Power Aware Routing for Sensor Databases

Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri
Dept. of Computer Science
University of California, Santa Barbara, CA 93106
{chiran,agrawal,suri}@cs.ucsb.edu

***Abstract*— Wireless sensor networks offer the potential to span and monitor large geographical areas inexpensively. Sensor network databases like TinyDB [1] are the dominant architectures to extract and manage data in such networks. Since sensors have significant power constraints (battery life), and high communication costs, design of energy efficient communication algorithms is of great importance. The data flow in a sensor database is very different from data flow in an ordinary network and poses novel challenges in designing efficient routing algorithms. In this work we explore the problem of energy efficient routing for various different types of database queries and show that in general, this problem is NP-complete. We give a constant factor approximation algorithm for one class of query, and for other queries give heuristic algorithms. We evaluate the efficiency of the proposed algorithms by simulation and demonstrate their near optimal performance for various network sizes.**

***Index Terms*— sensor networks, graph theory, mathematical programming/optimization**

I. INTRODUCTION

We consider the problem of energy efficient routing in a sensor network for data aggregation queries. In such networks, energy costs are dominated by communication and we give algorithms to optimally route aggregated data through the network.

A. Motivation

Sensor networks are large scale wireless networks composed of tiny *sensor nodes*. Each sensor node is a two part combination of an array of sensors and a tiny computer capable of wireless communication. The sensors periodically measure the physical environment around them and generate a stream of data. The computer takes this data and communicates it through a wireless network to other nodes in the network. Sensor database systems like TinyDB [1] have been proposed to provide a SQL like data interface to simplify the extraction and management of data in such a network.

In the Internet, data extraction and routing are relegated to different layers in the protocol stack. But in a sensor network, such separation of application and network layers is not desirable. Most sensor network are *not* designed to be general purpose networks and thus their architecture can be fine tuned for the intended application. Most importantly, sensor networks are severely power-limited: sensor nodes run on battery power, which is generally impossible to replenish once the network is deployed. Thus the lifetime and usefulness of the network crucially depends on the energy demands that the application places on it. A node can spend energy in sensing, communication and computation. In typical sensors, computation and sensing costs are several orders of magnitude smaller than the communication cost. As an example, for Berkeley sensor motes, transmitting one bit requires 1 μJ, while executing one instruction takes about 0.008 μJ [2] of energy. Thus the biggest power savings can be achieved by communication strategies which are optimal for the application at hand.

The authors were supported by NSF grants IIS-0121562, CNS-0423336 and Army Research Organization grant DAAD19-03-D-0004 through the Institute for Collaborative Biotechnologies.

B. Routing Challenges in Sensor Database Applications

In a sensor database system, users pose queries to a special node called the *base station* which disseminate the queries over the network. An example query could be to find the average temperature over all sensors in the network. Sensors process the query and send their replies back to the base station. The most natural way to answer queries is to gather data from all the sensors at the base station and process the query there. Since this process is very expensive in terms of the amount of data transferred, systems such as TinyDB have proposed *in-network aggregation* to process queries [3]. In-network aggregation has resulted in routing trees playing a central role in the database application. Prior research on power aware routing [4], [5], [6] and query processing has considered the two problems in isolation. In particular, because of in-network aggregation many data packets are often aggregated to a single packet. The typical energy efficient routing schemes do not take into account this collapse of data flow. Similarly database query processing techniques have critically relied on the availability of a routing tree for in-network aggregation without considering the optimality of such a tree.

In this paper we first classify database queries based on the level of aggregation they employ for processing. If a query can be processed in-network by transmitting a single value from each sensor node on the routing tree, we refer to this as a *fully-aggregated query*. The other case is when an intermediate node needs to transmit all incoming values toward the base station; this is referred to as an *unaggregated query*. The final category which we refer to as *partially aggregated query* is one in which the amount of data each node needs to transmit has an upper bound, e.g. a histogram over the data values. Each type of in-network aggregation results in a different optimal routing tree. Although transmission costs dominate wireless communication, reception costs also play a strong role in determining optimal routing tree. For example, the radio for MICA2 sensor motes drain 12mA current while transmitting and 8mA while receiving [7]. Simply keeping the radio switched on at idle also drains consumes almost the same amount of power as receiving. Thus both receive and idle time in a network can significantly affect the total energy consumption.

The problem of finding power efficient routing tree arises in other contexts as well. For example some sensor network routing schemes divide sensors into clusters with every node reporting to its cluster leader. The nodes in a cluster can transmit to the cluster leader in a single hop, or use lower power and try to reach the leader in multiple hops. Multiple hops are generally more power efficient than single hops and in that case we have a similar problem of needing to find the best routing tree to reach the cluster leader.

C. Our Contribution

We show that finding the optimal routing tree is NP-complete for aggregated, unaggregated or partially aggregated queries. For fully aggregated queries we give a routing algorithm with worst case constant factor performance guarantee. For unaggregated and partially aggregated queries we give approximation algorithms which in practice come very close to optimal performance. Finally we evaluate these algorithms experimentally in various network topologies.

The rest of the paper is organized as follows. In Section II we discuss the properties of our model and discuss some related work. Then in Sections III, IV and V, we discuss routing problems for fully aggregated, unaggregated and partially aggregated queries. Section VI is devoted to an experimental evaluation of the proposed algorithms and section VII concludes the paper.

II. Background and Related Work

A. The Model

Consider a network of N sensor nodes labeled as $i, i = 1, 2, \ldots N$ and having energy $e_i, i = 1, \ldots, N$. We assume that the node 1 acts as the base station. This node is special in that it is assumed to have unlimited power supply, i.e. $e_1 = \infty$. We assume a setting in which the environment is continually monitored by the sensor and the data value is sent to the base station. Formally, we assume that every node generates one unit of data every time unit. Transmitting one unit of data costs one unit of energy, while receiving one unit of data costs $c_r < 1$ amount of energy for each sensor.

Let us assume that the N sensors are distributed arbitrarily in a two dimensional plane and that that two sensors can communicate with each other if they are within range r. We assume that sensors transmit with fixed power; that is, they do not dynamically adjust their power for each transmission, depending on the distance to the receiver node. The radio links are also assumed to be symmetric. In that case, we can construct a communication graph where each node corresponds to a sensor and each edge links two sensors which can communicate with each other.

We assume that there exists a time synchronization model for computing aggregates. Time is divided into equal sized periods called *epochs*. Every epoch a leaf node transmits data to its parent. Once a parent receives data from all its children, it aggregates the data and transmits the aggregate to its parent in the next epoch. Thus if the depth of the routing tree is d, then it requires d epochs for the data from a leaf to reach the base station. Our task is to construct a routing tree on this graph so as to optimize system lifetime. Note that in any routing tree the nodes near the base station will be the most loaded because they have to route all the information from other nodes to the base.

Let us now consider how a query is answered using in-network aggregation within the Tiny AGgregation (TAG) [3] framework. Consider the query `SELECT AVG(temp) FROM sensors`. The base station floods this query to the network. As the query is propagated through the network the nodes organize themselves into a *routing tree* with the base station as the *root*. When a node hears a query it forwards the query to its children. When the nodes begin to reply, every node aggregates the replies from its children and forwards the aggregated reply to its parent. The data structure which encodes the reply is called a *partial state record* (PSR). For the AVG query, the PSR consists of a tuple $\langle$`sum, count`$\rangle$. For the leaf nodes, `count` is 1 and `sum` is the temperature reading from the sensor. For a parent node which receives two PSRs $\langle$`sum1,count1`$\rangle$ and $\langle$`sum2,count2`$\rangle$, the new PSR is

$$\begin{aligned}\langle\texttt{sum,count}\rangle &= \langle\texttt{sum1+sum2+ sensor\_value,}\\ &\quad\ \texttt{count1+count2+1}\rangle\end{aligned}$$

For a sensor network the power consumption is determined by the size of the PSR and the path that the PSR takes to reach the root. Here we summarize some example queries and associated PSR sizes.

- *Fully Aggregated*: `SELECT AVG(temp) FROM sensors`. The PSR size is constant irrespective of the number of sensors. `MAX/MIN` queries also fall into this class.
- *Unaggregated*: `SELECT nodeid FROM sensors WHERE (temp > 70)`. The PSR size near the root is proportional to the number of total sensors.
- *Partially Aggregated*: `SELECT HISTOGRAM(temp) FROM sensors`. The PSR grows in size as it flows toward the root, but has a maximum size (number of bins in the histogram) independent of the number of sensors.

Several metrics for power efficiency have been proposed in the literature [8], [4] such as total energy spent, energy spent per unit of data transmitted, and time for first node failure. Intuitively, an efficient routing tree will steer most of the traffic away from the nodes with low battery power and put it through the nodes which have more battery power and thus make sure that no node runs out of power prematurely. The success of the routing tree will be determined by the length of time it can avoid a node failure. Thus we define a metric called *system* *lifetime* which is the time required for first node failure to occur amongst all nodes. The problem that we address in this paper is as follows: for a given type of query with associated PSR size and a single base station, what is the optimal routing tree to maximize system lifetime?

B. Related Work

A description of the TinyDB system and its architecture can be found in the papers by Madden et. al. [3], [1]. An overview of the energy consumption issues for sensor networks can be found in the review article by Raghunathan et. al. [9].

The work by Singh et. al. [8] introduced various metrics for measuring the lifetime of wireless networks. Chang and Tassiulas [4] formulated the problem of routing multiple conserved data flows as a multi-commodity flow with the time for first node failure as the power efficiency metric. Sadagopan and Krishnamachari [5] have proposed another efficiency metric which is to maximize total amount of data extracted out of the network. This is not a suitable metric for sensor databases because the usefulness of a sensor database depends not only on the total amount of data extracted, but also on having as many of the nodes alive as possible. Zussman and Segall [6] have investigated power aware routing with first node failure as efficiency metric for disaster recovery networks.

Kalpakis et al. [10] have formulated the power aware routing problem for fully aggregated queries like AVERAGE as a linear programming problem. Krishnamachari et.al. [11] have looked at the power efficient routing tree problem for fully aggregated data with power metric as total number of transmissions required. This problem maps on to the well known Steiner tree problem which is known to be NP-complete. However, the total number of transmissions is not a good measure of useful system lifetime. Boukerche et. al. [12] have proposed an alternative way to conserve energy, which is to build a routing tree with a large number of leaves. The leaf nodes can turn themselves off to sleep until an interesting event occurs, while the non-leaf nodes are always awake for routing. The problem of determining spanning tree with maximum leaves is again known to be NP-complete. This approach is not applicable to database queries where all nodes need to be queried. Also a disproportionate burden of power is placed on backbone non-leaf nodes. Another way to increase power efficiency of a network is to reduce path length by utilizing multiple base station nodes. Bogdanov et al. [13] have demonstrated that optimal power efficient placement of base stations is NP hard.

III. Routing Tree for Fully Aggregated Queries

Let us consider the following model for a fully aggregated query. At every unit of time, every node produces one unit of data. A node which receives multiple units of data from other nodes aggregates that data with its own and forwards a single unit of aggregated data. Thus every node transmits a single unit of data, but might receive zero or more units of data depending on network topology and routing. An example of fully aggregated query is computing the average of data values over all sensors.

In this section we first discuss the problem of finding optimal routing tree for fully aggregated queries with and without reception cost. We demonstrate that the problem of finding optimal routing tree with reception cost is NP-complete. Then we give a near optimal routing tree algorithm for the problem and analyze its performance.

If we assume that there is no energy cost to receive data, then finding the optimal routing tree is a simple problem. The energy cost for any node is just the cost of transmitting one unit of data to its parent in the routing tree regardless of its energy level. Thus in any routing tree, every node consumes equal amount of energy regardless of the topology. To state it formally:

Proposition 1: In the model where the receive cost is assumed to be zero, every spanning tree is optimal for aggregated queries, even with arbitrary node energy levels.

In the introduction we have already seen that reception cost is only marginally smaller than transmission cost for sensor nodes. In fact, for fully aggregated queries, a node will receive packets from multiple neighbors, but transmit to only one parent node. Thus, for fully aggregated queries, the total reception cost can be significantly larger than the transmission cost.

Surprisingly, it turns out that once we introduce reception cost into the model, the problem of finding optimal routing tree becomes intractable. The amount of data received by a node is dependent on the number of neighbors and thus to minimize power consumption a node should have as few neighbors as possible. This optimization problem is NP-complete, as shown by the following theorem.

Theorem 1: Finding maximum lifetime routing tree for fully aggregated queries with reception costs is NP-complete.

Proof: We shall show that the problem is NP-complete even when all nodes have equal power. The total energy consumption for node i per unit time is $1 + (\text{nbr}(i) - 1)c_r$ where c_r is the cost to receive one unit of data and $\text{nbr}(i)$ is the number of neighbors of node i in the routing tree. Thus to maximize lifetime we need to find a spanning tree for the graph such that the number of neighbors for each node is the minimum, or equivalently we need to minimize the maximum degree of the spanning tree. But this problem is the same as Minimum Degree Spanning Tree (MDST) which is known to be NP-complete [14].

■

At this point we would like to point out an upper bound on the optimal routing lifetime T_{OPT}. Consider a network where the minimum energy node has energy $e_{\min}$. The energy consumption for this node is minimized when it acts as a leaf in the routing tree and does not need to receive any data. In that case the lifetime of this node itself is $e_{\min}/1 = e_{\min}$ (recall that transmission cost is 1). Thus we have the following upper bound:

$$T_{\text{OPT}} \leq e_{\min}. \tag{1}$$

A. *A Near Optimal Routing Tree for Fully Aggregated Queries*

Since finding optimal routing tree is NP-complete, in this section we present an algorithm to find a near optimal routing tree. To solve this problem, we convert the optimization problem to a decision problem. In other words, let us ask the following question: given a network topology and node energy levels, is there a routing tree which has a lifetime T? Then the optimization problem which corresponds to maximizing T can be solved by performing a binary search on T.

We note that for *unequal energy levels* for nodes, MDST is not the optimal solution to the routing tree problem. A node with large energy can support more neighbors than a node with little energy. So our solution strategy is two step. In the first step we reduce the general routing tree problem to the MDST problem. In the second step we solve the MDST problem using an approximation algorithm given by Fürer and Raghavachari[15], [16] in the context of general spanning tree algorithms.

Transformation of Routing Problem to the MDST Problem: Let us consider a network with nodes having unequal energies e_i. If the cost of receiving one unit of data is c_r ($0 < c_r < 1$), then to achieve a lifetime of T, each node i can have at most B_i neighbors such that the following power constraint is satisfied:

$$\frac{e_i}{1 + c_r(B_i - 1)} \geq T, \tag{2}$$

which means that the maximum number of neighbors that a node i can have is

$$B_i = \left\lfloor 1 + \frac{1}{c_r}\left(\frac{e_i}{T} - 1\right) \right\rfloor. \tag{3}$$

For the root node there is no power constraint and hence we set $B_{\text{root}} = N$. Now, for every node i in the original graph, introduce $N - B_i$ auxiliary nodes and connect them to node i. We call this graph with auxiliary nodes the *augmented graph*. We claim that if we construct the MDST on this graph and from the resulting MDST delete the auxiliary nodes, the resulting spanning tree will have degree at most B_i for every node i.

To prove this claim, consider the original and the augmented graphs and their respective spanning trees. Let us call the optimal spanning tree with non-uniform degree bounds B_i to be the Non-uniform Minimum Degree Spanning Tree (NMDST). Suppose there exists an NMDST on the original graph which satisfies the condition $\deg(i) \leq B_i$ for all i where $\deg(i)$ is the degree of node i in the NMDST. The MDST on the augmented graph is simply the NMDST with the auxiliary nodes attached to it. Then every node i on the MDST satisfies the *uniform* degree bound $\deg(i) \leq N$ by construction (we have introduced $N - B_i$ auxiliary nodes for each node i). Conversely, we see that if there exists an MDST with degree bound N, then there exists an NMDST which satisfies the degree bound $\deg(i) \leq B_i$.

We illustrate graph augmentation with the example shown in Fig. 1. We have a 4 node graph whose nodes have energy levels 1 and 3/2. The actual nodes and edges in the graph are shown in solid lines, while auxiliary nodes and auxiliary edges are shown in dashed lines. With $c_r = 1/2$ and $T = 1$ we find that we must introduce 3 auxiliary nodes for the nodes with energy 1 and 2 auxiliary nodes for the node with energy 3/2. Finding the MDST on this graph and deleting the auxiliary nodes immediately leads to the correct routing tree with lifetime 1.

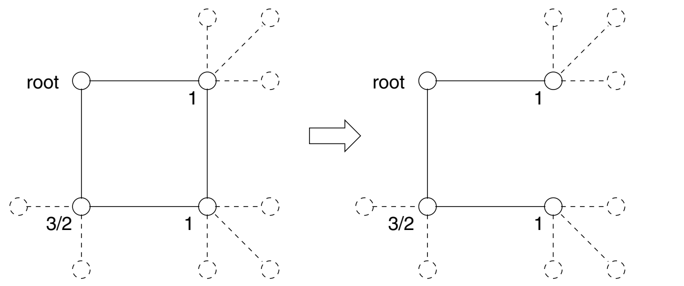

Fig. 1. Finding NON-UNIFORM MINIMUM DEGREE SPANNING TREE with node degree bounds. On the left is the augmented graph, while on the right we have the resulting NMDST. The auxiliary nodes are shown in dashed line.

Solving the MDST Problem: Now we can turn our attention to the MDST problem itself. The MDST problem is known to be approximable to within OPT+1 [15], [16]. In other words, if the maximum degree of minimum degree spanning tree is OPT, then there is a polynomial time algorithm which can produce a spanning tree with maximum node degree OPT+1. It is easy to convince oneself that if the MDST problem can be solved within OPT+1, then the NMDST problem also can be solved such that $\deg(i) \leq B_i + 1$. A detailed discussion of the OPT+1 approximation algorithm for finding the MDST can be found in the paper by Fürer and Raghavachari [15], [16]. Here we describe the algorithm without justifying it.

The full algorithm to find the routing tree is described here as the algorithm AGGREGATED-TREE (algorithm 1). The algorithm takes as input a graph $G(V, E)$ with vertices V and edges E and outputs a tree. The algorithm runs in $\mathcal{O}(NE\alpha(E, N)\log N)$ time [16] where α is the inverse Ackermann function, E is the number of edges.

Let us denote the lifetime achieved by the optimal tree as T_{OPT} and lifetime achieved by the approximation as T_{MDST}. Then using eqn. 2

$$\frac{T_{\text{OPT}}}{T_{\text{MDST}}} = \frac{1 + c_r B_i}{1 + c_r(B_i - 1)} = 1 + \frac{c_r}{e_i} T_{\text{OPT}}$$

Since the minimum value of e_i is e_{min}, and $T_{\text{OPT}} \leq e_{\text{min}}$ (eqn. 1),

$$\frac{T_{\text{OPT}}}{T_{\text{MDST}}} \leq 1 + \frac{c_r}{e_{\text{min}}} T_{\text{OPT}} \leq 1 + c_r < 2$$

Thus our approximation scheme is within a constant factor $(1 + c_r)$ of the optimal routing tree. As expected from proposition 1, for zero receive cost, the approximate lifetime coincides with the optimal lifetime.

IV. ROUTING TREE FOR UNAGGREGATED QUERIES

In unaggregated queries the volume of data is conserved. All data that is generated in the nodes must be delivered to the

Algorithm 1 AGGREGATED-TREE$(G(V,E), e)$

```
1: For each vertex $v$, augment it with $N - B_v$ vertices according to eqn. 3
2: Find a spanning tree $\mathcal{T}$ of $G$. Let $k$ be its maximum degree
3: Mark all vertices of degree $k$ and $k-1$ as bad. Mark all other vertices as good.
4: Remove all bad vertices of degree $k$ and $k-1$ generating a forest $F$ from $\mathcal{T}$.
5: while there is an edge $(u,v)$ connecting two different components in $F$ and all vertices of degree $k$ are marked bad do
6:     Find the cycle $C$ generated by $\mathcal{T}$ and the edge $(u,v)$
7:     Mark all bad vertices in $C$ as good
8:     Update $F$ by combining the components and vertices along cycle $C$.
9: end while{At this point we have identified a vertex of degree $k$ which is in a cycle $C'$}
10: if there is a vertex $w$ of degree $k$ marked good then
11:     Delete an edge incident on $w$ to break the cycle $C'$ and reduce the degree of $w$.
12:     Update $\mathcal{T}$ and if necessary update $k$.
13:     Goto 3.
14: end if
15: Delete all auxiliary nodes from $\mathcal{T}$ and output $\mathcal{T}$.
```

root. An example of such a query is `SELECT nodeid FROM sensors WHERE (temp > 70)`. If we assume that the condition `(temp > 70)` is satisfied with probability p_t by the sensors, then on the average, every sensor node produces p_t units of data every time period. Thus the problem is to route a total of Np_t data every time unit from the nodes to the root with every node generating p_t data and no data getting lost. With a rescaling of energy or time unit, we can convert it to the problem where every node generates one unit of data.

Unlike the fully aggregated routing tree problem, finding optimal routing tree for this problem is NP-complete *even when we do not take into account the cost to receive data* (Theorem 2). So for the sake of simplicity, in this section we shall take account of only transmission costs. But as the following argument shows, ignoring reception cost does not invalidate our conclusions for realistic networks. In the optimal routing tree, consider the node which is most likely to run out of power first. The data inflow and outflow for this node are f_{in} and $f_{\text{out}} = f_{\text{in}} + 1$ respectively. Total energy consumption per unit time for this node is

$$f_{\text{out}} + c_r(f_{\text{out}} - 1) = f_{\text{out}}(1 + c_r) - c_r$$

Since this is the highest loaded node, $f_{\text{out}} \gg 1 > c_r$, and we can say that total power consumption is proportional to the transmission power consumption. Thus by rescaling the energy of the nodes by a factor of $\frac{1}{1+c_r}$, we can take into account the most important effects of reception cost. The plan for the rest of the section is as follows: we first show that the problem of finding optimal routing tree without reception cost is NP-complete. Then we formulate the routing tree problem as an integer program and derive an upper bound on the maximum lifetime. Next we give two approximation algorithms to solve this problem and discuss their properties.

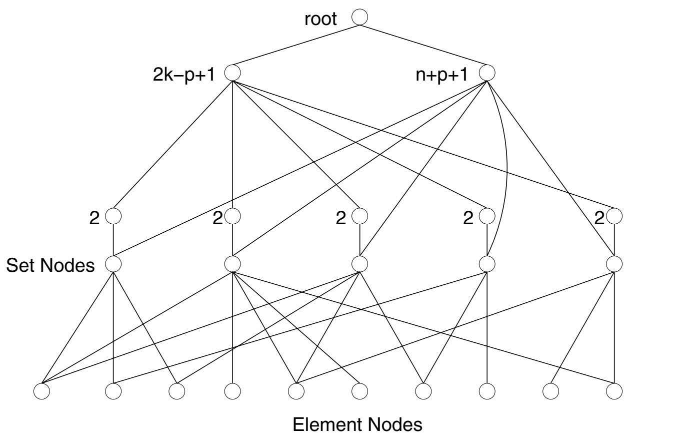

Fig. 2. Reduction from SET-COVER to the unaggregated routing tree problem. The numbers next to a node indicates the energy level of the node. The set nodes have energy equal to the size of the set + 1. The element nodes have energy 1.

A. Hardness of the Optimal Routing Tree Problem

To establish the hardness of the optimal routing tree problem, we recast it as a decision problem: given a connectivity graph and node energies, does there exist a routing tree with a given lifetime T?

Theorem 2: Finding maximum lifetime routing tree for unaggregated queries is NP-complete.

Proof: The decision problem for this instance is clearly in NP. Given a graph and the routing tree on it, it is easy to verify whether the routing tree achieves the lifetime T. To show NP-completeness, we shall exhibit a reduction from SET-COVER to the current problem. The decision problem for SET-COVER is as follows: we are given n elements numbered from $1 \ldots n$. We are also given k subsets of these elements $S_1, S_2, \ldots, S_k$. The problem is to decide if there exists a selection of p subsets from the collection of subsets such that the union of p subsets cover all the n elements.

Given this instance of SET-COVER, we construct the following instance of the communication graph. The nodes in the graph are arranged in five rows as shown in Fig. 2. The first row consists of the root node. The second row consists of two nodes, one with energy $2k-p+1$ and the other with $p+n+1$. Both these nodes are connected to the root node. These two nodes are used to decide which sets will be in the set cover and which will not. The third row consists of k nodes each with energy 2. All the nodes in the third row are connected to the node with power $2k-p+1$. The fourth row consists of k nodes which corresponds to the k subsets. The $i-$th node in the row has energy $|S_i|+1$. Each of these nodes are connected to the corresponding node on the third row. Also each of these nodes is connected to the node on second row with energy $p+n+1$. The fifth and last row corresponds to the n elements and each node has unit energy. Each node corresponding to S_i

is connected to the elements that it contains. For example if S_2 contains the elements $(2, 5, 6, 10)$, then those nodes have links to the node corresponding to S_2.

Now we claim that there exists a routing tree with lifetime 1 if and only if a set cover of size p exists. Suppose there exists a set cover of size p. Then we can construct the routing tree as follows. The data from the p sets and the n elements can be routed through the node with power $n + p + 1$ to the root. The data from rest of the $k - p$ subsets and the k nodes in the third row can be routed through the node with power $2k - p + 1$. Conversely if there is a routing tree of lifetime 1, then the third row nodes with energy 2 ensure that they are carrying data from only $k - p$ subsets. Thus the rest of the p nodes constitute a set cover. ■

Although we have used nodes with different powers in this proof, it can be shown [17] that this problem remains NP-complete even if we constrain all nodes to have the same energy. The proof, which we omit, relies on a pseudo-polynomial reduction [14] from SCHEDULING. The optimal unaggregated routing tree problem for equal energy nodes is similar to an NP-complete problem that has been studied in the network design literature by the name of CAPACITATED SPANNING TREE [14]. In CAPACITATED SPANNING TREE problem we are given a graph G with edge weights, a bound K and a root node. The problem is to find a minimum weight spanning tree such that every subtree that is hanging from the root node contains fewer than K vertices.

Now that we have seen that the optimal routing tree problem is NP-complete we would like to investigate the possibility of approximation algorithms for the problem. To evaluate the effectiveness of these approximation algorithms, it is useful to have an upper bound on the optimal solution. In the following section, we formulate the routing problem as a linear program which leads us to an upper bound.

B. An Integer Programming Formulation

In this section, we formulate the decision problem as an integer linear program. To do this, we define two sets of *integer* variables x_{ij} and f_{ij} for every edge ij on the graph. If j is the parent of i in the routing tree, then $x_{ij} = 1$, else $x_{ij} = 0$. f_{ij} is the data flow that i sends to j along the edge ij per unit time. We are interested in knowing whether there exists a routing tree with lifetime T. In that case, feasibility of the following integer linear program is equivalent to the decision problem.

$$\sum_{j=1}^{N} x_{ij} = 1, \quad i = 2, \dots N \tag{4}$$

$$\sum_{j=1}^{N} f_{ij} - \sum_{j=2}^{N} f_{ji} = 1, \quad i = 2, \dots N \tag{5}$$

$$x_{ij} \le f_{ij} \le \frac{e_i}{T} x_{ij} \tag{6}$$

$$x_{ij} \ge 0 \tag{7}$$

$$f_{ij} \ge 0 \tag{8}$$

The condition (4) ensures that every node has exactly one parent, i.e. the set of edges for which $x_{ij} = 1$ define a tree. Condition (5) is a flow conservation condition. The outflow from a node is exactly one unit larger than inflow. Condition (6) enforces two things. First, if j is i's parent, then j must send at least one unit of flow to its parent. Also if j is not i's parent, then flow along the edge ij is zero. Second, it makes sure that the outflow from node i respects the energy constraint of node i. Conditions (7) and (8) are usual positivity conditions.

Of course, finding a feasible solution to the integer program is still NP-complete, but it gives us a way to find an upper bound for the problem by relaxing the integrality conditions. Relaxing the variable x_{ij} to be any real number between 0 and 1 means that we allow data to be forwarded to more than one node. Thus the relaxed linear program produces a solution which is no longer a tree. In fact, we shall show that the relaxed problem is equivalent to a maximum flow problem. With relaxation of integrality conditions, f_{ij} can also be arbitrary real numbers. At this point the relaxed linear program can be written down without the x_{ij} variables at all. The new non-integer linear program is

$$\sum_{j=1}^{N} f_{ij} - \sum_{j=2}^{N} f_{ji} = 1, \quad i = 2, \dots N \tag{9}$$

$$\sum_{j=1}^{N} f_{ij} \le \frac{e_i}{T} \tag{10}$$

$$f_{ij} \ge 0 \tag{11}$$

If we define a new variable $y_{ij} \equiv T f_{ij}$, then the decision problem can be written as a maximization problem:

Maximize T, subject to

$$\sum_{j=1}^{N} y_{ij} - \sum_{j=2}^{N} y_{ji} = T, \quad i = 2, \dots N \tag{12}$$

$$\sum_{j=1}^{N} y_{ij} \le e_i \tag{13}$$

$$y_{ij} \ge 0 \tag{14}$$

This linear program is equivalent to a maximum flow problem where each node (except the root) has a node capacity e_i and acts as a source which sends a flow T to the sink which is the root. Using conventional maximum flow algorithms[18], this problem can be solved in polynomial time. Let us call the maximum lifetime solution to this flow problem T_{LP}. If the optimum lifetime achievable for the original problem with tree routing is T_{OPT}, then $T_{\text{LP}} \ge T_{\text{OPT}}$. From now on, we shall use T_{LP} as an upper bound on T_{OPT}.

For an example graph where $T_{\text{LP}} > T_{\text{OPT}}$, consider the topology shown in Fig. 3. We assume that all nodes have energy 1. In that case, the figure at right shows an optimal routing tree with lifetime $1/2$. The numbers next to the edges denote the amount of data flow along that edge. On the other hand in a maximum flow solution, we can make the bottom

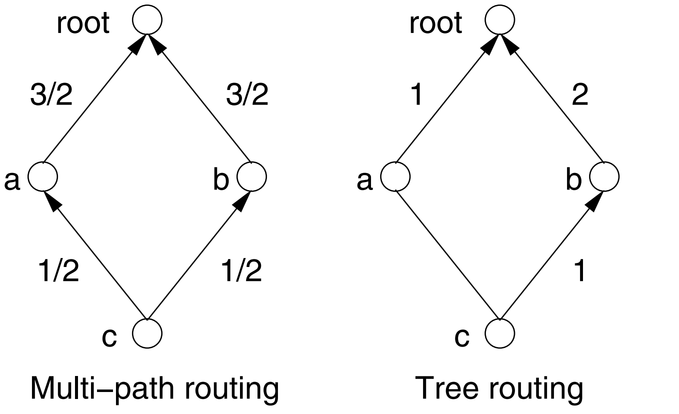

Fig. 3. Multipath routing and tree routing. The numbers next to the edges show the flow through that edge.

node route its data with probability $1/2$ along one link and with probability $1/2$ along the other link. In this case the data flow is more equally distributed and the system lifetime is $2/3$. We call this type of routing where outflow from a node can reach the root via multiple paths as *multipath* routing.

Although sensor databases use tree based routing, it is not essential to their functioning; the only requirement for correct functioning is that there should be no duplication or loss of data packets. Although multipath routing seems to yield significant energy savings, there are some hidden costs to it. For example if data is transmitted probabilistically to multiple nodes, all the receiving nodes need to be listening in even if the packet is not destined for them. As we have mentioned, even idle listening consumes significant amount of energy. Thus the benefits of randomized multipath routing depends on the complex interaction between MAC and network protocols. In this work we restrict ourselves to tree routing only.

C. Energy Conserving Routing Tree (ECRT)

Now we give a heuristic algorithm for computing an approximate routing tree whose basic idea is very similar to Prim's algorithm for the minimum spanning tree. We initialize the routing tree to be the root at first and grow the tree by adding edges one by one until the tree spans all the vertices. At every step we can determine the lifetime of the tree constructed so far by calculating the amount of data each node will be required to forward. To decide which edge to add, we add the edge that will maximize the lifetime of the resulting new tree. If there are multiple edges which result in the same lifetime, we choose the edge which will add the node with maximum energy. We call the lifetime of the tree achieved by this algorithm T_{ECRT}. The algorithm is formally described as ECRT in algorithm 2.

The algorithm ECRT has N stages and in each stage a new edge is added. To find the optimal edge to add, we need to check $\mathcal{O}(E)$ edges and for each edge, we need to check the lifetime of the resulting tree. If the diameter of the graph is d, then to determine the life of the new tree takes no more than $\mathcal{O}(d)$ time. Thus ECRT runs in $\mathcal{O}(NEd)$ time. This algorithm is not a constant factor approximation algorithm and we state

Algorithm 2 ECRT($G(V,E), e$)

```
1: Initialize tree $\mathcal{T}$ to contain the single node root.
2: while Not all nodes are in $\mathcal{T}$ do
3:     Find lifetime $T$ of the tree $\mathcal{T}$.
4:     Find the set of nodes $\mathcal{N}$ adjacent to $\mathcal{T}$.
5:     Add node $v \in \mathcal{N}$ which reduces $T$ by the least amount
6: end while
7: Output $\mathcal{T}$.
```

it formally as the following lemma. In the interest of brevity, the proof [17] is omitted.

Lemma 1: In the worst case the following lower bound exists for the performance ratio of the ECRT algorithm

$$\frac{T_{\text{OPT}}}{T_{\text{ECRT}}} = \Omega(\log N)$$

In other words, there exists instances of connectivity graphs where the optimal lifetime is better than the lifetime achieved by ECRT by a factor of $\log N$ or larger.

Now that we have seen that ECRT is not an optimal algorithm for the routing problem, we ask the question if there is some optimization that can improve is performance. With this aim, we present an algorithm which optimizes a pre-existing routing tree.

D. Local Optimization

Consider an optimal routing tree. If any single node in the tree switches its parent to another node, the resulting tree will have a lifetime less than or equal to the optimal tree. Thus for the optimal solution, the choice of parent for every node is optimal. We now define a routing tree to be *locally optimal*, if the lifetime of the tree can not be improved by switching the parent of *any single* node. Note that the optimal tree is always locally optimal, but the converse need not be true.

This criterion immediately suggests an approximation algorithm for finding a locally optimal tree which we call LOCAL-OPT. The algorithm accepts as input an arbitrary routing tree and considers the nodes sequentially in some particular order. We try to see if the lifetime of the tree will be improved by switching the parent of the current node in consideration. If such a switch is favorable, it is made; otherwise we proceed to the next node. We call this switch an *improvement step*. We denote the lifetime achieved by this algorithms as T_{LO} and describe it formally as algorithm 3.

This algorithm is not a constant factor approximation algorithm for the unaggregated routing tree problem as shown by the following lemma. In the interest of brevity, the proof [17] is omitted.

Lemma 2: In the worst case the following lower bound exists for the performance ratio of the LOCAL-OPT algorithm

$$\frac{T_{\text{OPT}}}{T_{\text{LO}}} = \Omega\left(\frac{\log N}{\log\log N}\right)$$

In simpler words, there exists an instance of a graph where the optimal lifetime is better than the lifetime produced by LOCAL-OPT by a factor of $\log N/\log\log N$ or more. The

Algorithm 3 LOCAL-OPT($G(V,E), \mathcal{T}, e$)

```
 1: done ← FALSE
 2: while done = FALSE do
 3:   done ← TRUE
 4:   for all $v \in V$ do
 5:     if switching the parent of $v$ improves $\mathcal{T}$ then
 6:       switch parent of $v$ to improve $\mathcal{T}$
 7:       done ← FALSE
 8:     end if
 9:   end for
10: end while
11: Output $\mathcal{T}$.
```

LOCAL-OPT algorithm can be used as a stand alone algorithm to find an approximate routing tree, or it can be used as an additional optimization on the tree produced by ECRT algorithm.

For performance analysis of LOCAL-OPT, consider a network with maximum and minimum energies e_{max} and e_{min} respectively. Then the maximum lifetime for *any* routing tree is $\frac{e_{\text{max}}}{\text{deg(root)}N}$ where deg(root) is the degree of the root node. Let us now look at the size of a local improvement step: the smallest size of the improvement step is

$$\frac{e_{\text{min}}}{N-1} - \frac{e_{\text{min}}}{N} \approx \frac{e_{\text{min}}}{N^2}.$$

Then the total number of improvement steps are bounded by

$$\frac{e_{\text{max}}}{e_{\text{min}}} \frac{N}{\text{deg(root)}}$$

Thus LOCAL-OPT runs in time polynomial in $e_{\text{max}}/e_{\text{min}}$ and N.

V. Routing Tree for Partially Aggregated Queries

A partially aggregated query has a PSR size which lies between fully aggregated queries and unaggregated queries. Consider as an example a histogram query with ℓ bins. The PSR can be just a listing of bins with their associated frequencies. When the query reply starts out at a leaf node the PSR consists of 1 bin which holds the value and the corresponding frequency which is 1. As more and more data points are added, they begin to fall into distinct bins and the size of the PSR increases. But once the PSR size reaches ℓ bins, any new data added falls into one of the older bins and hence the PSR size reaches a constant. So we model a partially aggregated query as follows. Every node produces one single unit of data every time period. As long as the total data inflow into a node is less than ℓ, the data remains conserved, i.e. if two PSRs reach a node, then the new PSR size is just the sum of the two old PSR sizes and 1 (size of its own data). If at any node the sum of the PSR sizes exceed ℓ, the PSR forwarded to the parent has size ℓ only. For $\ell = N$, this problem reduces to the problem of unaggregated queries.

The approximation algorithms used to solve the unaggregated routing tree problem can be immediately adapted to solve this problem. These algorithm depend on being able to find the lifetime T of a given tree $\mathcal{T}$. For the unaggregated case, we defined the lifetime T as the minimum of the ratio $e_i/f_{\text{out}}(i)$ where $f_{\text{out}}(i)$ is the outflow from node i. For partially aggregated queries f_{out} is bound by ℓ. With this modification, both the ECRT and LOCAL-OPT algorithms can be easily applied to this problem.

VI. Experimental Results

We evaluated our algorithms on simulated communication graph topologies. The simulation parameters are described below.

- *Node Distribution*: To generate the graph we randomly place N nodes in an area of size $d \times d$. The node numbered 1 is arbitrarily chosen as the base station.
- *Connectivity*: We set the radio range to be l_r. Since our length units are arbitrary, we define a scaled radio range r as follows:

$$r \equiv \frac{l_r}{l_0}, \;\; l_0 \equiv \left(\frac{d^2}{N}\right)^{1/2}.$$

  One can think of l_0 is the average separation between nodes. Thus for a given placement of nodes, the communication graph consists of nodes as vertices and edges between nodes which are within range of each other. We expect that for radio range $l_r \lessapprox l_0$, the graph will be disconnected. In practice the approximate connectivity threshold turns out to be $r = l_r/l_0 \lessapprox 1.5$. A representative connectivity graph with $r = 1.5$ is shown in Fig. 4.
- *Energy Distribution*: The energy levels for nodes are chosen randomly from a uniform distribution between e_{max}, the maximum and e_{min}, the minimum. The range of possible values of energy are controlled by tuning the *energy ratio* α defined by

$$\alpha = \frac{e_{\text{max}}}{e_{\text{min}}}.$$

  We chose e_{max} and e_{min} such that the average is 1000. For example when $\alpha = 1$, all nodes are assigned energy 1000, while for $\alpha = 2$, nodes are assigned random energy values between 667 and 1333.
- *Energy Cost*: we assume that transmitting one unit of data costs 1 unit of energy, while receiving the same amount of data costs 0.5 units, i.e. $c_r = 0.5$.

Currently the sensor databases use shortest path routing with path length equal to number of hops. We call this strategy MIN-HOP and compare our algorithms ECRT, and LOCAL-OPT to it. Note that the routing tree produced by the MIN-HOP algorithm is same as the breadth-first-search (BFS) tree for the graph.

A. System Lifetime for Fully Aggregated Queries

In Fig. 5 we plot the performance of the AGGREGATED-TREE algorithm vs the MIN-HOP algorithm. We used two different energy distributions : $\alpha = 1$, i.e. $e_{\text{max}} = e_{\text{min}} = 1000$ and $\alpha = 4$ with $e_{\text{max}} = 1600, e_{\text{min}} = 400$.

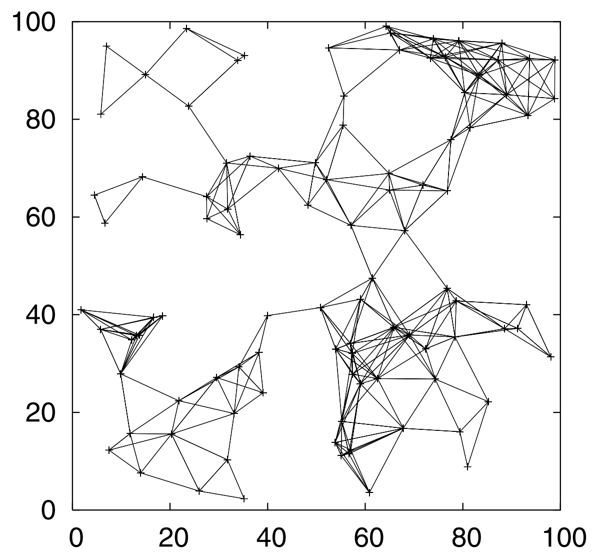

Fig. 4. An example connectivity graph for 100 nodes randomly placed in a 100×100 area with radio range set to 15 ($r = 1.5$).

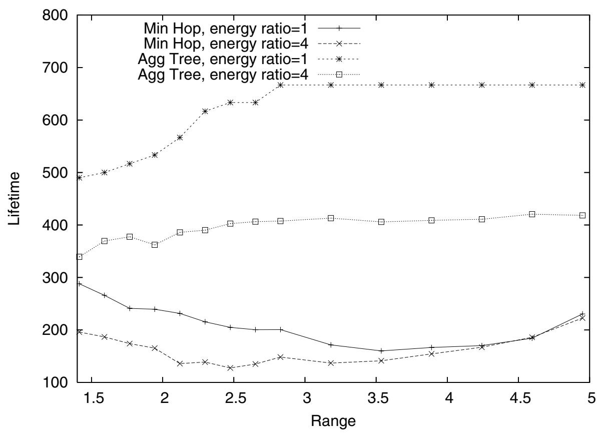

Fig. 5. System lifetime plotted against scaled range (r). There are $N = 50$ nodes.

We note that AGGREGATED-TREE significantly outperforms the MIN-HOP for all values of r. But apart from that there are several interesting features to be seen in Fig. 5. As the range r is raised from 1.5 to 3, the performance of MIN-HOP declines while the performance of AGGREGATED-TREE improves. With increasing range, the graph connectivity increases and MIN-HOP tree becomes bushier. Thus every node acquires more neighbors and MIN-HOP tree lifetime decreases. On the other hand, with more connectivity, AGGREGATED-TREE is able to reduce the maximum degree and thus perform better and better. After $r > 3.5$, the MIN-HOP lifetime begins to rise again because now most of the nodes are being able to connect to the root directly. In contrast, the AGGREGATED-TREE lifetime saturates. The optimal lifetime tree for energy ratio $\alpha = 1$ is limited by the maximum degree which can not be smaller than 2. For degree 2, the lifetime is $1000/(1 + 0.5) = 667$ and the graph shows that for $r \geq 3.0$, this lifetime is achieved. For energy ratio $\alpha = 4$, the lifetime is limited by the minimum energy node which has energy $e_{\min} = 400$. A lifetime of 400 can be achieved if the minimum energy node is a leaf and for $r \geq 3.0$, this is achieved (eqn. 1).

B. System Lifetime for Unaggregated Queries

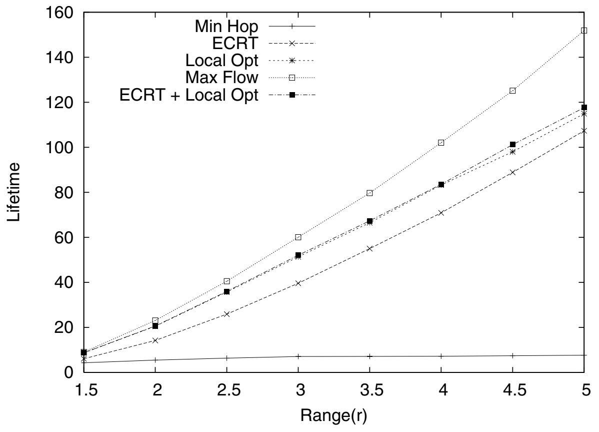

Fig. 6. System lifetime plotted against scaled range (r). There are $N = 400$ nodes with all equal energy ($\alpha = 1$) of 1000 units.

In Fig. 6 we plot the performance of ECRT, LOCAL-OPT, and MIN-HOP algorithms as a function of radio range. As an upper limit, we also plot T_{LP} which is equivalent to a maximum flow on the graph. Naturally the nodes closest to the root node are the most heavily loaded. As we increase the range r, the graph begins to become more and more connected and the number of nodes next to the root begins to rise; hence the lifetime increases. As expected, MIN-HOP performs the worst because it is not sensitive to energy. The LOCAL-OPT algorithm was initialized with a MIN-HOP tree and it performs better than ECRT. We also plotted the effect of optimizing the solution of the ECRT algorithm by using LOCAL-OPT on it. The optimized solution is only marginally better than LOCAL-OPT by itself.

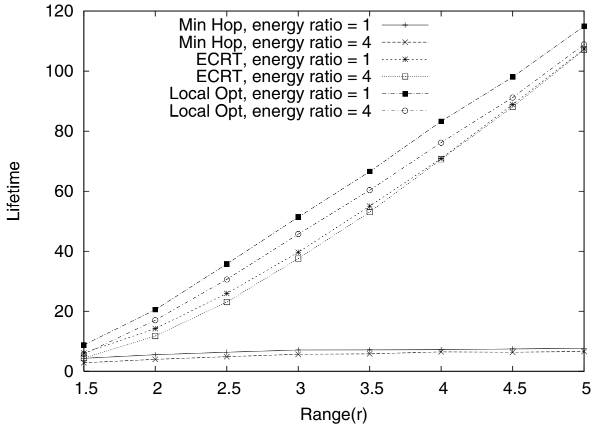

Fig. 7. Effect of changing the distribution of energy values on system lifetime ($N = 400$).

The effect of changing the distribution of energy values is shown in Fig. 7. We varied the energy ratio $\alpha = e_{\max}/e_{\min}$ from 1 to 4. The effect on the average system lifetime was minimal.

C. Effect of Network Size on Lifetime for Unaggregated Queries

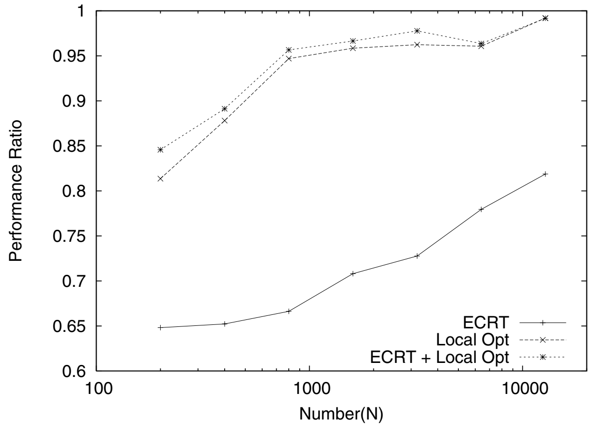

Fig. 8. Performance ratio achieved by ECRT and LOCAL-OPT routing algorithms ($r = 3.0$).

Consider a network with all nodes having equal energy. For any tree algorithm, the best performance is achieved when the sizes of the subtrees that are rooted at the immediate neighbors of the root node are equal. The chance of doing this increases as the number of nodes and hence number of paths in the network increases. So we expect that with increasing N, our approximation algorithms will perform close to optimal. Quantitatively, we define a quantity called *performance ratio* as follows:

$$\text{Performance ratio} = \frac{T_{\text{APPROX}}}{T_{\text{LP}}}.$$

Closer this ratio is to 1, better the algorithm. In Fig. 8, we plot the performance ratio of the ECRT and LOCAL-OPT algorithms as a function of N. The graph clearly demonstrates that both algorithms approach the optimal solution as N increases, but LOCAL-OPT is much quicker in convergence. Optimization of the ECRT solution by the LOCAL-OPT algorithm yields marginally better results.

D. System Lifetime for Partially Aggregated Queries

Recall that for partially aggregated queries, the message size (or the PSR size) increases from leaves towards the root, but it can not exceed a limit which we call ℓ. We plot the system lifetime as a function of ℓ in Fig. 9. For large $\ell (\ell \geq N)$, this problem becomes identical to the unaggregated query problem. The upper graph shows the results for equal energy case ($\alpha = 1$), while the lower graph shows the results for unequal energies ($\alpha = 4$).

As expected, we see that LOCAL-OPT and ECRT algorithms perform significantly better than the MIN-HOP algorithm for most values of ℓ. For small values of ℓ, being

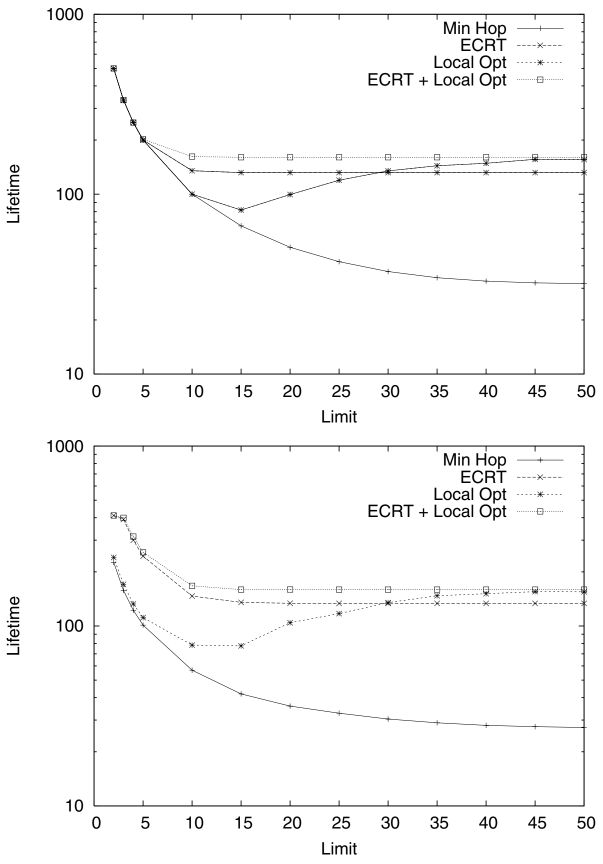

Fig. 9. System lifetime plotted as function of data flow limit. The top graph shows lifetime for equal energy levels ($\alpha = 1$), while the bottom graph shows lifetime for unequal energy levels ($\alpha = 4$). The total number of nodes N is 100 and range is $r = 3.0$.

able to distribute load uniformly across sensors is no longer crucial and thus all the algorithms perform similarly. Although for large ℓ, LOCAL-OPT outshines ECRT, for intermediate values of ℓ, ECRT does better. We conjecture that for small values of ℓ, small local changes in the routing tree do not lead to improvements in lifetime and thus LOCAL-OPT stops improvement steps too early. The best results are achieved when we optimize the output of the ECRT algorithm by using it as the input to LOCAL-OPT.

VII. Conclusion

We have argued that the problem of energy efficient routing in sensor databases is intimately connected with the type of query that the database is expected to answer. For many types of queries, we have shown that the problem of efficient tree routing is NP-complete. For fully aggregated queries, we have given a constant factor approximation algorithm; for other problems we have given heuristic algorithms which exhibit excellent performance in practice and for large networks approach the optimal solution.

References

[1] S. Madden, S. Szewczyk, M.J. Franklin, and D. Culler, "Supporting aggregate queries over ad-hoc sensor networks," in *Workshop on Mobile Computing and Systems Application*, 2002.

[2] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister, "System archtecture directions for networked sensors," *ACM SIGPLAN Notices*, vol. 35, pp. 93–104, 2000.

[3] S. Madden, M.J. Franklin, J. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," in *Proc. of OSDI '02*, 2002.

[4] Jae-Hwan Chang and Leandros Tassiulas, "Energy conserving routing in wireless ad-hoc networks," in *Proc. of INFOCOM*, 2000.

[5] N. Sadagopan and B. Krishnamachari, "Maximizing data extraction in energy-limited sensor networks," in *Proc. of INFOCOM*, 2004.

[6] Gil Zussman and Adrian Segall, "Energy efficient routing in ad hoc disaster recovery networks," in *Proc. of INFOCOM*, 2003.

[7] *MPR-Mote Processor Radio Board User's Manual*, Crossbow Technology, `www.xbow.com`, 2003.

[8] Suresh Singh, Mike Woo, and C. S. Raghavendra, "Power-aware routing in mobile ad hoc networks," in *Mobile Computing and Networking*, 1998, pp. 181–190.

[9] Vijay Raghunathan, Curt Schurgers, Sung Park, and Mani B. Srivastava, "Energy-aware wireless microsensor networks," *IEEE Signal Processing Magazine*, p. 40, March 2002.

[10] K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Maximum lifetime data gathering and aggregation in wireless sensor networks," in *IEEE Networks'02 Conference*, 2002.

[11] Bhaskar Krishnamachari, Deborah Estrin, and Stephen B. Wicker, "The impact of data aggregation in wireless sensor networks," in *Proc of the 22nd International Conference on Distributed Computing Systems*, 2002, pp. 575–578.

[12] Azzedine Boukerche, Xiuzhen Cheng, and Joseph Linus, "Energy-aware data-centric routing in microsensor networks," in *Proc. of the 6th international workshop on Modeling analysis and simulation of wireless and mobile systems*, 2003, pp. 42–49.

[13] Andrej Bogdanov, Elitza Maneva, and Samantha Riesenfeld, "Power-aware base station positioning for sensor networks," in *Proc. of INFOCOM*, 2004.

[14] Michael R. Garey and David S. Johnson, *Computers and Intractability : A Guide to the Theory of NP-Completeness*, W.H. Freeman, 1979.

[15] M Furer and B Raghavachari, "Approximating the minimum degree spanning tree to within one from the optimal degree," in *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1992, pp. 317–324.

[16] B. Raghavachari, "Algorithms for finding low degree structures," in *Approximation Algorithms for NP-hard Problems*, D. Hochbaum, Ed. 1997, PWS Publishing Company.

[17] Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri, "Power aware routing for sensor databases," *UCSB CS dept. Tech Report*, 2004.

[18] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows*, Prentice Hall, 1997.