The Cougar Project: A Work-In-Progress Report

Alan Demers,* Johannes Gehrke, Rajmohan Rajaraman, Niki Trigoni and Yong Yao†

ABSTRACT

We present an update on the status of the Cougar Sensor Database Project, in which we are investigating 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 give an overview of 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 problems in energy-efficient routing and in-network aggregation and some thoughts on how to approach them.

1. INTRODUCTION

A powerful paradigm in sensor network design 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 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 (1) carefully determining the data that should be stored in designated view nodes, and (2) coordinating the data dissemination to these nodes.

In this paper we overview several 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.

A second, related research direction is *aggregation tree selection*. Processing an aggregation query requires that data from a set of sensor nodes be routed to the site where the query was posed, with in-network aggregation on the route. The most natural way is along edges of a spanning tree that in some sense "embeds" the query plan. The feasibility and cost benefit of in-network aggregation depends in subtle ways on the choice of the routing tree.

Our third research 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. Routing protocols can be modified to interact symbiotically with the scheduling decisions, resulting in significant energy savings at the cost of higher latency.

*Corresponding author.

†A. Demers, J. Gehrke, N. Trigoni and Y. Yao are with the Department of Computer Science, Cornell University, Ithaca NY 14853, email: {ademers, johannes, niki, yao}@cs.cornell.edu. R. Rajaraman is with the College of Computer and Information Science, Northeastern University, Boston, MA 02115, email: rraj@ccs.neu.edu

2. 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 [20, 34]. 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 high packet loss rates.

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, the time stamp, the sensor type, and the value of the reading. Conceptually, the data distributed throughout the sensor network forms 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 applies to any query processing strategy that performs in-network processing by collecting data from multiple sensors onto a designated subset of nodes that we call the *view nodes*. The view nodes may either store the unprocessed sensor readings directly or materialize the result of more complex processing over them.

Synchronization Between Sensors. We assume that the clocks of neighboring nodes in the sensor network are reasonably synchronized, either through GPS or through

[1]Note that future generations of nodes might have variable-range radios; future extensions of this work will deal with variable-range radios.

distributed time synchronization algorithms (e.g., [7, 26]).

Embedding on a Grid. 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 the four horizontally and vertically neighboring cells. This constrains the size 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 shortest (rectilinear) path between any two nodes is at most a factor of 4 more hops than the optimal (non-grid) path. We assume that each grid cell is occupied by exactly one node.

The above assumption can be realized by layering our techniques on top of a protocol like GAF [41], which periodically elects a single representative node for each nonempty grid 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, some cells may be entirely empty. The treatment of such "holes" is an important consideration which we omit here due to space constraints.

3. 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 [6], 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 $\mathtt{n}_1, \ldots, \mathtt{n}_k$ spread in a plane. We assume that time is divided into *periods*, that queries can be executed only 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 $\mathtt{u}_i$, $\mathtt{i} = 1, \ldots, \mathtt{k}$, 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 $\mathtt{W} = \{< \mathtt{Q}_1, \mathtt{p}_1 >, \ldots, < \mathtt{Q}_n, \mathtt{p}_n >\}$, 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 $\mathtt{n}_1$, ..., $\mathtt{n}_k$ and as root the server where users present their queries. During each period, a set of queries is posed. Query evaluation, which happens at the end of the period, conceptually involves three phases. First, data from some sensors is forwarded proactively up the tree, to the selected view nodes, and partial aggregate results are materialized there. In the next phase, *request* messages characterizing the set of queries posed in the period are forwarded down the tree until they reach all the view and sensor nodes required to answer the queries. Finally, partial query results are sent upward. The results may be 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 message costs along all edges. Our objective is to minimize this expected cost.

[2]For simplicity of exposition, we assume that sensors are independent even though this is often not the case.

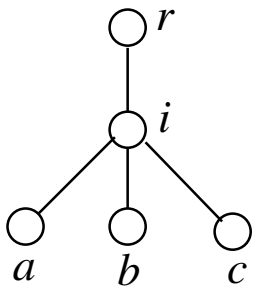

Figure 1: A dissemination tree for view selection.

Tree edges are classified as either *proactive* or *on-demand*. Along a proactive edge, all new data is sent upward unconditionally in every period. Thus, there is no need to send request messages down proactive edges. In contrast, an on-demand edge transfers only the data required to answer the queries posed in the current round. Thus, an on-demand edge requires an explicit request message in each period. The request message must be sent even if no partial result is required in the current period. This is a consequence of a (realistic) energy model in which radio receivers have substantial power requirements. In the presence of collisions at the MAC layer and imperfect clock synchronization, the energy cost (at the listener) of determining that no message will arrive in a period can be substantially more than the energy cost (at sender and listener) of transferring a short "nothing to request" message.

For a proactive edge, the per-period expected cost is determined by the probability of new sensor readings being generated in the subtree beneath it. For an on-demand edge, the expected cost is the cost of its (unconditional) request message plus the cost of partial result messages needed to answer the currently posed queries.

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.

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 aggregate queries, the sums $(a+b)$ and $(a+c)$. These queries

are issued independently, each with probability $(1-\epsilon)$. Let q be the cost of a sending query request message down a tree edge, and $r >= q$ the cost of sending a data result message.

Two limiting cases are easy to analyze. First, if we set $\epsilon = 0$ (so both queries are issued with probability 1) the optimal solution is to proactively forward everything: values of sensors a, b, and c are sent from the leaves to the intermediate node i, where the values of the queries $(a+b)$ and $(a+c)$ are computed and sent to the root. Every edge is proactive, so no request messages are sent at all, and the total cost is $5r$ per period – one data message sent from each sensor node to node i, and two data messages from node i to the root.

The second limiting case occurs when ϵ is nearly 1, so that in most periods no query is issued at all. In this case, proactively transmitting data along any edge wastes energy with high probability, since the data is unlikely to be needed. Thus, the optimal solution must make every edge on-demand, flooding the tree with request messages and sending the requested data up the tree as in the previous case. The expected per-period cost of this solution is obtained by summing the costs of the edges. The expected cost of the edge between the root and interior node i is $q + 2(1-\epsilon)r$, representing an unconditional request message and two result messages (one for each query) with independent probabilities $(1-\epsilon)$. By symmetry, the edges entering nodes b and c have equal expected costs of $q + (1-\epsilon)r$. Finally, the edge entering node a, which is shared by both queries, has expected cost $q + (1-\epsilon^2)r$. It is clear that when ϵ is sufficiently near 1 no edge can be made proactive without strictly increasing the cost of the solution.

Finally we consider intermediate cases in which the tree is neither entirely proactive nor entirely on-demand. By comparing the expected per-edge costs for the two limiting-case solutions above, we can see that it is beneficial to send a request message from the root to node i (that is, to make the edge on-demand) when $q < 2\epsilon r$. Similarly, it is beneficial to send request messages to nodes b and c when $q < \epsilon r$; and it is beneficial to send a request to node a when $q < \epsilon^2 r$.

For this simple example, the optimal solutions are now completely determined by the relative values of q, r and ϵ. When $q > 2\epsilon r$ (the first limiting case above), a completely proactive solution is best. For $\epsilon r < q < 2\epsilon r$, the best solution is to make on-demand *only* the edge from the root to node i, proactively sending data from the leaves to node i and materializing the two queries there. For $\epsilon^2 r < q < \epsilon r$ it becomes beneficial to make the edges to nodes b and c on-demand as well, but still materialize the value of a at node i. And for $q < \epsilon^2 r$ (the second limiting case above), a completely on-demand solution is optimal.

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. For example, $MIN\{a,b,c\}$ is completely determined by $MIN\{a,b\}$ and $MIN\{a,c\}$, while this is not true of AVG.

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 [10].
- 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.

4. AGGREGATION TREE SELECTION

In the previous section, we studied the problem of selecting views and view locations in a given aggregation tree, such that the total communication needed for processing a set of given aggregate queries is minimized. The effectiveness of the solution depends not only on the particular view selection, but also on the choice of the aggregation tree. In this section, we study *tree selection*, the problem of computing an optimal tree in the underlying sensor network that connects the root with the set of active sensors.

We consider two performance criteria for determining the quality of an aggregation tree, both based on the measure of energy consumed. One criterion is to minimize the *total energy consumed* for processing the given query set. Another criterion is to minimize the *maximum energy consumed at a node*, aimed at maximizing the network lifetime (where the lifetime is defined as the time until the first node dies).

Before we discuss the challenges in tree selection, we present our model for energy consumption. We assume that the energy consumed in communicating b bits on a link is of the form $\alpha + \beta b$, where $\alpha > 0$ is a fixed cost associated with every message exchange and β is the per-bit transmission and reception cost. Both the parameters α and β are sums of two components each, $\alpha_s + \alpha_r$ and $\beta_r + \beta_s$, respectively, where α_s (resp., β_s) is the part of the fixed cost (resp., per-bit cost) associated with the sender and α_r (resp., β_r) is the part associated with the receiver. In the following, we make the simplifying (if somewhat unrealistic) assumption that $\alpha = 0$; that is, the energy consumed in transferring b bits on a link is proportional to b, so that minimizing the number of bits transferred will minimizie the total energy consumed.

As discussed in Section 2, we assume that the nodes of the sensor network are organized in a grid. In such a network, the general problem of selecting a tree that minimizes the total energy consumed is intractable: an easy reduction from the NP-complete rectilinear Steiner tree problem [9] shows that tree selection is NP-hard even for the case of a single query. In the remainder of this section, we present a series of examples that indicate how the effectiveness of aggregation trees varies with problem instances and the performance criteria.

A given instance specifies an underlying sensor network G with a designated root r and a set Q of queries covering a set S of sensors. For a given tree T that connects every sensor node included in a query in Q to the root, the total cost (or the maximum cost) depends on the particular view selection algorithm used over T since different solutions may differ in the amount of communication along the tree links. Here we assume that the total cost (resp., maximum cost) is that of an optimal view selection algorithm.[3]

[3]Another natural notion for the total cost of a tree is to associate with each query q a separate cost $C_q(T)$ that is defined as the size of the subtree of T that contains the unique paths

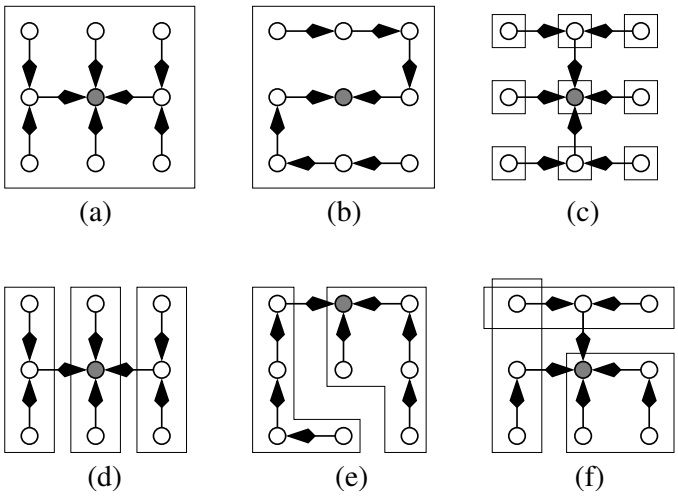

Figure 2: Aggregation trees and query set instances. Each of the six figures illustrates a query set instance and an aggregation tree. In each figure, a query is represented by the set of nodes within a bounding box and the aggregation tree is shown with edges directed toward the root, which is shaded gray. For reference, we label the query sets and trees in parts (a) through (f) by Q_a through Q_f and T_a through T_f, respectively.

Example: Consider a sensor network consisting of a 3×3 grid with 9 sensor nodes. We let (i, j) denote the node in row i, column j, $0 \leq i < 3$, $0 \leq j < 3$. Suppose the root server is the center $(1, 1)$. We consider different query set instances, in each of which all the sensor nodes are active; thus the desired tree is, in fact, a spanning tree. We assume that all the update probabilities and the query probabilities are 1.

1. Q_a contains the single set of all nodes. That is, the only query of interest is an aggregate of all the nodes. In this case, every spanning tree has the same total cost and is hence optimal. In particular, the tree T_a consisting of all the vertical edges together with the horizontal edges on row 1 is an optimal tree (see Figure 2(a)).

2. Q_b is the same as Q_a, but the performance criterion is different; it is network lifetime. Tree T_a is no longer optimal. In a "snakelike" tree T_b (see Figure 2(b)), each node sends and receives at most one message and the root receives two messages, while in T_a, certain intermediate nodes receive two messages and send one message and the root receives four messages. The exact benefit of T_b over T_a depends on the ratio of β_s to β_r.

3. Q_c contains all the singleton sets. That is, every query seeks the data on a single node. In this case, a tree with optimal total cost is a shortest path tree. Thus, tree T_a has optimal total cost. The same can be said for tree T_c consisting of all the horizontal edges together with the vertical edges on column 1 (see Figure 2(c)). Both the trees have a cost equal to the sum of distances from each node to the center. In the given network, it is 12; for a general $N \times N$ grid the cost is $\Theta(N^3)$. Clearly, the snakelike tree T_b is very poor, incurring a total cost of $\Theta(N^4)$.

4. Q_d equals $\{\{(i, j) : 0 \leq i < 3\} : 0 \leq j < 3\}$. That is, every query is an aggregate on a column. In this case, one can see that tree T_d (same as T_a) is a tree with optimal total cost 8 (see Figure 2(d)) while tree T_c has total cost 12 and is suboptimal.

5. In all the above instances, a tree with optimal total cost is a shortest path tree. However, this is not necessary. Suppose the root is at node $(0, 1)$ and the query set Q_e contains the query $\{(0, 0), (1, 0), (2, 0), (2, 1)\}$ and a set consisting of each of the other nodes (see Figure 2(e)). For this instance, in a tree T_e with optimal total cost, the node $(2, 1)$ is connected to the root $(0, 1)$ via the long path $(2, 1) \rightarrow (2, 0) \rightarrow (1, 0) \rightarrow (0, 0) \rightarrow (0, 1)$.

6. As a final example, we consider a query set Q_f with root $(1, 1)$ in which the queries are not disjoint (see Figure 2(f)). For this instance, the tree T_a has total cost 10. This is because the bases of the projection of the queries on to columns 0 and 2 are both of size two; consequently, the two horizontal edges on row 1 have to carry 2 information units, one corresponding to a column and the other corresponding to a node in row 0. In contrast, tree T_f has a cost of 9 only because only one subtree rooted at a child of the root has a basis of size more than one; in particular, only the vertical edge $((0, 1), (1, 1))$ carries two units of information, while every other edge carries one.

The above examples indicate that the optimal trees have diverse characteristics, depending on the particular query instance and the performance criterion being considered, even for the special case when the set of active sensors includes all the nodes.

To close this section, we note that it is not necessary to restrict our attention to trees; indeed, when queries are not disjoint the generalization to DAGs can be beneficial. It is not difficult to generalize the example of T_f above so that a DAG in which the shared sensor value is sent to two neighbors yields a lower cost solution than any tree. Aggregation DAGs are the subject of future research.

5. 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.

As discussed in Section 2, we assume that the nodes of the sensor network are organized in a grid with only nearest-neighbor communication allowed. Our goal is to compute a periodic edge activation schedule and an associated routing scheme for nodes arranged in such a grid, where the number of destinations (sites at which queries are posed) is modest.

from every node in q to the root r. Then, the cost of the tree T is simply $\sum_{q \in Q} C_q(T)$. While this notion of cost may be easier to analyze, it does not take into account optimizations that can be done when the projection of queries within a subtree are linearly dependent (see Section 3 for examples)

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 [30].

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.

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.

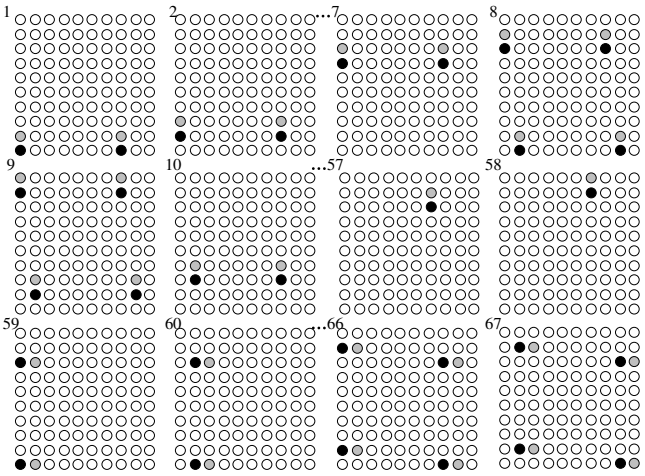

Figure 3: An illustration of the SimpleWave schedule in a 10 × 10 grid. The first two rows depict the edge activations in the first phase (north). The last row depicts the edge activations at the start of the next phase (east). The remaining phases that complete the schedule can be similarly drawn. In 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.

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. One of these, which we call *SimpleWave*, is illustrated in Figure 3. For each activation schedule, we can devise routing protocols that are biased towards minimizing energy, latency, or some combination.

6. RELATED WORK

Query Processing in Sensor Networks. Several research groups have focused on in-network query processing as a means of reducing energy consumption. The TinyDB Project at Berkeley investigates query processing techniques for sensor networks including an implementation of the system on the Berkeley motes and aggregation queries [29, 30,

31, 27, 19]. An acquisitional approach to query processing is proposed in [28], in which the frequency and timing of data sampling is discussed. The sensor network project at USC/ISI group [20, 17, 16] proposes an energy-efficient aggregation tree using data-centric reinforcement strategies (directed diffusion). A two-tier approach (TTDD) for data dissemination to multiple mobile sinks is discussed in [43]. In a recent study [12], an approximation algorithm has been designed for finding an aggregation tree that simultaneously applies to a large class of aggregation functions.

There has been a great deal of work on query processing in distributed database systems – a recent survey appears in [23]. However, there are major differences between sensor networks and traditional distributed database systems. Most relevant to sensor networks is existing work on distributed aggregation [38, 42], but these approaches do not consider the physical limitations of sensor networks. Madden et al. [30] give an extended classification of aggregate operators with properties relevant to sensor network aggregation.

View Management in Sensor Networks. Our high-level framework for distributed storage builds on the data-centric storage model proposed in [36, 37] and extended in [11]. Long-term storage in sensor networks is combined with multi-resolution data access and spatiotemporal data mining in [8].

There has been a lot of work on view design, maintenance and exploitation in centralized database systems. Chaudhuri et al. [6, 2] consider the automated selection of materialized views and indices given a query workload and discuss their implementation on Microsoft SQL Server. Several variations of the view selection problem have been extensively studied in the context of data warehouses: although the general goal is to select views that would minimize the query response time, the constraints under consideration is either the materialization time, the storage space or both [14, 3, 15, 22]. Kotidis et al. propose a dynamic view management system for data warehouses (DynaMat [24]), which unifies the view selection and the view maintenance problems under a single framework. Algorithms for query rewriting and making efficient use of existing materialized views to speed up query processing are discussed in [25, 35, 13].

Routing and MAC Layers in Sensor Networks. The data management layer should not be considered in isolation from the communication layers. For instance, an opportunity for cross-layer optimization is to design and adapt communication protocols to the particular communication needs of the data management layer. A number of protocols for ad-hoc networks have been proposed in the literature [32, 21, 4, 33]. Recent work on energy-aware routing proposes the selection of routes on the basis of available energy in order to increase network lifetime [5, 46, 34]. Heinzelman et al. present the SPIN family of network protocols for communication of large messages in sensor networks [18]. The PAMAS MAC-level protocol turns radios off when they are not transmitting or receiving packets [39]. TDMA protocols reduce the duty cycle thus trading idle-time energy consumption for latency [34]. Our wave scheduling approach achieves significantly greater energy savings by coordinating the radio usage across the sensor network.

An energy-efficient MAC protocol called S-MAC has been proposed in [44, 45], where the nodes are locally synchronized to follow a periodic listen and sleep scheme. GAF (Geographical Adaptive Fidelity) [41, 40] 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.

7. CONCLUSION AND FUTURE WORK

We introduced the problems of view selection, aggregation tree 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 three problems. We are interested in exploring efficient wave schedules given specific message generation patterns, view locations, and aggregation trees. Another interesting direction is to study fault-tolerance in the context of materialized views and scheduled data propagation.

Acknowledgements. We are grateful to Douglas Holzhauer and Zen Pryke for helpful discussions. This work is supported by NSF Grants CCR-0205452, IIS-0133481, and IIS-0330201, by the Cornell Information Assurance Institute, and by Lockheed Martin.

8. REFERENCES

[1] ACM SIGMOBILE. *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM-98)*. ACM Press, 1998.

[2] S. Agrawal, S. Chaudhuri, and V. R. Narasayya. Automated selection of materialized views and indexes for SQL databases. In *Proceedings of 26th International Conference on Very Large Data Bases*, pages 496–505, 2000.

[3] E. Baralis, S. Paraboschi, and E. Teniente. Materialized views selection in a multidimensional database. In *The VLDB Journal*, pages 156–165, 1997.

[4] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. [1], pages 85–97.

[5] J.-H. Chang and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *Proceedings of the 2000 IEEE Computer and Communications Societies Conference on Computer Communications (INFOCOM-00)*, pages 22–31, Los Alamitos, Mar. 26–30 2000. IEEE.

[6] S. Chaudhuri and V. R. Narasayya. Autoadmin 'what-if' index analysis utility. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 367–378, 1998.

[7] 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.

[8] D. Ganesan, D. Estrin, and J. 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.

[9] M. Garey and D. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32:826–834, 1977.

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

[11] A. Ghose, J. Grossklags, and J. 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.

[12] A. Goel and D. Estrin. Simultaneous optimization for concave costs: Single sink aggregation or single source buy-at-bulk. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, January 2003.

[13] J. Goldstein and P.-A. Larson. Optimizing queries using materialized views: a practical, scalable solution. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 331–342, 2001.

[14] H. Gupta. Selection of views to materialize in a data warehouse. In *ICDT*, pages 98–112, 1997.

[15] H. Gupta and I. S. Mumick. Selection of views to materialize under a maintenance cost constraint. *Lecture Notes in Computer Science*, 1540:453–470, 1999.

[16] J. Heidemann, F. Silva, Y. Yu, D. Estrin, and P. Haldar. Diffusion filters as a flexible architecture for event notification in wireless sensor networks. Technical Report ISI-TR-556, USC/Information Sciences Institute, April 2002.

[17] J. S. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *Symposium on Operating Systems Principles*, pages 146–159, 2001.

[18] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. pages 174–185. ACM SIGMOBILE, ACM Press, 1999.

[19] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Towards sophisticated sensing with queries. In *2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)*, page to appear, 2003.

[20] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. pages 56–67. ACM SIGMOBILE, ACM Press, 2000.

[21] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353 of *The Kluwer International Sereies in Engineering and Computer Science*. Kluwer Academic Publishers, 1996.

[22] P. Kalnis, N. Mamoulis, and D. Papadias. View selection using randomized search. *Data Knowledge Engineering*, 42(1):89–111, 2002.

[23] D. Kossmann. The state of the art in distributed query processing. *Computing Surveys*, 32, 2000.

[24] Y. Kotidis and N. Roussopoulos. DynaMat: a dynamic view management system for data warehouses. pages 371–382, 1999.

[25] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 95–104, San Jose, Calif., 1995.

[26] 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.

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

[28] S. Madden, M. J. Franklin, J. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2003.

[29] S. Madden and J. M. Hellerstein. Distributing queries over low-power wireless sensor networks. In *ACM SIGMOD Conference*, 2002.

[30] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.

[31] S. R. Madden, R. Szewczyk, M. J. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc sensor networks. In *Workshop on Mobile Computing and Systems Applications (WMCSA)*, 2002.

[32] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, Aug. 1994.

[33] C. E. Perkins. Ad hoc on demand distance vector (aodv) routing. Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-04.txt, October 1999.

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

[35] R. Pottinger and A. Y. Levy. A scalable algorithm for answering queries using views. In *The VLDB Journal*, pages 484–495, 2000.

[36] S. Ratnasamy, D. Estrin, R. Govindan, B. Karp, S. Shenker, L. Yin, and F. Yu. Data-centric storage in sensornets. In *First Workshop on Hot Topics in Networks (HotNets-I) 2002*, 2002.

[37] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. Ght: A geographic hash table for data-centric storage. In *First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

[38] A. Shatdal and J. F. Naughton. Adaptive parallel aggregation algorithms. In M. J. Carey and D. A. Schneider, editors, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 104–114, San Jose, California, 22–25 May 1995.

[39] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. [1], pages 181–190.

[40] Y. Xu, S. Bien, Y. Mori, J. Heidemann, and D. 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.

[41] Y. Xu, J. Heidemann, and D. 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.

[42] W. P. Yan and P.-Å. Larson. Eager aggregation and lazy aggregation. In U. Dayal, P. M. D. Gray, and S. Nishio, editors, *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases*, pages 345–357, Zurich, Switzerland, 11–15 Sept. 1995. Morgan Kaufmann.

[43] F. Ye, H. Luo, J. Cheng, S. Lu, and L. 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.

[44] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the IEEE Infocom*, pages 1567–1576, 2002.

[45] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated, adaptive sleeping for wireless sensor networks. Technical Report ISI-TR-567, USC/Information Sciences Institute, January 2003.

[46] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical Report UCLA/CSD-TR-01-0023, University of Southern California, May 2001.