Energy-Efficient Data Management For Sensor Networks: A Work-In-Progress Report

Alan Demers*[‡], Johannes Gehrke*, Rajmohan Rajaraman[†], Niki Trigoni*, and Yong Yao*
*Department of Computer Science, Cornell University, Ithaca, NY 14853
[†]College of Computer and Information Science, Northeastern University, Boston MA 02115
[‡]Corresponding author; Email: ademers@cs.cornell.edu; Phone: 607-255-9207; Fax: 607-255-4428

***Abstract*— We give a status update of the Cougar Project, in which we investigate a database approach to sensor networks: Clients "program" the sensors through *queries* in a high-level *declarative* language (such as a variant of SQL). In this paper, we overview our activities on energy-efficient data dissemination and query processing. Due to space constraints, we cannot present a full menu of results; instead, we decided to only whet the reader's appetite with some interesting problems in energy-efficient routing and in-network aggregation and some thoughts on how to approach them.**

I. Introduction

A powerful paradigm in sensor network design that has emerged recently: Give users a declarative query interface to the sensor data, thereby abstracting away the physical properties of the network when tasking the sensors. Such a *sensor database system* sends event data from source nodes to selected storage nodes called *view nodes* where the data is collected for further in-network processing. Many such sensor networks have strong constraints on their energy usage to maximize network lifetime. A significant amount of energy can be preserved by carefully determining (1) the data that should be stored in designated view nodes as well as (2) coordinating the data dissemination to these nodes.

In this paper we overview two ongoing research directions. Our first direction is *view selection*. In order to minimize the number of messages for a given query workload, we introduce a hybrid *pull-push* model, in which relevant data is collected at sensor nodes and *pushed* to view nodes, from where the data can be *pulled* when queries are issued. Our goal is to decide, given a query workload, what data we should store and where in the network this data should be stored in order to minimize the expected overall query cost given the query workload.

Our second direction is *wave scheduling*. We propose to *schedule* transmissions among nodes such that data flows quickly from event sources to storage nodes while avoiding collisions at the MAC layer. Since all nodes adhere to the schedule, most nodes can be turned off and only wake up during well-defined time intervals, resulting in significant energy savings. We show how routing protocols can be modified to interact symbiotically with the scheduling decisions, resulting in significant energy savings at the cost of higher latency.

In the remainder of the paper, we first introduce our model of a sensor network (Section II), we then overview ongoing work on view selection (Section III), and we then overview ongoing work on wave scheduling (Section IV).

II. Model

In this section, we describe our model for sensor networks and sensor data, and outline our architectural assumptions.

Sensor Networks. We consider a sensor network that consists of a large number of *sensor nodes* connected through a multi-hop wireless network [13], [8]. We assume that nodes are stationary, all node radios have the same fixed communication range[1], and that each node is aware of its own location. Sensor networks have the following physical resource constraints:

Communication. The bandwidth of wireless links connecting sensor nodes is usually limited, on the order of a few hundred Kbps; the network provides limited quality of service, with variable latency and large packet drop probability.

Power consumption. Sensor nodes have limited supply of energy; thus, energy-efficiency is a major design consideration.

Computation. Sensor nodes have limited computing power and memory sizes that restrict the types of data processing algorithms that can be deployed and intermediate results that can be stored on the sensor nodes.

Sensor Data. Each sensor can be viewed as a separate data source that generates structured records with several fields such as the id and location of the sensor that generated the reading, a time stamp, the sensor type, and the value of the reading. (We assume that some of the signals might have been postprocessed by a signal processing layer.) Conceptually, we view the data distributed throughout the sensor network as forming a distributed database system consisting of multiple tables with different types of sensor data.

Queries and View Nodes. The sensor network is programmed through declarative queries which abstract the functionality of a large class of applications into a common interface of expressive queries. Our work does not depend on any specific query language; instead it applies to any query processing strategy that performs in-network processing by collecting data from multiple sensors onto a designated subset of the nodes that we call the *view nodes*. The view nodes may either store directly unprocessed sensor readings or materialize the result of more complex processing over sensor readings.

Synchronization Between Sensors. We assume that the clocks of neighboring nodes in the sensor network are reasonably synchronized, either through GPS or through distributed time synchronization algorithms (e.g., [9], [2]).

[1]Note that future generations of nodes might have variable-range radios; an extension of this work to variable-range radios is future work.

III. View selection

As in a centralized database system, the contents of a view are defined through a user-defined query. It is our goal to *automatically* select the best views (and view nodes) in the sensor network in order to optimize the overall cost of a *query workload*. Our use of views in sensor networks follows a *hybrid pull-push* model in which the sensor data is processed inside the network and *pushed* to view nodes where the data is stored. Queries are routed to relevant view nodes from which the requested data is *pulled* to assemble the query answer.

While automated view and index selection algorithms have been proposed for relational databases [1], the view selection problem for sensor networks is much more complex: in addition to deciding *what* view to materialize, we also need to decide *where* the view should be stored. As we will discuss below, view content and location have complex interactions.

We consider a set of sensor nodes $\mathtt{n_1, \ldots, n_k}$ spread in a plane. We assume that time is divided into *periods*, that queries can only be executed at the end of a period, that queries refer to readings generated during that period, and that a sensor node generates one reading within that period. Let $\mathtt{u_i}$, $\mathtt{i = 1, \ldots, k}$, be the probability that node $\mathtt{n_i}$ generates a reading within a period[2]. A *query workload* $\mathtt{W}$ is a set of tuples $\mathtt{W = \{< Q_1, p_1 >, \ldots, < Q_n, p_n >\}}$, where $\mathtt{p_i}$ is the probability that query $\mathtt{Q_i}$ is asked during a period. Each query $\mathtt{Q_i}$ returns the aggregate value of an attribute $\mathtt{A}$ over a subset of the sensor nodes $\mathtt{S_i}$ for the preceding period. We assume that the aggregate function used is the same for all queries.

We consider a tree having as leaves the data sources $\mathtt{n_1, \ldots, n_k}$ and as root the server where users present their queries. It is possible to forward data up the tree proactively and materialize partial aggregate results in selected intermediate tree nodes called *view* nodes; we refer to such messages as *view update* messages. During each period, a set of queries is posed. Query evaluation happens at the end of the period, and proceeds in two phases. First, *request* messages characterizing the set of queries posed in the period are forwarded down the tree until they reach all the view nodes required to answer the queries. Then, partial query results are sent up from view nodes. The results are combined at intermediate nodes, and eventually reach the root. The cost of computing partial aggregates is negligible compared to the cost of sending messages along the edges of the tree, so the total cost of the tree is the sum of the costs of all edges. Our objective is to minimize this expected cost.

For an edge along which view update messages are sent proactively, the expected cost is determined by the probability of new sensor readings being generated in the subtree beneath it. For an edge between a view node and the root, the expected cost is the cost of a request message plus the cost of partial result messages needed to answer the currently posed queries. We assume a request message must be sent to a view even if no partial result is required for the current set of queries. This is because radio receivers have substantial power requirements — given a contention resolution MAC layer with the possibility of hidden terminals, the energy cost (at the listener) of determining that no message will arrive can be substantially more than the energy cost (at sender and listener) of transferring a short "nothing to send" message.

[2]In this exposition, we assume that sensors are independent even though this is clearly not the case.

Studying special instances of the view selection problem given a tree structure is the focus of current research. In particular, we consider the following design space:

- All queries and data updates occur with probability 1.
- Sensor updates have probability 1, but queries occur with arbitrary probabilities.
- Queries occur with probability 1, but sensor updates occur with arbitrary probabilities.
- Both sensor updates and queries occur with arbitrary probabilities.

Different query probabilities can result in different optimal solutions to the view selection and placement problem.

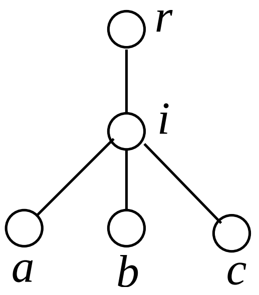

Fig. 1. A dissemination tree for view selection.

Example: Consider the dissemination tree shown in Figure 1. Sensors are at the leaf nodes a, b, and c; each sensor generates a new data value in each period. There are two queries, the sums $(a + b)$ and $(a + c)$. These queries are issued independently, each with probability $(1 - \epsilon)$. Two limiting cases should be clear. If we set $\epsilon = 0$ (so both queries are issued with probability 1) the optimal solution is to proactively forward everywhere: values of sensors a, b, and c are sent from the leaves to the intermediate node i, and values of the queries $(a + b)$ and $(a + c)$ are sent to the root r. No request messages are sent at all. If ϵ is near 1, the optimal solution is to flood the tree with a 1-bit request message, and in the (extremely unlikely) case that either of the queries is issued to send query results to the root as in the previous case. Now, let q_1, q_2 and r be the cost of a 1-bit query request message, a 2-bit request message, and a result message, respectively. It is beneficial to send a request message from the root (rather than unconditionally sending query results to it) if $q_2 + 2(1-\epsilon)r < 2r$. Similarly, it is beneficial to send a request message from node i to node b or c if $q_1 + (1-\epsilon)r < r$, and it is beneficial to send a request message to node a if $q_1 + (1-\epsilon^2)r < r$. So a suitable choice of ϵ makes it beneficial to send request messages to b and c, but to materialize a in a view at i. Under the assumption that q_2 is strictly less than $2q_1$ (i.e., the per-packet overhead is not identically 0), ϵ can be chosen so it is beneficial to send a request message from the root, but not along any of the other three edges; in this case, the optimal solution materializes both queries at i and requests them from the root.

This example illustrates that query probabilities affect the optimal choice of views. Similar examples can be given to show the effect of data update probabilities. Finally, the behavior is affected by the choice of aggregate function: AVG and MIN behave quite differently because the MIN operator has no inverse.

Due to space constraints, we only summarize our results here:

- We can show that the general problem is NP-complete through a reduction from the Set Basis Problem [4].
- We can give dynamic programming algorithms (with at least exponential worst-case complexity) for the complete design space above.

Because of the complexity of the dynamic programming algorithms, we are currently looking into approximation algorithms for this problem, and we are working on an implementation to obtain experimental results.

IV. Wave Scheduling

We now present *wave scheduling*, a class of simple activation schedules and associated routing protocols that achieve scalability and energy-efficiency with modest delay penalties.

We assume that the area is divided into a *grid* of square *cells*. The size of each cell is set so that a node anywhere in a cell can communicate directly with nodes in any of its four horizontal and vertical neighbor cells. This constrains the side of a cell to have length at most $r/\sqrt{5}$, where r is the transmission range of a node. In such a grid it can be shown that a (rectilinear) path between any two nodes is at most a factor of $\sqrt{10}$ more hops than the optimal (non-grid) path.

We assume initially that each grid cell is occupied by exactly one node: Our technique is layered on top of a protocol like GAF [16], which periodically elects a single representative node for each nonempty cell. This achieves significant power savings (only representative nodes expend energy on inter-cell message routing), and provides some fault-tolerance as well. Of course, a few cells may be empty, but we omit discussing the treatment of such "holes" due to space constraints.

In summary, our goal is to compute a periodic activation schedule and an associated routing scheme for nodes arranged in a rectilinear grid, with a modest number of view nodes.

Tree Scheduling. Consider first a network with only a single view server. The edges of an optimal activation schedule form a spanning tree. A natural schedule for such a tree simply activates edges in reverse order of their distance (in the tree) from the view server, enabling a message to propagate from any leaf of the tree to the view node in a single scheduling period. Routing in a tree is trivial: each non-view node forwards every message it receives to its parent. We note that this use of a tree to route messages from sensor nodes to a specific server is not new. For example, it is a key component of the TAG method for handling aggregate queries [12].

The above discussion ignores the effect of interference between edges, which could arise in the "bottom-up" schedule owing to the simultaneous activation of edges that are within collision range of one another. In fact, the immediate children of a tree node, which are always activated together, are certain to be within collision range. Thus, even when there is only a single view, this approach demands an effective MAC protocol.

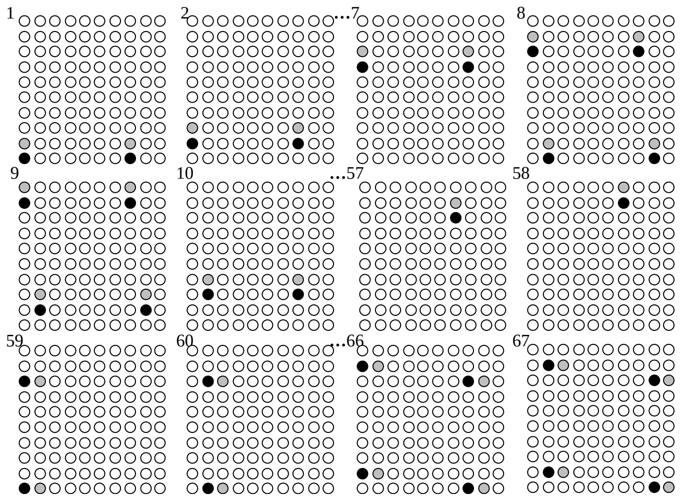

Fig. 2. An illustration of the SimpleWave schedule in a 10 × 10 grid. The first two rows depict the edge activations in the first phase (north). The last row depicts the edge activations at the start of the next phase (east). The remaining phases that complete the schedule can be similarly drawn. In each picture, the darkly shaded node is a sender, while a lightly shaded node is a receiver. The minimum distance of seven nodes between two senders simultaneously transmitting is computed to ensure non-interfering transmissions.

We next consider the more realistic case of a network with multiple view servers. To generalize tree scheduling to handle this case, we construct a forest containing one spanning tree rooted at each of the view servers. An edge activation schedule for the entire forest can then be derived from schedules for the individual trees in several ways. At one extreme is a *conservative* schedule, which is simply a concatenation of schedules for the individual trees, activating edges of each spanning tree in succession. At the other extreme, an *aggressive* schedule activates all the trees in parallel. Neither scheme scales well. With a conservative schedule, message latency grows linearly with the number of views. With an aggressive schedule, energy consumption grows linearly. In addition, an aggressive schedule tends to generate many more collisions, further increasing energy consumption (due to message retransmissions) and reducing network capacity.

Wave Scheduling. With the above motivation, we can now describe our *wave scheduling* technique, by which we avoid the scaling problems inherent in tree scheduling. Recall that tree scheduling handles multiple destination view nodes by computing a separate activation schedule for each view and then combining the schedules. Scaling problems arise because there is no obvious way to combine schedules without increasing either the period or the collision frequency.

To avoid these problems, we can compute a single "general-purpose" schedule, in which every edge of the network is activated exactly once per period, and which is guaranteed to have no collisions. Since every edge is activated infinitely often, it is always possible to route a message between any connected pair of nodes using such a schedule.

Unfortunately, even though a path can be followed in principle, its latency may be unacceptably high if the path and activation schedule do not “fit” together well. For example, suppose a path enters node n along edge e_1 and leaves it along e_2. Each message arriving along e_1 must be queued at n until the next time e_2 is scheduled. If e_2 is activated just *before* e_1 in the schedule, the message must wait nearly a full period in n’s queue before it can be forwarded (during the next iteration of the schedule). In the worst case, this phenomenon occurs at *every* node along the path. The resulting message latency (the product of the path length and scheduling period) is unacceptable for most applications.

Thus, we seek an activation schedule and associated routing algorithm that yield a “reasonably” low-latency path from any source node to any view server. Wave Scheduling is our proposed solution.

Periodic Activation Schedules. In a wave schedule, horizontal and vertical communication edges are activated in a periodic sequence of *phases*. Each phase has a direction – north, east, south or west – along which a “wave” of messages traverses the grid for some number of steps. For example, in a north-going phase there is a pattern of non-interfering north-going edges, containing at least one edge in each column (assuming the sensor network contains a large number of cells). The edges are activated simultaneously, then the entire pattern shifts north by one cell, wrapping around between the north and south edges of the grid as necessary to maintain the integrity of the pattern. This process is repeated one or more times for the duration of the phase. The east, south and west waves are scheduled analogously.

The preceding framework admits a number of different activation schedules. Due to space constraints, we only mention one schema that we call *SimpleWave*, which is illustrated in Figure 2. Due to space constraints, we omit the description of the impact of wave scheduling on routing, but we would describe this interaction in the presentation at the workshop.

V. Experimental Analysis

We have extensive simulation studies of our wave schedules; the experiments show that wave scheduling provides a graceful tradeoff between latency and energy usage. Due to space constraints, we omit relevant performance graphs, but we will present them at the workshop. Our plans for future work include an experimental evaluation of view selection algorithms.

VI. Related Work

Our work extends research on data storage in sensor networks [14], [5], [3], query processing for sensor network [12], [10], [7], [11], and efficient data dissemination [8], [6], [17]. Tree-based routing [12] provides an alternative to our wave scheduling scheme, but it is subject to interference between edges, and thus becomes unusable when the number of view servers increases above a small fraction of the nodes. An energy-efficient MAC protocol called S-MAC has been proposed in [18]. GAF (Geographical Adaptive Fidelity) [16], [15] is an algorithm that also conserves energy by identifying nodes that are equivalent from a routing perspective and then turning off unnecessary nodes. Our wave scheduling protocol is orthogonal and synergistic to GAF.

VII. Conclusion and Future Work

We introduced the problems of view selection and node scheduling in a sensor network, and presented a high-level description of our approach to solving them. In future work, we plan to investigate the interaction between these two problems. We are interested in exploring efficient wave schedules given specific message generation patterns and view locations. Another interesting direction is to study fault-tolerance in the context of materialized views and scheduled data propagation.

References

[1] Surajit Chaudhuri and Vivek R. Narasayya. Autoadmin ’what-if’ index analysis utility. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 367–378, 1998.

[2] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, pages 147–163, December 2002.

[3] Deepak Ganesan, Deborah Estrin, and John Heidemann. DIMENSIONS: Why do we need a new data handling architecture for sensor networks? In *Proceedings of the ACM Workshop on Hot Topics in Networks*, Princeton, NJ, USA, October 2002. ACM.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[5] Abhishek Ghose, Jens Grossklags, and John Chuang. Resilient data-centric storage in wireless ad-hoc sensor networks. In *Proceedings of the 4th International Conference on Mobile Data Management MDM 2003*, pages 45–62, 2003.

[6] John S. Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building efficient wireless sensor networks with low-level naming. In *Symposium on Operating Systems Principles*, pages 146–159, 2001.

[7] Joseph M. Hellerstein, Wei Hong, Samuel Madden, and Kyle Stanek. Beyond average: Towards sophisticated sensing with queries. In *2nd International Workshop on Information Processing in Sensor Networks (IPSN ’03)*, page to appear, 2003.

[8] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. pages 56–67. ACM SIGMOBILE, ACM Press, 2000.

[9] C. Liao, M. Martonosi, and D. Clark. Experience with an adaptive globally-synchronizing clock algorithm. In *Proceedings of the 11th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 106–114, June 1999.

[10] Sam Madden and Michael J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *ICDE*, 2002.

[11] Samuel Madden, Michael J. Franklin, Joseph Hellerstein, and Wei Hong. The design of an acquisitional query processor for sensor networks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2003.

[12] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.

[13] G. J. Pottie and W. J. Kaiser. Embedding the Internet: wireless integrated network sensors. *Communications of the ACM*, 43(5):51–51, May 2000.

[14] Sylvia Ratnasamy, Deborah Estrin, Ramesh Govindan, Brad Karp, Scott Shenker, Li Yin, and Fang Yu. Data-centric storage in sensornets. In *First Workshop on Hot Topics in Networks (HotNets-I) 2002*, 2002.

[15] Ya Xu, Solomon Bien, Yutaka Mori, John Heidemann, and Deborah Estrin. Topology control protocols to conserve energy inwireless ad hoc networks. Technical Report 6, University of California, Los Angeles, Center for Embedded Networked Computing, January 2003. submitted for publication.

[16] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, pages 70–84, 2001.

[17] Fan Ye, Haiyun Luo, Jerry Cheng, Songwu Lu, and Lixia Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2002.

[18] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the IEEE Infocom*, pages 1567–1576, 2002.