phi_i d_i \tag{2}$$

where $y = [y_1, y_2, \cdots, y_M]^T$ is the sum of measurements and Φ is an $M \times N$ measurement matrix in which each column ϕ_i is the transform vector of the corresponding node. In theory, the measurement matrix Φ should fulfill the Restricted Isometry Property (RIP) [23], and the number of measurements M should satisfy

$$M \geq cK \log \frac{N}{K}, \tag{3}$$

where c is a positive constant Nevertheless, in reality a random measurement matrix is normally used because of convenience, and $M = 3K \sim 4K$ is usually sufficient.

At the end of the compressing and sampling process, the sink can reconstruct the sensor reading d accurately by solving the following minimum ℓ_1 -norm optimization problem

$$x = \arg\min \|x\|_{l_1} \quad s.t. \quad y = \Phi \Psi x. \tag{4}$$

B. Hybrid Compressive Data Gathering

The traditional method to transmit sensed data to the sink is through multi-hop relay as illustrated in Fig. 1(a). Obviously, the closer a sensor node is to the sink, the more readings need to be transmitted. As a result, sensor nodes closer to the sink will run out of energy first. With the compressive data gathering (CDG) approach, as clarified in Fig. 1(b), instead of delivering individual sensor readings, every node applies the CS scheme so that only a few weighted sums of all readings will be sent. In other words, assume that the process uses M sets of different weights in the sensor network with N nodes. According to CS scheme, nodes only relay the same number of M measurements. Hence CDG has the benefit of a balanced load. In additional, because M is usually smaller than N, CDG also produces less global data traffic than traditional methods. To be exact, the overall message complexity of the traditional schemes is $O(N^2)$, but that of CDG is only $O(M N)$.

Nevertheless, blindly applying CS coding on all sensor nodes may lead to unnecessary higher traffic at early stages of transmissions. Therefore, a hybrid CS aggregation [17] technology is introduced to avoid this problem. As illustrated in Fig. 1(c) , a sensor node applies CS coding only when it needs to deliver more than M samples. Otherwise, the traditional method is used. The nodes that apply the CS coding scheme are called aggregators. Each aggregator generates a series of coefficients using the seed produced from the sink and the node id of the received samples. After that, the aggregator uses the coefficient to encode received samples and combine them into one vector through summation. Finally, the aggregator will send out the encoded data of the exact size M. The hybrid CS aggregation mechanism combines the traditional scheme and the CS-based scheme to reduce the overall traffic load at early stages of the transmission. However, the performance of the hybrid CS aggregation mechanism is highly dependent to the routing path. To the best of our knowledge, there have been little prior works [9], [10] that provide the solution which needs the entire network topology information and complex computing to construct the optimal routing path.

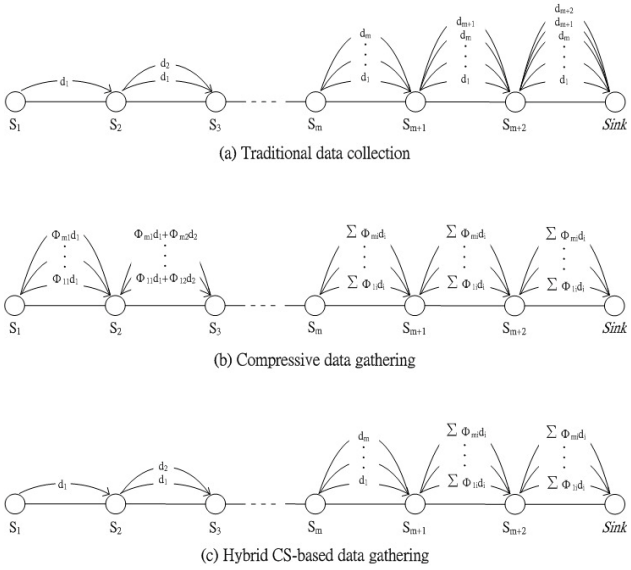


Fig. 1. Comparison of different data collection mechanisms.

IV. Main Idea

The problem we seek to address is to minimize the overall data traffic in hybrid compressed data aggregation with low construction overhead. We assume the data aggregation proceeds in rounds by properly scheduling the network, and no transmission error occurs when applying the oversampled CS source coding scheme. Consequently, the routing path in the data gathering process will form an aggregation tree rooted at the sink. Besides, we assume that a node becomes an aggregator when the size of data it relays is larger than M.

Our compressive data aggregation algorithm consists of two steps. The first step is to construct the data aggregation tree distributively, and the second step is to reduce the data traffic by adjusting the routing path locally. In this section, we will introduce the two options to construct the data aggregation tree and the local minimization to reduce the data traffic. The notations used in this paper are summarized in Table I.

TABLE I: Definition of Notations

Symbol	Definition
dist(x)	The shortest path distance from node *x* to the sink
edge_dist(x,y)	Edge distance between neighboring nodes *x* and *y*
MSC(x)	The minimum set of relay nodes for node *x*
N(x)	The one-hop neighbor set of node *x*
$N^2(x)$	The two-hop neighbor set of node *x*

A. CS Data Aggregation Trees

1) The shortest path tree construction

If we consider nodes in a WSN as vertices and each wireless link between two neighboring nodes as an edge, the WSN can be naturally regarded as a graph. Its Shortest Path Tree (SPT) is composed of shortest paths from the sink to all nodes in a WSN. Although SPT is not unique for a WSN, it can be constructed efficiently in rounds distributively by Dijkstra's algorithm or Bellman-Ford algorithm [24].

Let $edge\_dist(x,y)$ denote the edge distance, i.e. the straight-line distance, between any two neighboring nodes x and y. The first phase of constructing the SPT is to compute the shortest path distance from all nodes to the sink, which can be conducted in the following steps.

1) Assign the distance 0 to the sink, and a tentative distance of infinity to all other nodes.
2) Select the sink as the current node x, mark all other nodes as *unvisited*, and create the *unvisited set* as the set of all *unvisited* nodes.
3) For each *unvisited* neighbor y of x, if $dist(x)+edge\_dist(x,y)$ is less than y's previously recorded tentative distance, then overwrite y's tentative distance as $dist(x)+edge\_dist(x,y)$.
4) After all neighbors of the current node are considered, mark the current node as *visited*, remove it from the *unvisited set*, and record its tentative distance as the shortest path distance $dist(x)$.
5) If the *unvisited set* is empty or the smallest tentative distance among all nodes in the unvisited set is infinity, stop. Otherwise, choose the node with the smallest tentative distance as the next current node and go to Step 3.

The second phase of constructing the SPT is to find the edges for the short path tree. After the shortest path distance of all nodes is found, we can assign a parent p_x to each non-sink vertex x such that p_x is connected to x and

$$dist(p_x)+edge\_dist(p_x,x)=dist(x).$$

In the case that multiple choices for p_x exist, choose the p_x for which there exists a shortest path from the sink to p_x with the smallest number of edges. After the parent of all non- sink nodes is determined, the short path tree consists of edges between all nodes and their parent.

2) The minimum relay tree construction

On the other hand, based on the distributed approach that broadcasts messages with minimum relay nodes in [25], we propose a similar strategy to collect sensory data with minimum relay nodes. This strategy uses a distributed algorithm to locally determine the minimum collection set of neighbors

for each node x, denoted as $MCS(x)$, such that the union of the neighbors of nodes in the set covers all two-hop neighbors of x. In other words, $MCS(x) \subseteq N(x)$ and $N^2(x) = \cup_{i \in MCS(x)} N(i)$. Since it is a minimum set to cover $N^2(x)$, $MCS(x)$ should be able to collect and aggregate the sensory data from two-hop neighbors efficiently.

Before constructing the minimum relay tree, every node is required to collect the two-hop neighbors' information. A *LINK* message with an *EXECUTION* flag is used to notify a node to set up its parent and to determine whether to execute the tree construction algorithm or not. When a node x receives a *LINK* message for the first time, it sets its parent as the sender of the *LINK* message. If the *EXECUTION* flag in the message is true, it also performs the following steps.

1) Node x adds the nodes that belong to $N(x)$ and are the only neighbor of some nodes in $N^2(x)$ to $MCS(x)$, and then send a *LINK* message with *EXECUTION* = true to those nodes.
2) While there still exist a node in $N^2(x)$ that is not covered by $MCS(x)$:
   a) For each node in $N(x)$ that is not in $MCS(x)$, compute the number of nodes that it covers among the uncovered nodes in $N^2(x)$.
   b) Add the node with the largest coverage number to $MCS(x)$ and send a *LINK* message with *EXECUTION* = true to that node.
3) Node x sends a LINK message with EXECUTION = false to all neighbors that are not in $MCS(x)$.

After the construction of the SPT or MRT aggregation tree, each node decides to compress the received data or not locally. If the data received plus the data sensed by a node is larger than M, the node compresses these data according to compressive sampling theory [14] and forwards the compressed data to its parent. With fewer relay nodes, the degree of inner nodes in this aggregation tree is maximized and data can converge quickly. As a result, inner nodes can more easily reach the lower bound of compression and hence become an aggregator.

B. Local Minimization

After the aggregation tree is constructed, a distributed local minimization (LM) algorithm is used to calculate whether switching to a different neighbor can reduce the data traffic. Each node performs the following steps.

1) Every node x collects the size of received data and the parent's ID of its two-hop neighbors by exchanging INFO messages with its neighbors.
2) For every non-child neighbor Ni with an equal or lower hop distance to the sink, x measures the change of local data traffic if x switches its parent to Ni. The original local traffic is the sum of the output of node x, Ni and their parents. The new local traffic is calculated similarly, assuming node x is redirected to Ni. In the case where Ni and x have the same parent, the parent's traffic is counted only once. If the new traffic is less than the original traffic, x records the size of reduced traffic and the neighbor's ID.
3) After the measurement of all neighbors is completed, if node x is not locked, x selects the neighbor y that minimizes the local traffic, and sends it a *LOCK* message to prevent it from updating its parent. Otherwise, node x defers its action until it is unlocked by an *UPDATE* message to continue. After the *LOCK* message is acknowledged by y (i.e., y is locked successfully), x updates its parent to y, and then broadcasts an *UPDATE* message to notify its neighbors so that they can run this algorithm with the updated information. The *UPDATE* message also unlocks y so that it can continue its own calculation and update. If x does not receive an acknowledgment from y after a period of time, it waits for a random period of time and sends a *LOCK* message to y again.

Note that the *LOCK*, *LOCK* acknowledgment, and *UPDATE* messages in the local minimization algorithm are used to avoid deadlocks and the race condition where neighboring nodes update their parent simultaneously and thus result in the disconnection of the aggregation tree.

V. Performance Analysis

To analyze the performance of the proposed algorithms, we implement them with ns-2 simulator [26] and compare them with other protocols, including SPT, MRT and the near optimal solution MECDA [9]. Three metrics used are to evaluate the performance. The global data traffic is the sum of outgoing traffic of all sensor nodes in the CS-based aggregation process. This metric shows how effective the protocol is. The overhead is the number of messages that a protocol needs to transmit for the construction of the aggregation path. The average energy consumption of top 5% nodes is used to estimate the network lifetime. Note that the energy consumption for data transmission and reception at sensor nodes is several magnitudes higher than the other tasks, such as compression, computation and waiting. Consequently, in the simulation only the energy spent on data communications is considered. In addition, the energy consumption for each data transmission and reception is assumed to be the same, so the energy consumption for a node can be represented by the number of packets it has transmitted and received.

We simulate different network densities by changing the size of simulation region while maintaining a fixed number of 1000 sensor nodes. For each network density, 20 different topologies are generated and simulated. We evaluate the performance with compression threshold ranging from 100 to 200 bits. The main parameters used in the simulation are listed in Table II.

TABLE II: Simulation Parameters

Parameter	Value
Mac protocol	802.11
Number of nodes	1000
Node deployment	Random
Radio propagation model	Two-ray ground
Radio Transmission range	200 m

A. Global Data Traffic Comparison

In this subsection, we compare the global data traffic of each protocol with density=7 neighbors per node. To understand the improvement of the local minimization

algorithm, protocols with or without LM are studied, with the exception of MECDA where LM does make any improvement.

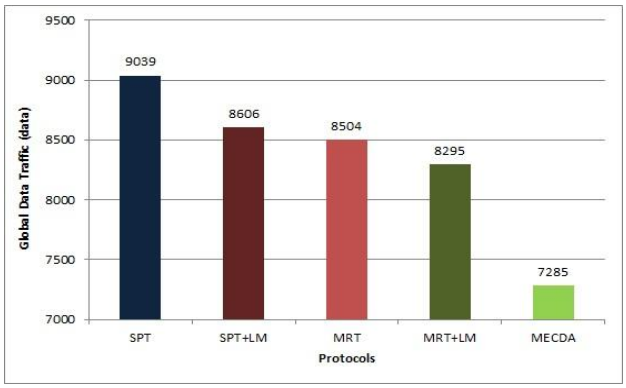


Fig. 2. Global data traffic of different protocols

The results for global data traffic are depicted in Fig. 2. MRT has smaller data traffic than SPT because the aggregators receive more data from relay nodes and thus reduce the data traffic. Fig. 2 also shows that local minimization reduces the global data traffic of each protocol by approximately 2 ~ 5%. Compared with the global data traffic of near optimal solution MECDA, MRT with local minimization is 14% higher and SPT with local minimization is 18% higher.

B. *Overhead Comparison*

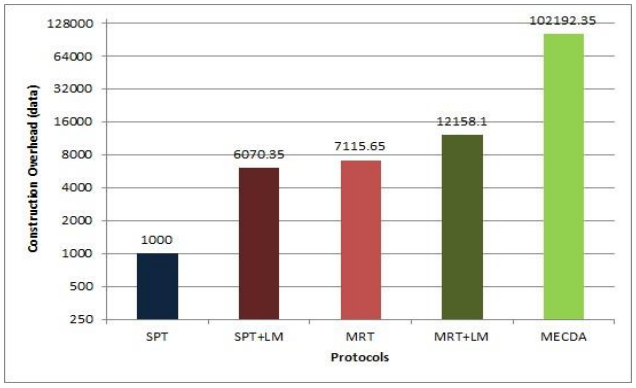


Fig. 3. Construction overhead of different protocols

The results for tree construction overhead are illustrated in Fig. 3. SPT has the lowest construction overhead because the shortest routing path can be constructed by broadcasting a visit message from the sink. Both MRT and local minimization algorithms need to collect two-hop neighbor information. How- ever, in MRT each node needs to send additional messages to construct the routing path, while in the local minimization algorithm, a node sends an update message only when there is a change in the data flow. Finally, due to the need to collect topology information from entire network to compute the near optimal solution, the overhead of MECDA is ten times higher than other protocols.

C. *Effect of Different Network Density*

In networks with higher density, sensor nodes are closer to the sink so the number of data relays is reduced. As shown in Fig. 4, the average global data traffic of all protocols is reduced. In Fig. 5, for the same reason, the average energy consumption of top 5% nodes of all protocols is also reduced. For networks with higher density, the energy consumption reduction of MRT is smaller than other protocols because the average load of its relay nodes increases as the density increases.

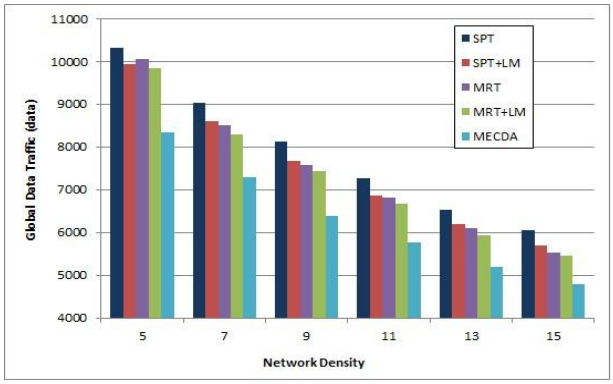


Fig. 4. Global data traffic for different density

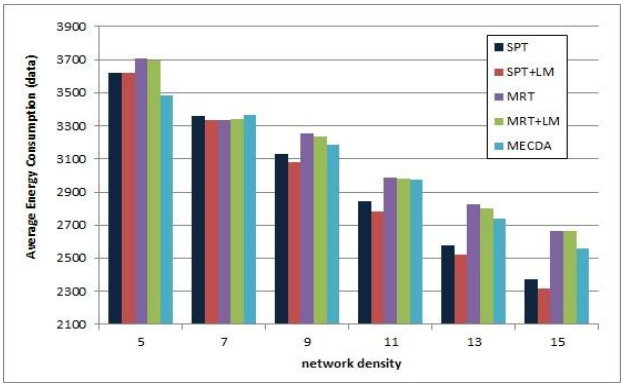


Fig. 5. Average energy consumption for different density

D. *Effect of Different Compression Threshold*

In this subsection, we consider networks with density = 7 neighbors per node and the compression threshold between 100 and 200. Fig. 6 shows that the global data traffic of all protocols increases with the compression threshold. When the compression threshold increases, fewer nodes become aggregators so the network has to relay more data without aggregation. The average energy consumption, as shown in Fig. 7, also increases with the threshold because the aggregators have to compress and deliver more data. In general, a value of compression threshold between 15% and 20% of the total number of nodes as used in [14] usually allows a satisfactory data recovery.

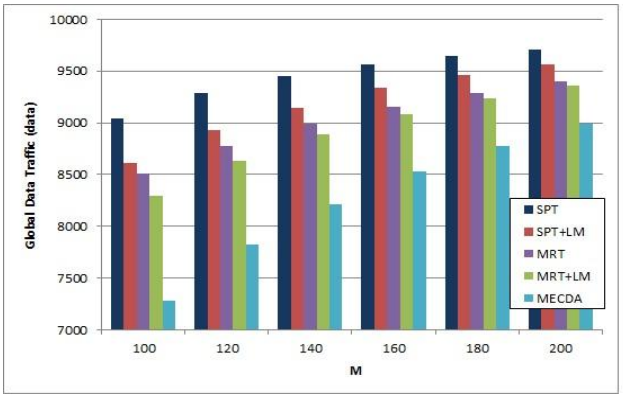


Fig. 6. Global data traffic for different compression threshold

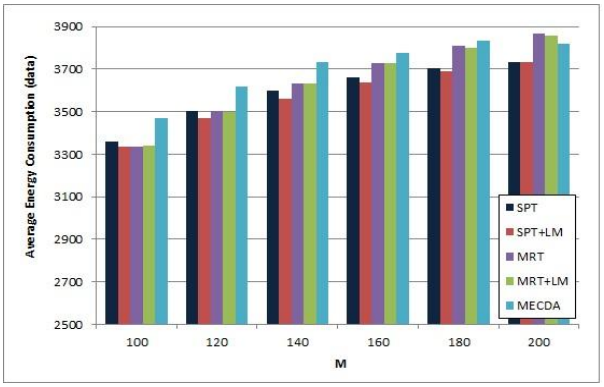


Fig. 7. Average energy consumption for different compression threshold

VI. Conclusion and Future Work

The MECDA protocol [9] is known to be energy efficient in compressive data aggregation, but it needs the complete network topology information to compute the near optimal solution. With such a large overhead, MECDA is not suitable for practical applications. The method we propose in this paper is a distributed scheme to construct the routing path for compressive data aggregation. Simulation results show that our algorithm significantly outperforms MECDA in overhead with slightly higher data traffic.

From the simulation results, we can observe that the tree structure has a significant impact on the efficiency of compressive data aggregation. In the future, we would like to explore the uses of network skeletons for compressive data aggregation. In addition, it is also interesting to consider the impact of multiple sinks to compressive data aggregation in large-scale wireless sensor networks.

References

[1] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An Analysis of a Large Scale Habitat Monitoring Application," in *Proc. of ACM SenSys'04*, 2004, pp. 214-226.

[2] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A Wireless Sensor Network for Structural Monitoring," in *Proc. of ACM SenSys'04,* 2004, pp. 13-24.

[3] C. W. Chen and Y. Wang, "Chain-Type wireless sensor network for monitoring long range infrastructures: architecture and protocols," *Int. J. Distrib. Sen. Netw*., vol. 4, no. 4, pp. 287-314, Oct. 2008.

[4] D. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theor*., vol. 52, no. 4, pp. 1289-1306, Apr. 2006.

[5] R. Baraniuk, "Compressive sensing," *IEEE Signal Processing Magazine*, vol. 24, pp. 118-121, Jul. 2007.

[6] E. Candes and M. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, pp. 21-30, Mar. 2008.

[7] J. Haupt, W. Bajwa, M. Rabbat, and R. Nowak, "Compressed sensing for networked data," *IEEE Signal Processing Magazine*, vol. 25, pp. 92-101, Mar. 2008.

[8] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput*., vol. 23, no. 1, pp. 90-93, Jan. 1974.

[9] L. Xiang, J. Luo, and A. V. Vasilakos, "Compressed data aggregation for energy efficient wireless sensor networks," in *Proc. of IEEE SECON'11*, 2011, pp. 46-54.

[10] S. Mehrjoo, J. Shanbehzadeh, and M. Pedram, "A Novel Intelligent Energy-Efficient Delay-Aware Routing in WSN, Based on Compressive Sensing," in *Proc. of IEEE IST'10*, 2010.

[11] F. Reichenbach, M. Handy, and D. Timmermann, "Monitoring the ocean environment with large-area wireless sensor networks," in *Proc. of EUROMICRO'05*, 2005.

[12] M. Perillo, Z. Cheng, and W. Heinzelman, "An analysis of strategies for mitigating the sensor network hot spot problem," in *Proc. of ACM MOBIQUITOUS'05,* 2005, pp. 474-478.

[13] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Compressive wireless sensing," in *Proc. of ACM IPSN'06*, 2006, pp. 134-142.

[14] C. Luo, F. Wu, J. Sun, and C. W. Chen, "Compressive data gathering for large-scale wireless sensor networks," in *Proc. of ACM MobiCom'09*, 2009, pp. 145-156.

[15] Z. Charbiwala, S. Chakraborty, S. Zahedi, Y. Kim, M. B. Srivastava, T. He, and C. Bisdikian, "Compressive oversampling for robust data transmission in sensor networks," in *Proc. of IEEE INFOCOM'10*, 2010, pp. 1190-1198.

[16] J. Wang, S. Tang, B. Yin, and X.-Y. Li, "Data gathering in wireless sensor networks through intelligent compressive sensing," in *Proc. of IEEE INFOCOM'12*, 2012, pp. 603-611.

[17] J. Luo, L. Xiang, and C. Rosenberg, "Does compressed sensing improve the throughput of wireless sensor networks?" in *Proc. of IEEE ICC'10*, 2010, pp. 1-6.

[18] B. Zhang, X. Cheng, N. Zhang, Y. Cui, Y. Li, and Q. Liang, "Sparse Target counting and localization in sensor networks based on compressive sensing," in *Proc. of IEEE INFOCOM'11*, 2011, pp. 2255-2263.

[19] J. J. Meng, H. Li, and Z. Han, "Sparse event detection in wireless sensor networks using compressive sensing," in *Proc. of CISS'09*, 2009, pp. 181-185.

[20] H. Zheng, S. Xiao, and X. Wang, "Sequential compressive target detection in wireless sensor networks," in *Proc. of IEEE ICC'11*, 2011, pp. 1-5.

[21] M. Lin, C. Luo, F. Liu, and F. Wu, "Compressive data persistence in large-scale wireless sensor networks," in *Proc. of IEEE GLOBECOM'10*, 2010.

[22] A. Talari and N. Rahnavard, "CStorage: distributed data storage in wireless sensor networks employing compressive sensing," in *Proc. of IEEE GLOBECOM'11*, 2011.

[23] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theor*., vol. 51, no. 12, pp. 4203-4215, Dec. 2005.

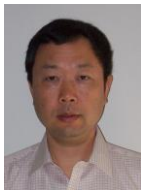
[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, Third Edition. The MIT Press, 2009.

[25] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint Relaying for Flooding Broadcast Messages in Mobile Wireless Networks," in *Proc. of HICSS'02*, 2002, pp. 3866-3875.

[26] The Network Simulator NS-2. [Online]. Available: http://www.isi.edu/nsnam/ns/.

Tsung-Yi Tsai received his M.S. degree in Department of Computer Science and Information Engineering at National Central University of Taiwan in 2012. His research interests include wireless sensor networks and wireless ad-hoc networks. He is currently with Quanta as an engineer.

Wei-Chi Lan is a Ph.D. student in Department of Computer Science and Information Engineering at National Central University, Taiwan. He received his B.S. degree in Department of Computer Science and Information Engineering from National Dong Hwa University of Taiwan in 2007, and M.S. degree in Department of Computer Science and Information Engineering at National Central University of Taiwan in 2011. His research interests involve wireless sensor ad-hoc networks.



Chunlei Liu received his B.S and M.S. in Computational Mathematics from Wuhan University, China and his Ph.D. in Computer and Information Science from The Ohio State University. He is now an Associate Professor in the Department of Mathematics and Computer Science at Valdosta State University. His research interest is in computer networks and scientific computing.

Min-Te Sun received his B.S. degree in Mathematics from National Taiwan University, the M.S. degree in Computer Science from Indiana University, and the Ph.D. degree in Computer and Information Science from the Ohio State University. Since 2008, he has been with Department of Computer Science and Information Engineering at National Central University, Taiwan. His research interests include distributed algorithm design and wireless network protocol engineering.