Computers, Environment and Urban Systems 33 (2009) 79–89

Contents lists available at ScienceDirect

Computers, Environment and Urban Systems

journal homepage: www.elsevier.com/locate/compenvurbsys

Optimizing query processing using selectivity-awareness in Wireless Sensor Networks

Muhammad Umer *, Lars Kulik, Egemen Tanin

National ICT Australia, Department of Computer Science and Software Engineering, University of Melbourne, Victoria 3010, Australia

ARTICLE INFO

Keywords:
Wireless Sensor Networks
Monitoring
Query optimization
Selectivity
Localized Steiner tree approximation

ABSTRACT

Monitoring queries are fundamental for Wireless Sensor Networks (WSNs) that collect data for physical phenomena. In this work we address three key characteristics of monitoring queries. First, a monitoring query can be selective, i.e., it requests readings only from parts of a WSN. Second, a monitoring query can be continuous, i.e., it draws sensor readings for long periods of time. Finally, since physical phenomena are spatially correlated, a monitoring query selects spatially co-located nodes. In our earlier work, we proposed the Pocket Driven Trajectories (PDT) algorithm; a selectivity-aware data collection technique that tailors data collection paths for a monitoring query based on the spatial layout of selected nodes. In this work, we extend the basic PDT algorithm with an adaptive behavior. We show that the enhanced PDT algorithm is ideal for real world WSNs due to its two major strengths; the PDT algorithm is local, i.e., it does not require any global information about node locations or network connectivity. Furthermore, the PDT algorithm efficiently adapts its data collection paths over the lifetime of a query as changes in the spatial layout of selected nodes occur. Using extensive simulations, we show that in terms of energy efficiency the PDT algorithm clearly outperforms well-known WSN data collection algorithms.

1. Introduction

Recent advances in hardware miniaturization and wireless communications have given rise to a new class of networked systems known as Wireless Sensor Networks (WSNs). These systems are empowered by a large number of self-contained sensor nodes, each smaller than a human palm, providing an unprecedented capability to sense and monitor the physical world. The low cost of the sensor nodes establishes an economy of scale that makes it possible to deploy these nodes in large numbers over vast areas. WSNs are finding applications in diverse fields such as volcanic activity monitoring, tsunami detection, and ecological monitoring (NASA Volcano Sensorweb, 2007; Secure CITI, 2006; Tolle et al., 2005). As in other large computing environments, the success of a WSN is tied to its sustainability, i.e., the ability of the system to maintain itself over time. Typically, the main constraint for a sensor node is its power. In a large WSN deployment, it is often not viable to change the batteries of the nodes. Therefore, minimizing power consumption is a prime objective common to all WSN operations, including sensing, data processing, and communication.

Monitoring of physical phenomena is one of the main areas of application for WSNs. In such applications, a database-oriented view of WSNs has proven to be useful. According to this view, a WSN is considered as a distributed data source where the sensed values generated by a set of WSN nodes form the rows of a relation split across all nodes in the set (Madden, Franklin, Hellerstein, & Hong, 2003; Yao and Gehrke, 2003). The database-oriented view motivates the design of WSN data acquisition regimes targeted at two fundamental aims. First, similar to classical database systems, a WSN database should provide SQL-like abstractions so that nodes can be easily programmed for data sensing and collection. Second, the data collection process should minimize the overall energy expenditure. Research into current sensor hardware has shown that the energy consumption of a sensor node is a function of its communication workload (Pottie & Kaiser, 2000). For a data acquisition system, this insight motivates the optimization of the communication process either by involving a smaller number of nodes, or by establishing efficient communication paths between the sensor nodes and the base station. In this paper, we consider these two techniques in tandem, and argue that certain distinct features of monitoring queries can be exploited to achieve higher levels of energy efficiency.

1.1. Characteristics of monitoring queries

We define a monitoring query as a continuous data collection task that requests sensed values from nodes fulfilling selection criteria based on certain physical conditions. For instance, queries

* Corresponding author.
E-mail addresses: mumer@csse.unimelb.edu.au (M. Umer), lars@csse.unimelb.edu.au (L. Kulik), egemen@csse.unimelb.edu.au (E. Tanin).

0198-9715/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compenvurbsys.2009.01.006

that monitor herd movements in a farm would report the sensed values only from the nodes that have recently sensed animal movements. The following are key aspects of monitoring queries:

- *Monitoring queries are selective*: typically, a WSN can cover an area much larger than the area of interest at any given point in time. For instance, in the herd monitoring example presented above, although nodes are present all over the farm, the herd may only be found within a limited area. We argue that for energy efficient optimization of such queries, the data collection task should be *selectivity-aware*.
- *Monitoring queries are continuous*: a monitoring task, by design, is expected to query readings from sensor nodes over an extended period of time. The mainstream WSN database systems have realized the need for continuous queries and provide SQL clauses to define such queries. For instance, the Cougar database system provides DURATION and EVERY clauses (Yao & Gehrke, 2003), which specify the lifetime of a query and the rate of the query answers, respectively.
- *Monitoring queries select spatially correlated nodes*: physical phenomena are characterized by their spatial correlation; hence, when monitoring a physical phenomenon, sensor nodes at proximal locations tend to have similar values. Therefore, this spatial correlation, coupled with the notion of selectivity, results in clustered or pocketed node participation. For instance, if a node is selected by a query based on its sensed temperature value, there is a high probability that neighboring nodes will also be selected by the same query.

1.2. Our contributions

In this work, we present an adaptive location-based data collection algorithm, Pocket Driven Trajectories (PDT), which optimizes the monitoring queries by exploiting their key characteristics. The PDT algorithm first discovers the set of pockets of selected nodes for a given query. It then aligns a data collection tree to the spatially optimal path connecting these pockets. This path minimizes the use of non-selected nodes in the data collection tree and consequently optimizes the overall energy efficiency of the data collection process. The PDT algorithm is *localized*; i.e., no computation in the algorithm requires any global information such as node connectivity or locations. Moreover, the PDT algorithm is *adaptive*; i.e., it continuously adapts the data collection tree to changing node participation. The PDT algorithm does incur set-up and tree adjustment costs; however, due to the continuous nature of the monitoring queries, these costs can be amortized over the lifetime of the query.

A basic version of the PDT algorithm was presented in one of our earlier works (Umer, Kulik, & Tanin, 2006). The basic version establishes the strength of selectivity-awareness in the optimization of monitoring queries. However, it does not adapt to changing node participation and environmental conditions. The basic version is thus limited in its potential applications. In this paper, we extend and improve our previous work by introducing adaptability into the algorithm and analyze the performance for various dynamic situations.

1.3. Organization

The remainder of this paper is organized as follows. In Section 2 we review related research and position our work in the context of the WSN literature. Section 3 presents the formal underpinnings of our algorithm, including a discussion of its adaptability and costs. Section 4 details the experiments and discusses the results of the simulation-based analysis of our algorithm. Finally, Section 5 presents the conclusions and discusses possible future directions.

2. Related work

Data collection has been an active research topic in the WSN community. As shown in Fig. 1, we identify three major trends in WSN data collection research: *Systems*, *Testbeds*, and *Algorithmic*. Systems research has been mainly led by data collection systems such as Directed diffusion (Intanagonwiwat, Govindan, Estrin, Heidemann, & Silva, 2003), TinyDB (Madden et al., 2003), and Cougar (Yao & Gehrke, 2002). The main motivation behind these systems is to simplify the users' access to a WSN by providing abstractions at the communication, routing, and node programming levels. Although these systems are based on a number of data collection algorithms, new applications and evolving requirements have generated a constant need for better algorithms. This need has thus motivated the algorithmic research in the WSN domain. Since our work also falls under the category of algorithmic research, we discuss this category in detail below.

2.1. In-network data aggregation

In-network data aggregation was one of the first data collection optimization approaches proposed for WSNs (Bonfils & Bonnet, 2003; Considine, Li, Kollios, & Byers, 2004; Madden, Franklin, Hellerstein, & Hong, 2002; Nath, Gibbons, Seshan, Anderson, 2004). In-network aggregation algorithms exploit the fact that a sensor node consumes less energy for information processing than for information communication. These techniques establish multi-hop data collection paths in a network where a node first aggregates the incoming packets of the nodes in communication range and communicates only the aggregated information to the next node. Some in-network aggregation techniques, such as TAG (Tiny AGgregation) (Madden et al., 2002, 2005), randomly create the data collection path and use it to compute all aggregate queries. In contrast, several recent approaches propose to tailor the data collection paths specifically to a group of similar queries (Bonfils and Bonnet, 2003; Pattem et al., 2004; Sharaf, Beaver, Labrinidis, & Chrysanthis, 2004). A data collection path can be tree-based or multi-path (graph) based (Bawa, Gionis, Garcia-Molina, Motwani, 2004; Nath et al., 2004). Interested readers are referred to (Umer et al., 2006) for a detailed review of major in-network data aggregation schemes.

2.2. Suppression-based data collection

Similar to data aggregation, suppression-based methods also attempt to reduce the amount of data required to be reported to the base station. As a spatial suppression method, clustered in-network aggregation exploits the spatial correlation of sensor readings to preserve energy (Pattem et al., 2004; Xu, Heidemann, & Estrin, 2001; Yoon & Shahabi, 2005). Data suppression can also be achieved by approximating physical phenomena using statistical models (Chu, Deshpande, Hellerstein, & Hong, 2006; Deshpande, Guestrin, Madden, Hellerstein, & Homg, 2004). For instance, the BBQ framework uses a statistical model along with live data acquisition so that a large number of queries can be answered locally by the base station (Deshpande, Guestrin, Madden, Hellerstein, & Homg, 2004). Similarly, *Ken* (Chu, Deshpande, Hellerstein, & Hong, 2006) proposed a dynamic probabilistic model for WSN data.

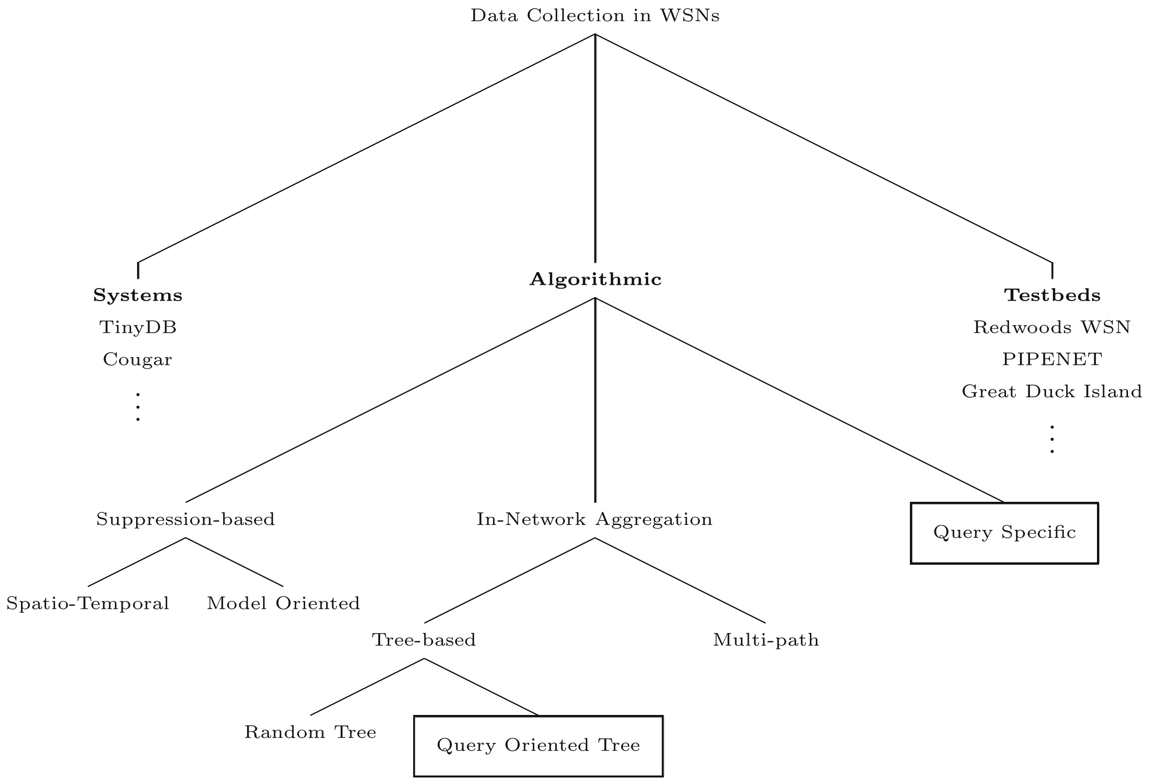


Fig. 1. A classification of recent works in the WSN data collection domain (see Section 2 for detailed references). Boxed items represent the classes under which our work falls.

2.3. Data collection algorithms for selective queries

Our selectivity-aware query optimization scheme minimizes the use of non-selected parts of a WSN to achieve overall energy efficiency. In general, this problem is an instance of the well known minimum Steiner tree problem (MSTP) of graph theory (Robins & Zelikovsky, 2000). In a graph, the MSTP seeks a minimum path tree spanning only a subset of nodes (terminal nodes) in the graph while minimizing the inclusion of non-terminal nodes. The MSTP is known to be NP-hard and has been widely discussed in the literature (Oliveira & Pardalos, 2005).

In the WSN domain, Krishnamachari, Estrin, and Wicker (2002) reported the use of an MSTP approximation algorithm that was proposed earlier by Takahashi and Matsuyama (1980). The main disadvantage of their work, and of most MSTP heuristics in general, is the global and static nature of heuristics, in that they assume the availability of global network knowledge and its stability. In a WSN, these assumptions are not practical since nodes have limited capability and WSNs are prone to node failures. In this work, we propose an MSTP heuristic that is both localized and adaptive in nature, and hence is better suited to large scale WSNs.

3. An adaptive selectivity-aware data collection algorithm

One could argue that a simple yet optimal data collection strategy could be to use *only* the selected nodes while routing the query results back to the base station. Such a path minimizes the unnecessary forwarding of results by non-selected nodes en-route from a selected node to the base station. However, using *only* the selected nodes in the data collection path is not possible due to the communication limitations of the individual nodes. Any attempt at a simplistic solution like this would quickly result in a disconnected network. Hence, the need for a certain number of non-selected nodes in the data collection path cannot be eliminated. As noted in Section 2, this problem can be directly mapped to the minimum Steiner tree problem (MSTP). Considering the limitations of the known MSTP heuristics, we propose the Pocket Driven Trajectories (PDT) algorithm, which is a localized and adaptive MSTP heuristic for establishing data collection paths in response to monitoring queries.

3.1. Algorithm overview

An overview of the main steps of the PDT algorithm is given below:

Establishing a random query tree: the base station initiates a query broadcast followed by each node, resulting in the formation of a random query tree.
Pocket discovery by locationaggregation: during the first sampling period each selected node piggybacks its location information with the required data. Each internal node performs spatial aggregation of the location information it receives from its child nodes and forwards the aggregated data to its parent. By the end of the first sampling period, the base station receives a set of spatial aggregates, each representing a pocket of selected nodes.
Establishing the initial data collectionpaths: the base station computes the shortest path, referred to here as the PDT, that connects all the pockets with itself. The base station broadcasts the PDT information.
Adapting the PDT: during each successive sampling period, location aggregation is performed alongside data collection using the PDT so that the changes to the original pockets can be monitored. The base station computes and broadcasts a new PDT as soon as the old pockets shift or vanish and new pockets emerge.

In the following subsections, we briefly discuss each of the above steps and analyze the cost and benefits of our data collection scheme.

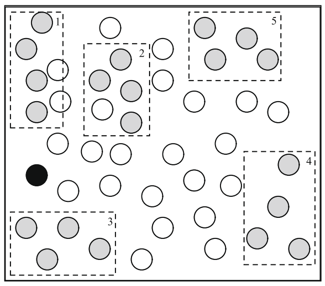
3.1.1. Establishing the initial PDT

3.1.1.1. System model. We assume that a WSN with n nodes is represented by a connected unit disk graph $G = (V, E)$, where V is the set of sensor nodes, and E is the set of direct communication links between node pairs. Each query Q issued by a base station S selects a subset T of V. Furthermore, a pocket is a cluster of proximal nodes selected by Q.

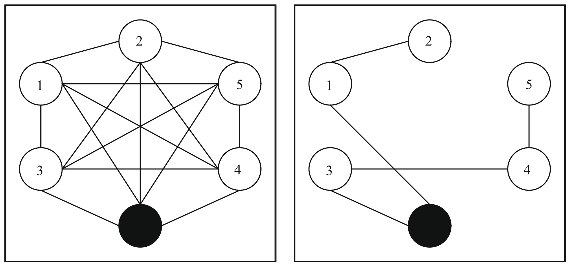
Step 1. *Establishing a random query tree: the* data collection process is started by the base station, which broadcasts the query Q to all nodes in its communication range. Each of the receiving nodes chooses the base station as its parent node and rebroadcasts Q, reaching the nodes located further away. Each node that receives Q sets the sender node as its parent and rebroadcasts Q. It also discards any further transmissions of Q that it may receive from other nodes in its communication range. In a connected WSN, this process is guaranteed to reach every node in the network and is self-terminating, i.e., it automatically stops as soon as every node has received and rebroadcasted the query. It is possible to approximate the amount of time required by the tree creation process using application-specific knowledge about the network size, deployment area, and data rate. This approximation allows a user to specify the sampling period of a query such that none of the nodes start sending its data until the entire network has received the query.

Step 2. *Pocket discovery by location aggregation: in* the first sampling period of the query Q, the leaf nodes selected by Q start the data collection phase by sensing and reporting the requested sensor readings to their parent nodes. Along with the sensed readings, the leaf nodes also piggyback their location information in the same data packet. Each node v receives this data from one or more of its child nodes and combines the location information in a set, λ_v. It then performs *location aggregation* as follows. First, it identifies each item of the location information set as atomic information (belonging to a single node) or aggregated information (belonging to a set of proximal nodes). Second, it aggregates the atomic locations by computing an axis-aligned minimum bounding rectangle (MBR) for all atomic locations that are within a certain threshold distance, ε (which we set equivalent to the communication radius of the nodes for simplicity). Finally, it attempts to further aggregate the location information set by merging the newly aggregated location information with the aggregates reported by its child nodes.After applying the location aggregation, each internal node forwards the aggregated information to its parent node, where the process is repeated. Eventually, the base station receives and aggregates the location information. We refer to the final set of MBRs in the aggregated location information of the base station as *pockets*.

Step 3. *PDT Computation: at* the end of the first sampling period, the base station uses the discovered set of pockets to compute the PDT. Representing itself and each pocket as a node, the base station creates an MBR-based representation of node participation as a complete graph G'. Each edge of G' carries a weight equivalent to the geographic distance between the corresponding nodes. The base station then creates a minimum spanning tree MST' for G'. We refer to MST' as the PDT for data collection from the pockets discovered so far. For the MBR-based representation of node participation, MST' minimizes the length of the data collection tree that connects all pockets with the base station.Using MST', the PDT information, $\mathscr{P}$, can be encoded as follows: $\mathscr{P}$ comprises of a series of location tuples, where (i) each location in a tuple corresponds to either the base station location or the center point of one of the pockets, and (ii) the location tuple corresponds to an edge in MST'.



(a) A selective participation scenario with five pockets.



(b) A complete graph as an abstract representation of the scenario (c) MST'

Fig. 2. The computation of the PDT for a selective participation scenario.

Step 4. *PDT Broadcast and establishment: after* successful computation of the PDT information, $\mathscr{P}$, the base station starts to establish it as the new data collection path. It broadcasts $\mathscr{P}$ to its direct children. A node that receives the PDT information decides to join the PDT based on its query selection status and the distance from the PDT. All nodes that decide to join the trajectory switch their current parent to the sender of the PDT information and rebroadcast the PDT information. The process stops automatically after each joining node switches its parent and broadcasts the PDT information. The successive forwarding and parent switches lead to a new data collection tree that is aligned with the PDT.Fig. 2 illustrates the above steps of the PDT-based data collection scheme. In Fig. 2a, the black circle represents the base station while the shaded circles represent the nodes currently selected by a query. An MBR-based approximation of a pocket may include some non-selected nodes, as shown by the unshaded circles in pockets 1 and 2 in the example in Fig. 2a. Algorithm 1 outlines the PDT setup method for an arbitrary node v. A detailed formal construction of these steps can be found in our earlier work (Umer et al., 2006).

Algorithm 1: PDT Setup for node v

```
// $L_p$: list of PDT data packets where each PDT packet $P$ is composed
   of:
// $s_p$: node id of message sender
// $t_p$: packet type, as either Query, PDT path, or Data
// $Q_p$: aggregation query
// $\mathcal{P}_p$: PDT path information
// $D_p$: aggregated sensed data as requested by $Q_p$
// $\lambda_p$: location information of node $s_p$ computed using a location
   aggregation function, $\mathcal{L}$
// $\mathcal{J}(\mathcal{P}_p)$: a Boolean PDT join decision function: 1 if a node decides
   to join the PDT, 0 otherwise
Input: PDT packet $P$
Output: PDT packet $P$

1.1  while transmission interval is not expired do
1.2      Receive incoming packet $P$
1.3      if ($t_p$ = Query AND $p_r$ is null) OR ($t_p$ = PDT path and $\mathcal{J}(\mathcal{P}_p) = 1$) then
1.4          Set $s_p$ as parent node
1.5          Reset transmission interval as sampling period of $Q_p$
1.6          $s_p \leftarrow v$
1.7          Broadcast $P$
1.8      else if $t_p = Data$ then
1.9          Append $P$ to $L_p$
1.10
1.11 if $v$ is selected by $Q_p$ OR $L_p$ is not empty then
1.12     Create an empty PDT packet $P$
1.13     foreach packet $p$ in $L_p$ do
1.14         $\lambda_P \leftarrow \lambda_P \cup \lambda_p$
1.15         $D_P \leftarrow D_P \cup D_p$
1.16     if $v$ is a selected node then
1.17         Append $v$'s location and data to $\lambda_P$ and $D_P$
1.18     $\lambda_P \leftarrow \mathcal{L}(\lambda_P)$
1.19     Perform query specific aggregation on $D_P$
1.20     if $v$ is not the base station then
1.21         Send $P$ to the parent node
1.22     else
1.23         $\mathcal{P}_p \leftarrow MST(\text{Clique}(\lambda_v))$
1.24         Create an empty PDT packet $P'$
1.25         Add $\lambda_P$ to $P'$
1.26         Report $D_P$ to the user
1.27         Broadcast $P'$
```

3.1.2. Adapting the PDT

A continuous query can consist of a large number of sampling periods. Even if the change per period is small, the node selection can change significantly during the lifetime of a query. In the context of such queries, we categorize changes in sensor readings into two major types: changes in the values of *selected* attributes and changes in the values of *predicate* attributes. In order to adapt the PDT to changes during a continuous query, it is important to understand the respective impact of each of these change categories. Consider the monitoring queries given in Table 1. Query 1 represents a case where changes occur only in the values of selected attributes; Query 2 represents the case where changes can occur

Table 1
Example queries to demonstrate the impact of change in sensor readings.

Query 1	Query 2	Query 3

SELECT AVG (Humidity)
FROM Sensors
WHERE Location
BETWEEN ((0,0), (10,10))
SAMPLE PERIOD 60 min
FOR 48 h
	SELECT MAX (Altitude)
	FROM Sensors
	WHERE Temperature
	BETWEEN (31,40)
	SAMPLE PERIOD 60 min
	FOR 48 h
		SELECT AVG (Humidity)
		FROM Sensors
		WHERE Temperature
		BETWEEN (31,40)
		SAMPLE PERIOD 60 min
		FOR 48 h

due to variations in the predicate attributes' values; Query 3 represents a hybrid case where changes could arise from variations in either or both the selected and the predicate attributes.

Changes in the selected attributes do not have a spatial effect on the execution of a query. For instance, Query 1 always draws its sample from a specific set of sensors with no spatial change throughout the query's lifetime. Temporal suppression (Silberstein, Braynard, & Yang, 2006) and adjustments to the sampling period can be used to improve the query execution efficiency in this case. Such methods can be used in tandem with the PDT algorithm.

Adapting to changes in predicate attributes presents a major challenge for our approach. This type of change can have dramatic spatial effects on pocket layout during the lifetime of a query. For example, for Queries 2 and 3, the set of participating nodes can continuously change, resulting in new spatial configurations for the pockets. The goal of adaptivity in the PDT algorithm is to recompute and maintain energy efficient data collection paths in the face of changing spatial pocket layouts.

We propose the following mechanism to efficiently adapt the PDT algorithm to changes in the set of selected nodes. As soon as a new node finds itself to be fulfilling the query predicate, it sends its data from the random query tree established during the first sampling period to its parent. Each internal node maintains a separate count for its child nodes that have sent their data using the random query tree as well as a count for the ones that have used the PDT. These counts are propagated and incremented as the data move up towards the base station. Once the base station receives the data for the current sampling period, it determines the ratio of the number of nodes using the random tree to the number of nodes using the PDT. As soon as this ratio reaches a threshold value, the base station updates the PDT by using its current location information set λ_S and broadcasts this updated PDT information. This information is then forwarded by internal nodes in the network, as in the PDT setup phase.

3.2. Communication overheads

The extra message load generated during the PDT information broadcast phase can be significant, and it is a function of the number of nodes that decide to join the PDT. However, in return the use of the PDT reduces the number of non-selected intermediate nodes in the query tree. Therefore, the overall energy usage is still expected to be less, as the PDT setup (and refresh) costs can be amortized over a number of sampling periods. Assume that a random tree-based algorithm establishes a query tree that uses R nodes to collect the data from $R' \ll R$ selected nodes. For the same query, the PDT algorithm provides a $k\%$ improvement in node usage, and hence uses P nodes, where $P = R - R \times \frac{k}{100}$. Furthermore, assume that the node participation remains fixed during the next e sampling periods. Each node generates a single message per sampling period. Therefore, the total cost of the random tree method in terms of the number of messages generated will be $M_r = e \times R$, while the cost of the PDT will be $M_p = (e \times P) + P$. The extra P messages in the latter case are generated during the PDT setup phase. Setting $M_r = M_p$ and solving for e leads to $e = \frac{1-k}{k}$, i.e., the minimum number of sampling periods required by the PDT algorithm to amortize the extra setup cost and break even with the random tree method. We show in Section 4 that the typical reduction in node usage between the PDT algorithm and a random query tree-based algorithm for selective queries is 30%. Thus, in such cases, the PDT algorithm requires only 2.3 sampling periods to break even with its setup costs, and any samples drawn afterwards increase the benefit. However, if the reduction in node usage per sampling period is small or the number of samples between successive PDT refreshes is small, the setup and update costs may outweigh the benefits of using the PDT.

4. Experimental evaluation

In this section, we compare the performance of the PDT algorithm with that of other major data collection schemes. In addition to the PDT algorithm, we implement two well-known data collection algorithms: Tiny Aggregation (TAG) and Static Clustering (SC). In order to compare the data collection schemes with the optimal data collection tree, i.e., the minimum Steiner tree, we implement a global Steiner tree approximation algorithm that was proposed by Kou, Markowsky, and Berman (1981) (henceforth referred to as KMB). The KMB algorithm allows us to compute a Steiner tree that has been shown to achieve a mean efficiency that is worse by no more than 5% compared to the optimal Steiner tree (Oliveira & Pardalos, 2005; Doar & Leslie, 1993). The KMB algorithm is applied centrally, i.e., it is assumed that the base station has complete knowledge of the network graph. KMB forms a benchmark for our comparisons.

4.1. Evaluation parameters

We first lay out the evaluation parameters that we use to analyze the impact of the spatial and temporal characteristics of monitoring queries.

4.1.1. Query selectivity

The PDT algorithm is primarily designed for continuous selective queries. Therefore, the primary parameter that we consider for its evaluation is query selectivity. We define the selectivity of a query as the ratio of the number of selected nodes to the total number of nodes in a WSN.

4.1.2. Average change in node participation

We define the average change in a monitoring query as the average change in the node selection in successive sampling periods. Formally, if T_i is the set of selected nodes during sampling period i, then we compute the change in node participation between periods i and $i+1$ as

$$c_{i,i+1} = \frac{|T_i \setminus T_{i+1}| + |T_{i+1} \setminus T_i|}{|T_i \cup T_{i+1}|}.$$

Intuitively, $c_{i,i+1}$ is the ratio of the total number of nodes that change their selection status between sampling periods i and $i+1$, to the total number of selected nodes in both sampling periods. The average change in node selection for e sampling periods is then defined as

$$\hat{c} = \frac{\sum_{i=1}^{e-1} c_{i,i+1}}{e-1}.$$

4.1.3. Cost of data collection algorithms

We use the total data transmission (in MBs) as an indication of energy usage, and hence as the basic metric for the cost comparison of the aggregation algorithms. The level of data transmission can be related to the energy expenditure by a simple function, such as $\varepsilon = \sigma_s + \delta_s x$, where ε is the total amount of energy consumed by sending a message with x bytes of content, and σ_s and δ_s represent per-message and per-byte communication costs, respectively (Silberstein, Braynard, & Yang, 2006).

4.2. Results

In order to present a thorough comparative analysis, we design the following experiments. In Section 4.2.1, we analyze the impact of different levels of query selectivity on the PDT algorithm and compare it with other major data collection schemes. In Section 4.2.2, we simulate the case where the change in attribute values results in a continuously changing set of selected nodes while the query selectivity remains approximately the same. Finally, also in Section 4.2.2, we simulate queries where both the query selectivity and the set of selected nodes vary during each sampling period.

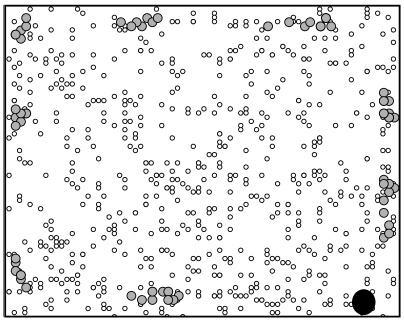
4.2.1. Impact of query selectivity

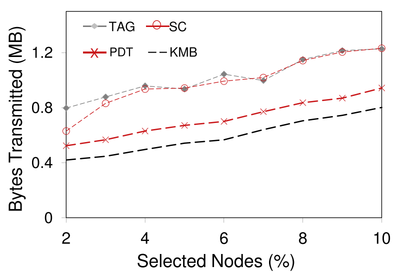
In this set of experiments, we investigate the impact of query selectivity on four different aggregation schemes: PDT, SC, TAG, and KMB. In the first experiment, presented in Fig. 3, we decrease the selectivity of an aggregate query and hence increase the number of selected nodes in 1% increments from 2% to 10% of the total nodes in the network. As shown in the deployment snapshot in Fig. 3a, the selected nodes are spatially clustered. In each instance in this experiment, we compute the average of a sensed attribute while running the query for 100 sampling durations. The network consists of 750 nodes deployed randomly in a 75 m^2 region, while the communication radius of all nodes is set at 5 m. To capture the impact of selectivity independent of other attributes, we assume that there is no change in the node participation over the query lifetime. Fig. 3b shows the mean value of the number of bytes transmitted by each algorithm at each selectivity level (the average of five runs is used to find the mean value). Fig. 3c shows the results of a similar experiment where we increase the number of selected nodes from 10% to 60% in discrete increments of 10% per step.

Fig. 3b shows that the PDT algorithm clearly outperforms the TAG and SC based data collection schemes, and is closest to the lower bound established by KMB. In cases where 2% to 10% of the nodes are selected by a query, the PDT algorithm is, on average, 41% more energy efficient than TAG and 37% more energy efficient than SC. In addition, the PDT algorithm is only 21% less efficient than the approximated lower bound established by the KMB algorithm, whereas TAG and SC are 72% and 67% less efficient, respectively. Similarly, the trend in Fig. 3c shows that the PDT algorithm remains energy efficient even in queries with low selectivity, but its advantage decreases as the selectivity decreases. When 10% of the nodes are selected, the PDT algorithm requires 30% fewer data transmissions than TAG and 31% fewer than SC; however at the selectivity level of 60% this advantage reduces to 4% and 5% for TAG and SC, respectively. The decrease in efficiency results from the fact that with the increase in the number of nodes selected by a query, the benefit of spatial correlations diminishes, and hence all schemes tend to use nearly the same number of nodes. This effect can also be observed from the fact that at the 60% selectivity level, the PDT algorithm is just 3% less efficient than the KMB lower bound.

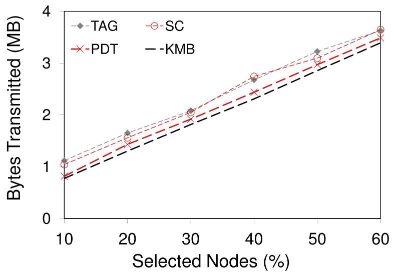
4.2.2. Impact of change in node participation

4.2.2.1. Moving pockets with fixed overall selectivity. The experiments in Section 4.2.1 show the performance gain of the PDT algorithm over other data collection techniques in selective queries.


(a) A spatial configuration for selective node participation. Larger circles represent nodes selected by the query. The black circle represents the base station.



(b) Detailed comparison of data transmissions for each algorithm in low participation scenarios



(c) The overall trend for each algorithm in terms of data transmissions

Fig. 3. The performance of aggregation techniques for varying levels of partial node participation.

However, in these experiments, the set of selected nodes remained fixed over the query lifetime. In this section, we relax this assumption and analyze the cost and adaptability of the PDT algorithm in selective queries where the query selectivity remains fixed but the set of selected nodes changes. This change can be considered as a periodic shift in the node participation correlated with a changing physical phenomenon. Queries similar to Queries 2 and 3 in Table 1 can lead to such changes in the set of selected nodes.

In this experiment, we simulate a herd-monitoring query using real world animal tracking data from the Starkey Project database (Kie et al., 2005; Starkey Project, 2006). In order to use the recorded Starkey data, we assume the presence of a randomly deployed, connected WSN in the Starkey forest area ($8000 \times 14,000\ \mathrm{m}^2$). For this experiment, we configure an 850 node network where the communication and sensing radius of each node is set at 600 m. In this experiment, the base station issues an aggregate query to monitor the movement of cattle on a continuous basis. Fig. 4 shows the network deployment and participation snapshots during different sampling periods of the above query.

The charts in Fig. 5 show the node usage by the PDT, TAG, and KMB algorithms with the value of D set to four days and the value of d set to 60 min. On average, the PDT algorithm generates 9.25%

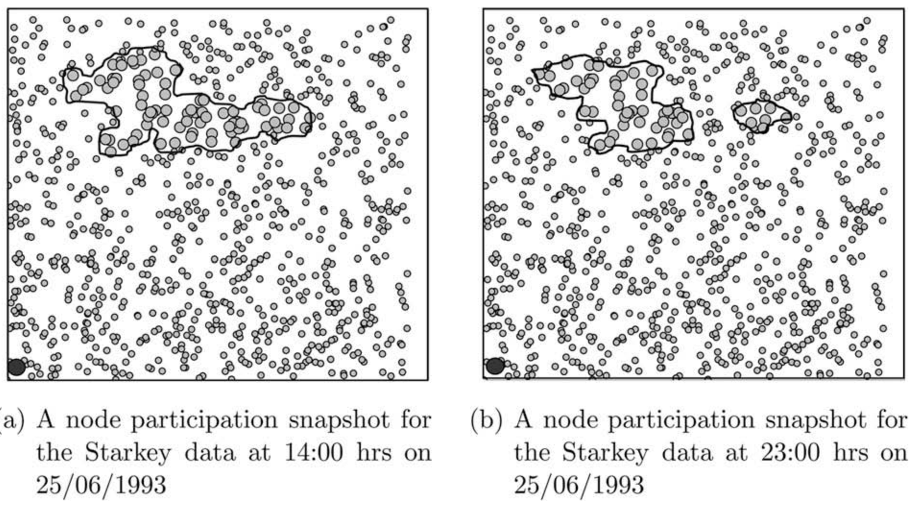
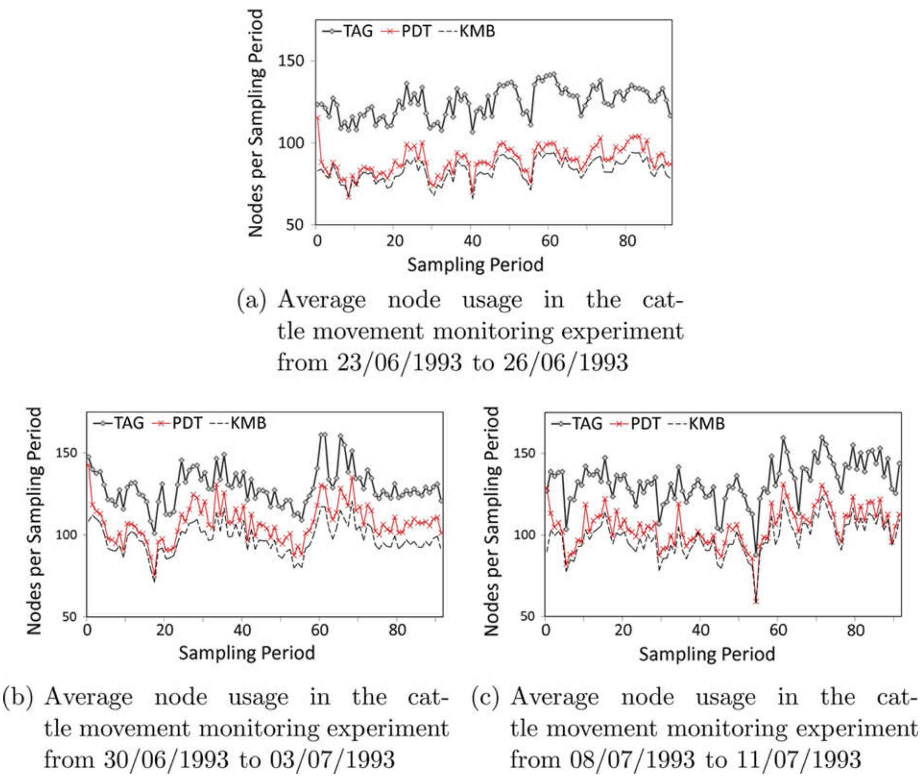

(a) A node participation snapshot for the Starkey data at 14:00 hrs on 25/06/1993 (b) A node participation snapshot for the Starkey data at 23:00 hrs on 25/06/1993

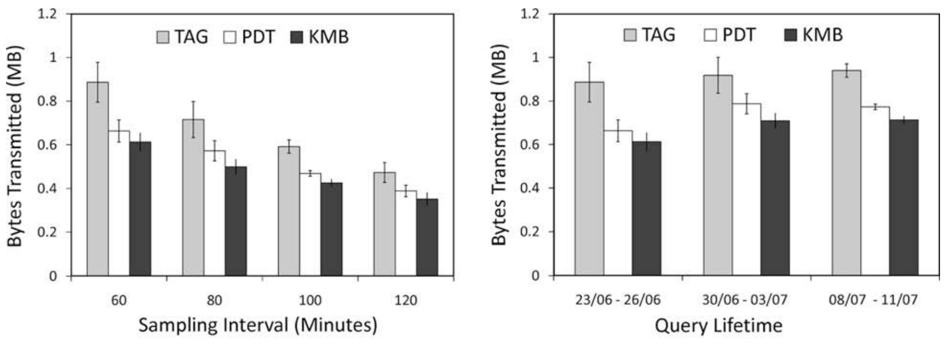
Fig. 4. Intermediate spatial layout of the node participation in the cattle movement monitoring experiment. Large circles represents the sensors participating in the cattle movement monitoring query. The base station is represented by the black circle.



(a) Average node usage in the cattle movement monitoring experiment from 23/06/1993 to 26/06/1993

(b) Average node usage in the cattle movement monitoring experiment from 30/06/1993 to 03/07/1993 (c) Average node usage in the cattle movement monitoring experiment from 08/07/1993 to 11/07/1993

Fig. 5. Average node participation on different days in the cattle movement monitoring experiment with sampling period d = 60 min.



(a) Algorithm performance with respect to increasing sampling period d. Query lifetime D = 4 Days from 23/06/1993 to 26/06/1993

(b) Algorithm performance for various day ranges. Sampling period d = 60 minutes

Fig. 6. Analyzing the performance of algorithms during different days and with respect to decreases in sampling frequency.

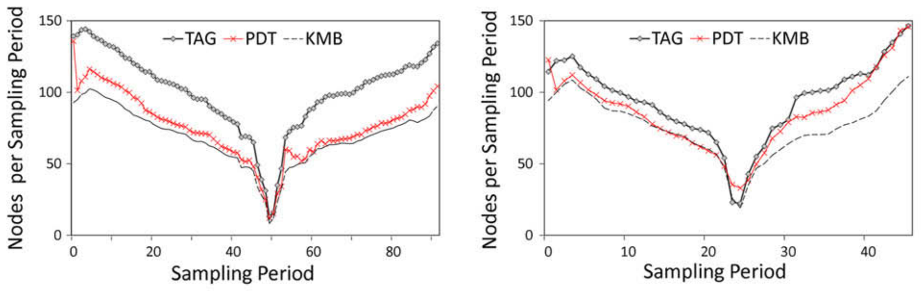
more data than the minimum bound of KMB, while TAG is clearly much more expensive, as it generates 35.3% more data on average.

Fig. 6 shows that the performance of the PDT algorithm, despite its localized nature, is similar to that of the globally approximated KMB. In the query with a sampling period of 60 min (Fig. 6a), the PDT algorithm is just 8.2% more expensive than KMB, while for the same experiment TAG is 44.8% more expensive. For a 60 min sampling period, the PDT algorithm generates 25.2% less data than TAG. The average change per sampling period in this experiment is 12.6% while the average overall node participation is 8%, which varies within a range of $\pm 1\%$ of the total nodes. For the query for the same duration but with a sampling period of 120 min, the PDT algorithm becomes 10.7% more expensive than KMB, while the query lifetime and node participation stay the same. The average change per sampling period, however, increases slightly as a result of a longer sampling period. Fig. 6b summarizes the performance of each algorithm during three separate day ranges. On average, PDT generates 9.25% more data than the minimum bound of KMB, while TAG is clearly much more expensive, as it generates 35.3% more data on average.

The decrease in the performance of the PDT algorithm between experiments with short and long sampling periods can be explained by the fact that in a long running query, the cost of establishing and refreshing the optimized data collection path through PDT is amortized over the number of sampling periods that pass between successive PDT establishment phases. With shorter sampling periods, queries poll the WSN more frequently and hence become ideal candidates for savings. On the contrary, a scheme that uses extra nodes during each sampling period, such as TAG, becomes more and more expensive. However, the performance gain drops with the decrease in sampling frequency as the setup and maintenance cost cannot be amortized as effectively.

4.2.2.2. Moving pockets with variable selectivity. The Starkey data set provides good insight into the performance of the PDT algorithm in a realistic setting by relaxing the assumption of a static node participation set. However, in the simulated phenomenon in the experiments above, the query selectivity, i.e., the number of selected nodes, is assumed to be fixed. In the next experiment, we relax this assumption. We model this experiment on the naturally occurring phenomenon of change in shadow lengths during the day. The predicate in the monitoring query is formulated such that the nodes falling inside a shaded area qualify to participate in the query. In order to keep the results comparable, we retain the network dimensions, number of nodes, communication range, sampling period and query lifetime from the previous experiment. We assume the presence of several ground objects and simulate the shadows by the relative position of objects with respect to the sun during different times of a day.

In this simulation, the number of selected nodes decreases with the decrease in shadow lengths and reaches a minimum at around midday, after which the shadow lengths start to increase again, thus increasing the number of selected nodes. Figs. 7 and 8 show the average node usage and overall data transmission by TAG, the PDT algorithm, and KMB during queries with 60 and 120 min sampling periods. In the first experiment (Fig. 7a), the ratio of selected nodes to the total nodes in the network falls from 7% to 1% during the first half of the query lifetime, and rises again to 7% during the second half. The average change in selected nodes per sampling period is 6.5%. In this experiment, TAG is 43.8% more



(a) Average node participation in the shadow experiment. d = 60 minutes (b) Average node participation in the shadow experiment. d = 120 minutes

Fig. 7. The performance of aggregation techniques in the shadow experiment.

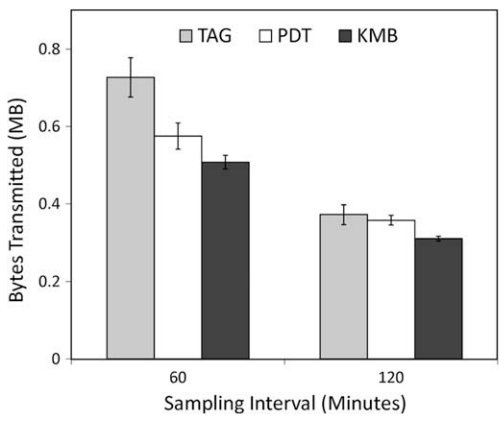


Fig. 8. Performance of algorithms with respect to increasing sampling periods in the shadow experiment.

expensive than the lower bound established by the KMB algorithm, while the PDT algorithm is 13.2% more expensive. In this experiment, the PDT algorithm is 20.8% cheaper than TAG. In the second experiment (Fig. 7b), we increase the sampling period from 60 to 120 min, while retaining other parameters. As a result, the change per sampling period increases to 13.4%. We see a deterioration in the performance of the PDT algorithm. In this experiment, the PDT algorithm is just 3.7% cheaper than TAG. This result is expected, as a decrease in sampling frequency means that the cost of establishing the PDT algorithm cannot be amortized as effectively. Furthermore, a higher rate of change per sampling period results in frequent abandonment of the data collection paths established by the PDT algorithm.

4.3. Summary of results

Our experiments establish that the PDT algorithm can achieve performance that is closer to the centralized approximation of the minimal Steiner tree than other major data collection schemes. However, the cost of achieving this performance is an important factor that must be considered before prescribing the PDT algorithm fit for use for all queries. The experiments show that for low selectivity queries, the PDT algorithm yields significant performance gains if the query requires frequent sampling over a long lifetime. If the query does not involve a large number of samples, the cost of the PDT algorithm setup and refresh phases can become considerably higher compared to the accumulated savings per sampling period.

Changes in node participation over the lifetime of a monitoring query leads to pocket movement with either fixed or variable selectivity. Our experiments show that the PDT algorithm adapts to the changing node participation in both cases. We observed that in some natural phenomena, even though the node participation changes over time, the change does not lead to a significant difference in the layout of the data collection paths. This helps in delaying the PDT algorithm refresh phase, as the initial data collection paths remain useful for several sampling periods. However, the PDT algorithm struggles to adapt to a rapid increase in node participation, as there are not enough sampling periods between successive refresh phases.

5. Conclusions and future work

Selective queries are required for effective monitoring of physical phenomena using WSNs. In this paper, we presented the PDT algorithm, a data collection method for long running selective queries. Using extensive simulations, we showed that the PDT algorithm is not only more energy efficient than other major data collection schemes, but, more importantly it is also similar to a well-known approximation of the global optimum, i.e., the minimum Steiner tree. The PDT algorithm discovers pockets of selected nodes using purely local information and approximates a minimum Steiner tree for data collection from these pockets. We showed that this can lead to significant energy savings in various kinds of selective queries. The PDT algorithm can also adapt to changing environmental conditions in an efficient manner.

There are several research directions that we plan to follow in our future studies. First, it is important to study the latency incurred by the PDT algorithm by increasing the data collection path while decreasing the energy consumption. Second, we aim to improve our PDT algorithm by a guided query multicast phase as opposed to the query broadcast reported in this paper. Our work can be viewed as a method to discover restricted regions inside which a query tree maximizes the use of selected nodes. Once such pockets are discovered, many data collection schemes can then be used inside them. This paper reports only the application of a random tree-based data collection inside a pocket. In the future, we plan to investigate the efficiency of other methods for data collection inside a pocket, including spatio-temporal and model-based data suppression strategies.

References

Bawa, M., Gionis, A., Garcia-Molina, H., Motwani, R. (2004). The price of validity in dynamic networks. In *Proceedings of SIGMOD* (*ACM SIGMOD international conference on management of data*) (pp. 515–526).
Bonfils, B., & Bonnet, P. (2003). Adaptive and decentralized operator placement for in-network query processing. In *Proceedings of IPSN* (*International conference on information processing in sensor networks*) (pp. 47–62).
Chu, D., Deshpande, A., Hellerstein, J., & Hong, W. (2006). Approximate data collection in sensor networks using probabilistic models. In *Proceedings of ICDE* (*International conference on data engineering*) (p. 48).
Considine, J., Li, F., Kollios, G., & Byers, J. (2004). Approximate aggregation techniques for sensor databases. In *Proceedings of ICDE* (*International conference on data engineering*) (pp. 449–460).
Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J., & Homg, W. (2004). Model-driven data acquisition in sensor networks. In *Proceedings of VLDB* (*International conference on very large data bases*) (pp. 588–599).
Doar, M., & Leslie, I. M. (1993). How bad is naive multicast routing? In *Proceedings of INFOCOM* (*Annual joint conference of the IEEE computer and communications societies*) (pp. 82–89).
Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., & Silva, F. (2003). Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking, 11*(1), 2–16.
Kie, J. G., Ager, A. A., Cimon, N. J., Wisdom, M. J., Rowland, M. M., Coe, P. K., et al. (2005). The Starkey databases: Spatial-environmental relations of North American elk, mule-deer, and cattle at the Starkey experimental forest and range in Northeastern Oregon. In M. J. Wisdom (Ed.), *The Starkey project: A synthesis of long-term studies of elk and mule-deer* (pp. 29–41). Lawrence, Kansas, USA: Alliance Communications Group.
Kou, L., Markowsky, G., & Berman, L. (1981). A fast algorithm for Steiner trees. *Acta Informatica, 15*, 141–145.
Krishnamachari, B., Estrin, D., & Wicker, S. B. (2002). The impact of data aggregation in wireless sensor networks. In *Proceedings of ICDCSW* (*International conference on distributed computing systems*) (pp. 575–578).
Madden, S., Franklin, M. J., Hellerstein, J. M., & Hong, W. (2002). A Tiny AGgregation service for ad-hoc sensor networks. *SIGOPS Operating Systems Review, 36*(SI), 131–146.
Madden, S., Franklin, M. J., Hellerstein, J. M., & Hong, W. (2003). An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems, 30*(1), 122–173.
NASA Volcano Sensorweb. (2007). <http://sensorwebs.jpl.nasa.gov/>.
Nath, S., Gibbons, P. B., Seshan, S. Anderson, Z. R. (2004). Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of SenSys* (*International conference on embedded networked sensor systems*) (pp. 250–262).
Oliveira, C. A. S., & Pardalos, P. M. (2005). A survey of combinatorial optimization problems in multicast routing. *Computers and Operations Research, 32*(8), 1953–1981.
Pattem, S., Krishnamachari, B., & Govindan, R. (2004). The impact of spatial correlation on routing with compression in wireless sensor networks. In *Proceedings of IPSN* (*International conference on information processing in sensor networks*) (pp. 28–35).

Pottie, G. J., & Kaiser, W. J. (2000). Wireless integrated network sensors. *Communications of the ACM, 43*(5), 51–58.

Robins, G., & Zelikovsky, A. (2000). Improved Steiner tree approximation in graphs. In *Proceedings of SODA* (*Annual ACM-SIAM symposium on discrete algorithms*) (pp. 770–779).

Secure CITI – Exploratory Research on Sensor-Based Infrastructure for Early Tsunami Detection. (2006). <http://www.cs.pitt.edu/s-citi/tsunami/>.

Sharaf, M. A., Beaver, J., Labrinidis, A., & Chrysanthis, P. K. (2004). Balancing energy efficiency and quality of aggregate data in sensor networks. *The VLDB Journal, 13*(4), 384–403.

Silberstein, A., Braynard, R., & Yang, J. (2006). Constraint chaining: On energy-efficient continuous monitoring in sensor networks. In *Proceedings of SIGMOD* (*ACM SIGMOD international conference on management of data*) (pp. 157–168).

Starkey Project. (2006). <http://www.fs.fed.us/pnw/starkey/index.shtml/>.

Takahashi, H., & Matsuyama, A. (1980). An approximate solution for the Steiner problem in graphs. *Math Japonica, 24*, 573–577.

Tolle, G., Polastre, J., Szewczyk, R., Culler, D., Turner, N., Tu, K., et al. (2005). A macroscope in the redwoods. In *Proceedings of SenSys* (*International conference on embedded networked sensor systems*) (pp. 51–63).

Umer, M., Kulik, L., & Tanin, E. (2006). A location based aggregation algorithm for selective aggregate queries in sensor networks. In *Proceedings of GSN* (*International conference on geosensor networks*).

Xu, Y., Heidemann, J., & Estrin, D. (2001). Geography-informed energy conservation for ad hoc routing. In *Proceedings of MobiCom* (*ACM/IEEE international conference on mobile computing and networking*) (pp. 70–84).

Yao, Y., & Gehrke, J. (2003). Query processing for sensor networks. In *Proceedings of CIDR* (*Biennial conference on innovative data systems research*) (pp. 233–244).

Yao, Y., & Gehrke, J. (2002). The Cougar approach to in-network query processing in sensor networks. *ACM SIGMOD Record, 31*(2), 9–18.

Yoon, S., & Shahabi, C. (2005). Exploiting spatial correlation towards an energy efficient clustered aggregation technique (CAG). In *Proceedings of IEEE ICC* (*IEEE international conference on communications*) (pp. 82–98).