Scheduling Data Collection with Dynamic Traffic Patterns in Wireless Sensor Networks

Wenbo Zhao and Xueyan Tang
School of Computer Engineering, Nanyang Technological University, Singapore 639798
Email: {zhao0101, asxytang}@ntu.edu.sg

***Abstract*—The network traffic pattern of continuous sensor data collection often changes constantly over time due to the exploitation of temporal and spatial data correlations as well as the nature of condition-based monitoring applications. This paper develops a novel TDMA schedule that is capable of efficiently collecting sensor data for any network traffic pattern and is thus well suited to continuous data collection with dynamic traffic patterns. Following this schedule, the energy consumed by sensor nodes for any traffic pattern is very close to the minimum required by their workloads given in the traffic pattern. The schedule also allows the base station to conclude data collection as early as possible according to the traffic load, thereby reducing the latency of data collection. Experimental results using real-world data traces show that, compared with existing schedules that are targeted on a fixed traffic pattern, our proposed schedule significantly improves the energy efficiency and time efficiency of sensor data collection with dynamic traffic patterns.**

I. Introduction

Energy efficiency and time efficiency are two major considerations for sensor data collection in wireless sensor networks. Energy efficiency concerns the amount of energy spent in data collection. Since sensor nodes are normally powered by batteries, it is critical to conserve energy as much as possible to extend the lifetime of a sensor network [1], [2]. Time efficiency, on the other hand, refers to the latency of collecting data from sensor nodes to a base station (or a sink node). Sensor data are often required to be quickly gathered after acquisition for timely processing [1].

TDMA is a promising MAC protocol for efficient data collection in wireless sensor networks [3]–[5]. TDMA is contention-free and eliminates collisions by scheduling only non-interfering transmissions to proceed in the same time slot. It avoids the energy cost and latency overhead required by contention-based MAC protocols to compete for channel access and to perform retransmissions upon collisions. In addition, TDMA allows sensor nodes to turn their radios off whenever they are not transmitting or receiving, further conserving energy at sensor nodes.

Most existing TDMA schedules are constructed for a static network traffic pattern in which a fixed set of nodes report data to the base station at each sampling interval [3]–[5]. In practice, however, continuous sensor data collection often exhibits dynamically changing network traffic patterns over time due to energy conservation concerns and the nature of monitoring applications. For example:

- To save energy, temporal and spatial correlations among sensor measurements are usually exploited to reduce the amount of data that need to be collected [6]. In this approach, a sensor node updates new measurements with the base station only when the new measurements differ considerably from past ones or their correlations with other measurements in the vicinity change substantially. As a result, the set of reporting nodes normally changes from one sampling interval to another.
- Condition-based monitoring is widely used in surveillance applications such as active volcano detection [1] and structural health monitoring [7]. In these applications, only sensor measurements satisfying certain conditions need to be reported to the base station. For instance, to monitor volcanic activities, only the spikes of seismic and infrasonic signals need to be collected [1]. Thus, the set of sensor nodes that report to the base station is expected to vary over different sampling intervals.

The above scenarios of continuous data collection share the common characteristic that the network traffic pattern changes over different sampling intervals in a largely unpredictable manner. One possible approach to cope with dynamic traffic patterns is to construct and deploy a new TDMA schedule tailored to the new traffic pattern whenever the traffic pattern changes. However, identifying new traffic patterns and disseminating new schedules over the network both require sensor nodes to communicate with each other, which introduces extra energy consumption and latency overheads. When the traffic pattern changes frequently, the overheads are very likely to cancel out or even outweigh the benefits of deploying new schedules [4]. *Therefore, it is highly desirable to consistently use a single TDMA schedule that is able to efficiently handle a wide variety of traffic patterns.*

This paper develops a novel TDMA schedule that achieves high energy efficiency and time efficiency for any traffic pattern and is thus well suited to the needs of continuous data collection with dynamic traffic patterns. Following this schedule, the energy consumed by each node self-adapts to its required workload given in any traffic pattern. The scheduling algorithm also incorporates effective strategies for letting the base station conclude data collection as early as possible according to the traffic load, thereby reducing the latency of data collection. Experimental results using real-world data traces show that, compared with existing schedules, our proposed schedule significantly reduces the energy cost and latency of sensor data collection with dynamic traffic patterns.

II. Traffic Pattern Oblivious Scheduling

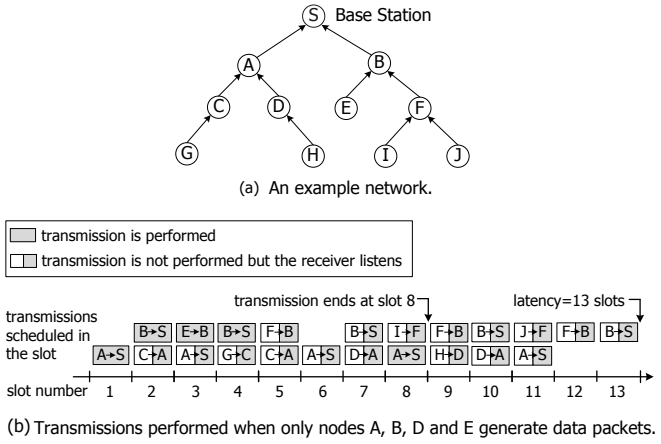
A. System Model

We consider a network of sensor nodes that periodically sample local phenomena like temperature and solar radiation. The data acquired by all sensor nodes are continuously collected by a base station (or a sink node). Following common practices [8], we assume that the sensor nodes are organized into a tree structure rooted at the base station for data collection. Due to various reasons as described in the introduction, a sensor node may not report its acquired data to the base station at every sampling interval. The data reported by each sensor node in a sampling interval, if any, fits into one packet and the packets generated by different sensor nodes are not aggregated on their ways toward the base station. Similar to other studies [3]–[5], clocks are assumed to be synchronized among sensor nodes. Time is divided into slots and the duration of a time slot allows a sensor node to transmit only one packet.

To avoid collisions, only transmissions that do not conflict with each other are allowed to be scheduled in the same time slot. For example, for sensor nodes equipped with single omnidirectional transceivers, conflict-free scheduling normally requires that a node cannot transmit and receive simultaneously, and for a node to successfully receive the transmission of another node, no other neighbor of the receiving node should transmit at the same time [3]–[5].

B. A Motivating Example

We start with an example illustrating how a traditional TDMA schedule constructed for a static traffic pattern introduces idle listening and unnecessary delay under dynamic traffic patterns. Fig. 1(a) shows a network of 10 sensor nodes organized into a tree structure for data collection. For simplicity, in this example, we assume that the transmission from each node to its parent conflicts with only the transmissions of its siblings and grandparent in the tree. Fig. 1(b) shows a typical schedule [3], [4] targeted on the full traffic pattern in which each node generates one data packet to send to the base station. In the full traffic pattern, the base station receives the last packet in slot 13 (from node B). So, the latency of data collection is 13 time slots. In the data collection process, node B listens to transmissions in four time slots (slots 3, 5, 9 and 12) and transmits in five time slots (slots 2, 4, 7, 10 and 13).



(a) An example network.

(b) Transmissions performed when only nodes A, B, D and E generate data packets.

Fig. 1. Inefficiency of a schedule targeted on a static network traffic pattern.

Now suppose the traffic pattern changes such that only nodes A, B, D and E generate data packets to send, and the change is not known a priori to any node as well as the base station. Using the same schedule, actual transmissions occur only in the fully shaded rectangles shown in Fig. 1(b). Although the base station has received all the four data packets by the end of slot 8, it has to continue listening for possible transmissions in slots 10, 11 and 13. This is because packets generated by nodes I, H and J, if any, are scheduled to be forwarded to the base station in slots 10, 11 and 13 respectively by nodes B, A and B. Since the base station does not know beforehand whether nodes I, H and J generate any data packet or not, it has to listen in these slots in order not to miss any potential transmission. The base station cannot make certain that all data have been received until the end of slot 13. Therefore, the latency of data collection remains to be 13 time slots. For node B, it receives only one packet from node E in slot 3. Similar to the base station, since B is not aware of which descendants generate data packets, B still has to listen for possible transmissions in all the four slots 3, 5, 9 and 12, the last three of which result in idle listening as shown by half shaded rectangles in Fig. 1(b). Similar observations can also be made for other nodes in the network.

C. Reducing Idle Listening and Latency

Our key strategy for reducing idle listening and latency is to allow each node to transmit all data in its successive transmission slots starting from its first transmission slot, irrespective of the traffic pattern. In this way, if a node does not send out any packet in a scheduled transmission slot, the node will leave all of its subsequent transmission slots idle as well. Therefore, on observing an idle transmission slot of the node, its parent is assured that no more data will come from the node and thus can safely turn off its radio in all subsequent transmission slots of the node to avoid idle listening.

By applying the above strategy to all nodes in the network, a parent node listens to each child for at most one more slot than the actual number of transmission slots used by the child to send packets. Therefore, the energy consumed by sensor nodes for any traffic pattern is very close to the minimum required by their workloads given in the traffic pattern. For the base station, on observing an idle transmission slot of a child node, the base station can infer that the child node has finished transmitting all data. Therefore, instead of listening for possible transmissions till the end of the schedule, the base station concludes data collection when it infers that all of its child nodes have finished transmissions.

To illustrate our strategy, consider the example network of Fig. 1(a). Suppose that node B and its children's transmission slots are arranged in temporal order as shown in Fig. 2. In this order, B's first transmission slot is scheduled after E and F's first transmission slots, and B's second transmission slot

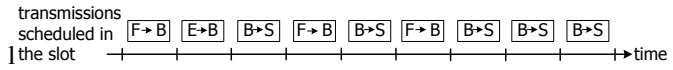


Fig. 2. A temporal order of transmission slots that allows node B to transmit all data in its successive transmission slots.

is scheduled after F's second transmission slot. Then, node B is able to transmit all data in its successive transmission slots irrespective of the traffic pattern, provided that nodes E and F both transmit data in their successive transmission slots starting from their first transmission slots. This is because based on whether E and F's first transmission slots are idle or not, node B would have learned whether E and F have any packet to forward to it. In addition, node B certainly knows whether itself generates any packet to send to the base station. Thus, prior to node B's first transmission slot, B is already aware of whether it needs to send at least one packet to its parent or not (but B may not know the total number of packets to send to its parent). If at least one packet needs to be sent, B would send out a packet in its first transmission slot, which may be any packet available at B (i.e., either the packet generated by B or the packet received from E or F in their first transmission slots). Otherwise, B leaves its first transmission slot idle and will not have anything to send in all subsequent transmission slots either. Similar arguments apply to B's second transmission slot as well. Therefore, node B is committed to transmit all data in its successive transmission slots starting from its first transmission slot.

Let T_v be the subtree rooted at a node v, and $|T_v|$ be the size of T_v. In general, we have the following *necessary and sufficient condition* for enabling a node v to transmit all data in its successive transmission slots starting from its first transmission slot, assuming that all of its children do so. Due to space limitations, the proof is omitted here.

Condition $\mathcal{S}$: for each $1 \leq i \leq |T_v|$ and each child c of node v, the ith transmission slot of node v should be scheduled after the ith transmission slot of node c if $|T_c| > i$, and after all transmission slots of node c if $|T_c| \leq i$.

It is easy to verify that the temporal order of transmission slots in Fig. 2 satisfies condition $\mathcal{S}$ for node B. Exploiting condition $\mathcal{S}$ in scheduling would enable each node to transmit all data in its successive transmission slots so as to reduce idle listening for any traffic pattern and create opportunities for the base station to conclude data collection sooner.

D. Our Proposed Scheduling Algorithm

Our proposed scheduling algorithm is called TPO (Traffic Pattern Oblivious) in that the constructed schedule effectively deals with any network traffic pattern. Algorithm 1 presents the pseudo code of the TPO scheduling algorithm, where T is the routing tree for data collection, $C(v)$ denotes the set of node v's children in the tree, T_v stands for the subtree rooted at node v, $|T_v|$ denotes the size of T_v, and $S(t)$ represents the set of nodes that are assigned slot t as their transmission slots. In this algorithm, two counters are maintained for each node: the `toschedule` counter records the number of transmission slots allowed to be assigned to the node, and the `scheduled` counter records the number of transmission slots that have already been assigned to the node. Initially, only leaf nodes can be assigned transmission slots: their `toschedule` counters are set to 1. The `toschedule` counters of non-leaf nodes are set to 0. In addition, the `scheduled` counters of all nodes are set to 0. These are shown in steps 1 to 6 of Algorithm 1.

Algorithm 1: TPO Scheduling Algorithm

```
1  for each node v ∈ T do
2      v.scheduled = 0;
3      if C(v) = ∅ then
4          v.toschedule = 1;
5      else
6          v.toschedule = 0;
7  t = 1;
8  E = {v | v.toschedule > v.scheduled};
9  while E ≠ ∅ do
10     S(t) = ∅;
11     for each node v in E do
12         if v does not conflict with any node in S(t) then
13             S(t) = S(t) ∪ {v};
14             v.scheduled = v.scheduled + 1;
15             let p be v's parent;
16             if ∀c ∈ C(p), c.scheduled = |T_c| then
17                 p.toschedule = |T_p|;
18             else
19                 p.toschedule = min{c.scheduled | c ∈
                   C(p), c.scheduled<|T_c|};
20     E = {v | v.toschedule > v.scheduled};
21     t = t + 1;
```

The algorithm schedules one time slot at a time. For each time slot, only the nodes with `toschedule` counters greater than `scheduled` counters are eligible to be assigned the time slot as their transmission slots. These nodes are maintained by an eligible set E (steps 8 and 20). Each time slot is scheduled in a greedy manner by examining all eligible nodes (step 11). If an eligible node v does not conflict with any other node already scheduled for transmission in the current time slot, v is assigned the current time slot as a transmission slot and its `scheduled` counter is incremented (steps 12 to 14).

Condition $\mathcal{S}$ implies that at any time, a non-leaf node cannot be assigned more transmission slots than any of its children except those children that have finished scheduling all their transmission slots. Therefore, to make use of condition $\mathcal{S}$, when a node v is assigned a new time slot, the `toschedule` counter of its parent p is updated as follows:

$$p.\texttt{toschedule} = \begin{cases} \min\{c.\texttt{scheduled} \mid c \in C(p),\ c.\texttt{scheduled} < |T_c|\} \\ \qquad\qquad \text{if } \exists c \in C(p),\ c.\texttt{scheduled} < |T_c|, \\ |T_p| \qquad\qquad \text{if } \forall c \in C(p),\ c.\texttt{scheduled} = |T_c|, \end{cases}$$

where $C(p)$ is the set of p's children. Note that $|T_p|$ is the maximum possible number of transmission slots required by

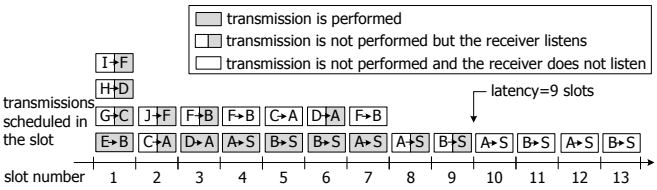


Fig. 3. The TPO schedule constructed for the network of Fig. 1(a).

node p when every node in subtree T_p generates one packet to send to the base station. The above update is performed in steps 15 to 19. The algorithm completes when no more node is eligible for scheduling (step 9).

A schedule constructed by the TPO algorithm is to be consistently used throughout the duration of continuous data collection, irrespective of the traffic pattern. Thus, the scheduling overhead, amortized over the duration of continuous data collection, would be minimal even if the TPO algorithm is implemented in a centralized manner. We are currently investigating a distributed implementation of the TPO algorithm.

Fig. 3 shows the schedule constructed by our TPO algorithm for the network of Fig. 1(a) and how the schedule handles the traffic pattern in which only nodes A, B, D and E generate packets to send to the base station. The fully shaded rectangles are the actual transmissions that take place; the half shaded rectangles represent that the sending node does not perform transmission, but the receiving node listens for transmission in the time slot; the non-shaded rectangles mean that the sending node does not transmit and the receiving node does not listen either. As seen from Fig. 3, since nodes F, I and J do not generate packets, node F leaves all its transmission slots (slots 3, 4 and 7) idle. Node B, on observing that the slot 3 is idle, turns off its radio in slots 4 and 7. Thus, including listening to node E in slot 1, node B listens for a total of two slots only. As a result, the schedule of Fig. 3 reduces the number of time slots in which node B listens by half (from four slots down to two slots) compared to the schedule of Fig. 1(b). In addition, the base station infers that nodes A and B have finished transmissions respectively when it observes that slots 8 and 9 are idle. So, the base station concludes data collection at the end of slot 9. Therefore, the latency of data collection is four slots shorter than that in the schedule of Fig. 1(b).

III. Performance Evaluation

A. Experimental Setup

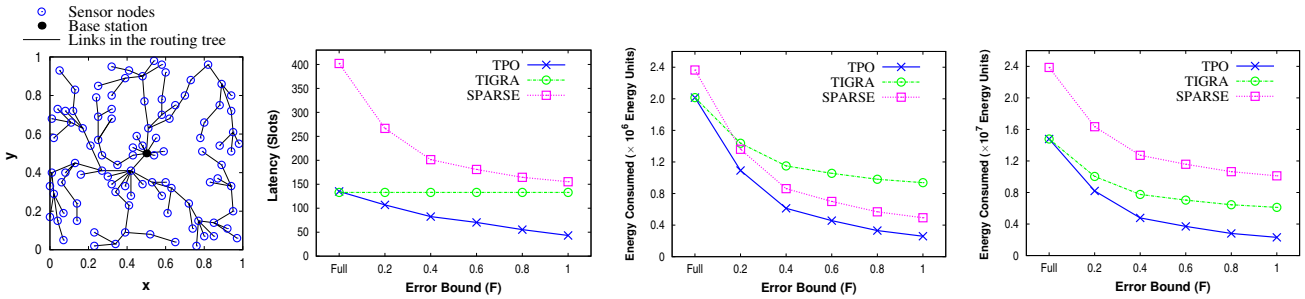
We developed a simulator to evaluate the proposed TPO scheduling algorithm. We simulated a network of sensor nodes randomly deployed over a square field with the base station located at the center. A breadth first search tree rooted at the base station was constructed and used as the routing tree for data collection [3], [4]. The following conflict constraints were placed on scheduling: when a node is scheduled to receive data from another node, no other neighbor of the receiving node is allowed to be scheduled for transmission in the same time slot [3]–[5]. Due to space limitations, we shall only report the representative results for a sample network topology of 100 nodes placed over a 1×1 field as shown in Fig. 4(a), in which the transmission range of sensor nodes was set at 0.15.

We made use of the temperature data trace provided by the LEM project [9] in our simulation. We extracted a large number of subtraces and associated different subtraces with different sensor nodes in the simulated network. Each subtrace contained 20,000 readings.

To simulate dynamic network traffic patterns, we implemented error-bounded approximate data collection [6], in which the base station would like to be assured that its knowledge of sensor readings are always kept within a required error bound e from the exact sensor readings. To suppress unnecessary updates, each sensor node maintains a filter window $[u-e, u+e]$ centered at the reading u that it last updated with the base station. At each sampling interval, if the new reading is beyond the filter window, the sensor node reports the new reading to the base station and updates its filter window. It is intuitive that the average proportion of sensor nodes reporting data to the base station at a sampling interval decreases with an increasing error bound. In addition to approximate data collection, we also simulated exact data collection with a full traffic pattern (denoted by "Full" in the performance results) in which all sensor nodes report their readings to the base station at every sampling interval.

For comparison purposes, we also implemented two state-of-the-art scheduling algorithms: TIGRA [5] and SPARSE [4]. TIGRA uses a graph coloring algorithm to construct a latency-optimized schedule for the aforementioned full traffic pattern. SPARSE aims to build a minimum-latency schedule for a given network traffic pattern. To identify the traffic pattern and construct the schedule, messages must flow over the entire routing tree to complete an in-order tree traversal and one top-down pass from the base station to all sensor nodes [4]. In our simulation of SPARSE, at each sampling interval, a new schedule tailored to the network traffic pattern of that sampling interval was constructed prior to data collection. In contrast, TIGRA and our TPO algorithm consistently use a single schedule throughout all sampling intervals.

Each simulation run was performed for 20,000 sampling intervals. We compared different scheduling algorithms in terms of time efficiency and energy efficiency. For time efficiency, we recorded the latency of data collection at each sampling interval and calculated the average latency over the 20,000 sampling intervals simulated. For energy efficiency, we focused on the energy spent by sensor nodes in transmitting data, receiving data and idle listening. Following the power consumption of a Mica2 mote [10], we assumed that the energy costs for a sensor node to perform transmission and listen for transmission in a time slot are 1 and 0.75 units of energy respectively. We recorded the total amount of energy consumed by each sensor node over the 20,000 sampling intervals simulated. Then, we added up these amounts to obtain the total energy consumed in the whole network. Since the network lifetime is largely determined by the most energy-consuming node [8], we also found out the node that consumed



(a) A sample topology. (b) Average latency of data collection. (c) Energy consumed at the most consuming node. (d) Total energy consumed in the whole network.

Fig. 4. A sample network topology and performance results for different error bounds of approximate data collection.

the largest amount of energy and plotted this amount for performance comparison.

B. Performance Results

Figs. 4(b), (c) and (d) show the performance of different scheduling algorithms as a function of the error bound of approximate data collection. Fig. 4(b) shows that the TIGRA schedule produces the same latency of data collection for all error bounds. This is because TIGRA targets at the full traffic pattern. But in approximate data collection, the traffic pattern changes over time in an unpredictable manner. Since the base station does not know a priori which nodes would report data in a sampling interval, it has to always wait until the end of the TIGRA schedule to make sure that all data have been received. In contrast, TPO and SPARSE are able to take advantage of lighter traffic load at larger error bounds to reduce the latency of data collection. However, for the sample network topology shown in Fig. 4(a), the latency overhead of SPARSE for constructing a new schedule at each sampling interval is 136 time slots, which far outweighs the benefit of deploying the new schedule for data collection. Therefore, the latency of SPARSE is much higher than that of TIGRA, even when the error bound is large. Our proposed TPO schedule considerably reduces the latency of data collection compared to TIGRA and SPARSE over a wide range of error bounds.

Figs. 4(c) and (d) show that our proposed TPO schedule leads to significantly lower energy consumption than TIGRA and SPARSE. In the TIGRA schedule, irrespective of the traffic pattern, the number of time slots each node has to listen equals the number of its descendants in the routing tree. Therefore, the nodes close to the base station spend a lot of energy in idle listening when the traffic load is light. This explains why the energy consumption of the most consuming node is generally much higher in TIGRA than in the other schedules (see Fig. 4(c)). On the other hand, while the SPARSE schedule is able to avoid idle listening, the energy overhead for constructing a new schedule at each sampling interval is substantial. Thus, as shown in Fig. 4(d), SPARSE results in even higher total energy consumption in the whole network than TIGRA. In our proposed TPO schedule, a receiving node can safely stop listening to a sending node when it identifies the end of transmissions performed by the sending node, thereby conserving energy. Fig. 4(c) shows that in approximate data collection, the most consuming node in TPO spends much less energy compared to TIGRA and SPARSE. This implies our TPO schedule can substantially prolong network lifetime over the other two algorithms.

IV. Conclusions

We have presented a TDMA schedule that is well suited to continuous data collection with dynamic traffic patterns. Our proposed schedule is traffic pattern oblivious in that it achieves high energy efficiency and time efficiency of data collection irrespective of the traffic pattern. Following this schedule, the energy consumed by sensor nodes for any traffic pattern is very close to the minimum required by their workloads given in the traffic pattern. The schedule also allows the base station to conclude data collection as early as possible according to the traffic load. Experimental results using real-world data traces show that, compared with existing schedules, our proposed schedule considerably reduces the latency of data collection and achieves significant energy savings.

References

[1] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh, "Deploying a Wireless Sensor Network on an Active Volcano," *IEEE Internet Computing*, vol. 10, no. 2, Mar.-Apr. 2006.

[2] 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, Jun. 2004.

[3] S. Gandham, Y. Zhang, and Q. Huang, "Distributed Minimal Time Convergecast Scheduling in Wireless Sensor Networks," *Proc. IEEE ICDCS '06*, Jul. 2006.

[4] Y. Zhang, S. Gandham, and Q. Huang, "Distributed Minimal Time Convergecast Scheduling for Small or Sparse Data Sources," *Proc. IEEE RTSS '07*, pp. 301–310, Dec. 2007.

[5] L. Paradis and Q. Han, "TIGRA: Timely Sensor Data Collection Using Distributed Graph Coloring," *Proc. IEEE PerCom '08*, Mar. 2008.

[6] X. Tang and J. Xu, "Adaptive Data Collection Strategies for Lifetime-Constrained Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 6, pp. 721–734, Jun. 2008.

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

[8] C. Buragohain, D. Agrawal, and S. Suri, "Power Aware Routing for Sensor Databases," *Proc. IEEE INFOCOM '05*, Mar. 2005.

[9] Live From Earth And Mars (LEM) Project, http://www-k12.atmos.washington.edu/k12/, 2009.

[10] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," *Proc. ACM SenSys '04*, Nov. 2004.