A Dynamic Query-tree Energy Balancing Protocol for Sensor Networks

H. Yang, F. Ye and B. Sikdar
Department of Electrical, Computer and Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY 12180
Email: {yangh2, yef, sikdab}@rpi.edu

Abstract— **Static broadcast tree protocols have been proposed in literature to optimize the querying procedure in sensor networks. In this paper we address the issue of how to mitigate the unevenness of energy distribution and its undesirable effects like reduced network lifetime and loss of connectivity in a sensor network that are caused by static broadcast trees. We propose a "Dynamic Query-tree Energy Balancing" (DQEB) protocol to dynamically adjust the tree structure and minimize the overall broadcast cost. The proposed algorithm scales well, is distributed and does not need any global information. Locally, the broadcast power consumption is minimized while globally, the broadcast load and power distribution are balanced across the whole sensor network. Our simulation results verify that the DQEB protocol achieves significantly better balance in the battery power distribution and extends the network's lifetime considerably.**

I. Introduction

During the past several years advances in the design and development of low-cost, low-power-consumption sensors have led to active research about large scale, self-configurable wireless sensor networks and its applications. A large fraction of these applications, such as monitoring and measuring, rely on "*querying*" to collect information across the whole sensor network, wherein a set of sensors is specifically asked to report information of interest. Due to sensor network's unique characteristics like limited power, relatively lower processing ability of nodes and large scale, querying is usually carried out by broadcasting or multicasting. However, the constraint of limited energy in sensor networks makes broadcast/multicast through flooding for issuing queries impractical because of its large amount of unnecessary query re-transmissions and the consequent energy wastage. This paper focuses on developing techniques to improve the querying procedure to minimize energy consumption and maximize the sensor network's lifetime.

The problem of optimizing broadcasting or multicasting in all-wireless networks, or specifically ad-hoc networks, has received significant attention over the last few years [13], [8], [9], [12]. Notable among these are "broadcast tree protocols" for optimizing the query procedure in sensor network which have been proposed to mitigate the energy consumption in sensor networks. In [15] it has been shown that the minimum-energy broadcast tree problem is NP-complete and the authors have proposed an approximate algorithm to provide a bounded performance guarantee for the problem in a general setting. The authors of [5], [11], [14] propose heuristic solutions to address the problem of constructing broadcast and multicast trees. A fault-tolerant, distributed, energy-efficient protocol for the construction of broadcast trees is proposed in [4]. Protocols to support querying in sensor networks along with an analysis for their complexity are presented in [3], [12]. However, these protocols are useful only for the development of static trees where the capabilities and constraints of each node is assumed to stay the same for the lifetime of the network. Furthermore, these are un-weighted protocols whose assumptions are valid only when the nodes are first deployed. As the broadcast tree is used, the remaining energy of the non-leaf nodes of the tree become substantially less than those of leaf nodes due to the fact the non-leaf nodes have to re-transmit broadcast queries, while leaf nodes do not. The fact that the nodes' remaining energy is not always uniform and the existence of unequal energy depletion rates is of critical importance for the performance of the sensor network. A static query tree will cause the whole sensor network's energy distribution to become unbalanced, and further, cause the sensor network to become disconnected. To the best of our knowledge, it remains an unaddressed issue as how to distribute the broadcast load evenly on nodes so that the energy distribution is balanced and the lifetime of the sensor network is maximized. In this paper, we develop a protocol to address this issue.

The construction of a query tree using the existing proposals involves a fairly large number of messages that need to be exchanged between nodes. Thus periodically re-constructing the whole query tree is not acceptable due to energy constraints and scalability issues. In this paper we propose a Dynamic Query-tree Energy Balancing protocol to distributively update the query tree structure in order to avoid uneven energy depletion and the associated problems of disconnected networks and shortened network lifetime. The DQEB protocol scales well in that all of its operations are conducted locally and do not require any global information for each update. It is an energy-aware protocol that dynamically and distributively updates the query tree. The key contribution of this paper is that the proposed algorithm globally balances the energy distribution across the sensor network, and locally minimizes the energy depletion rate. Thus both nodes and the sensor network's lifetime is prolonged. The paper also presents a model for a sensor network's energy consumption rate, which facilitates the analysis on the cost of broadcasting.

The rest of this paper is organized as follows. Section II presents the necessary assumptions for our algorithm and a model for the energy consumption rate of the query tree, which forms the groundwork of our algorithm. Section III presents and analyzes the proposed Dynamic Query-tree Energy Balancing Algorithm. In Section IV we develop the algorithm and evaluate the effectiveness of the algorithm. In the last section we present the concluding remarks.

II. Background and Energy Consumption Model

While many architectures and protocols have been developed for sensor networks, for a majority of their applications, these networks share certain characteristics which affect the design of querying mechanisms. First, querying and reporting is the primary working mechanism in sensor networks and the energy cost of these protocols is a critical measure to evaluate their performance. Second, attribute based naming is specially important for sensor networks where, owing to the large number or sensors, it is typically impractical to address an individual sensor. In fact, in many application scenarios users are more interested in event information in a given area, instead of that from a specific sensor. Finally, it is generally assumed that there exists a **sink** node that acts as the interface between the sensor network itself and any external control unit and thus usually initiates the queries. We now enumerate the specific assumptions made in this paper.

A. Assumptions

We develop our distributive query tree energy balancing algorithm based on following assumptions and definitions:

- **A Cluster Based Structure** is assumed for the sensor network. However, we do not make any assumptions on the cluster forming algorithm. Each cluster is managed by its own cluster head and our algorithm only involves the cluster heads, which contributes to its scalability.
- **Nodes** have uniform hardware, software and battery capacity. The term "node" in this paper will mean a cluster head. Each node has a unique ID obtained during the sensor network's initialization period.
- **Leaf and Non-leaf Nodes** are respectively defined as nodes that forward and do not forward query messages. A special non-leaf node is the sink node that initiates the queries.
- **A Query Tree** is assumed to exist with the sink node as the root and all nodes in the network being either leaf or non-leaf nodes of the tree. All queries are first broadcasted to the cluster heads who are then responsible for forwarding the queries to the sensors in their cluster. Restricting the query tree to only cluster heads simplifies the querying process and contributes greatly to the algorithm's scalability.
- **Weight of a Node** represents the remaining lifetime of a node's battery and is numbered between 0 and 1. The weight of a node is inversely proportional to its remaining battery life. When a node is first deployed and has a fully charged battery, its weight is initialized to 0.
- **Broadcasting** is the means of disseminating the query and forms the communication pattern under study in this paper.

B. Model for Energy Consumption with Query Broadcast

There are three types of nodes in a sensor network query tree, namely, sink node, leaf nodes and non-leaf nodes. Leaf nodes only receive broadcast or multicast queries, while non-leaf nodes have to receive and forward them. From our assumptions, the sink node is not power constrained and stays on for the lifetime of the network. Thus, it does not have any effect on the power consumption characteristics of the network within the scope of this paper. The energy of non-leaf nodes is depleted faster than that of leaf nodes since they also forward the queries they receive. Based on this observation and the assumption that all nodes are uniform initially, we can see that the energy distribution across the whole sensor network will become unbalanced if the query tree is static.

As we mentioned earlier, every node is associated with a weight in the range $[0, 1]$, denoted by ω. A node's weight is function of its remaining battery lifetime P (also within [0,1]), and is given by:

$$\omega = (1 - P)^{\beta} \tag{1}$$

where β is the power attenuation factor and determines the rate at which depleting power affects the weight of a node. The choice of Equation (1) to calculate the node weight is motivated by the desire to increase the weight faster as the remaining battery becomes lower. This allows us to avoid overly using an "aging" node by identifying the nodes with critically low powers and designating them as leaf nodes. We do note that there are many other functions possible which have similar characteristics. Our algorithm is not dependent on the choice of the function to represent ω as long as the weight of a node is an increasing function of the drainage of the node battery.

The energy cost of broadcast depends on the number of leaf and non-leaf nodes in the query tree as well as the amount of remaining battery power at a node. For instance, the cost of broadcasting a query is higher for a node when it has very little power remaining than when its battery is fully charged, specially when extending the network lifetime is an objective. Let L denote the set of all leaf nodes, $\overline{L}$ denote non-leaf nodes and let ω_i be the weight associated with node i. We assume the power for transmitting a query by a node is λ times as much as the power for receiving the same bytes of data [7]. Without loss of generality, we assume receiving each query costs γ units of energy for each node. Thus, the overall cost function for the query tree to broadcast each query can be defined as:

$$C = \sum_{i \in L} \gamma\omega_i + \sum_{i \in \overline{L}} (\lambda + 1)\gamma\omega_i = \gamma\Big(\sum_{i \in L \cup \overline{L}} \omega_i + \lambda \sum_{i \in \overline{L}} \omega_i\Big) \tag{2}$$

In Equation (2), $L \cup \overline{L}$ includes all nodes except the sink node, so the first term is a constant in that it represents the overall energy consumed by all nodes to receive a broadcast query.

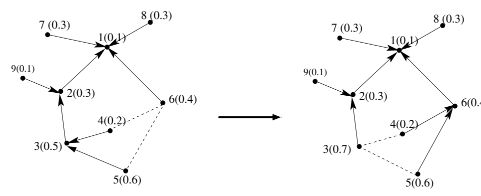

Fig. 1. Part of a Query Tree

The second term is the energy used by all non-leaf nodes to forward the broadcast query. We can see that minimizing the overall energy consumption per query is equivalent to minimizing the cost function in Equation (2). The optimal is achieved when the sum of the weights of the non-leaf nodes is minimum.

If a non-leaf node's battery drains faster and results in a faster increasing weight as compared to leaf nodes, Equation (2) suggests an intuitive strategy to counteract this energy unevenness. When the weight of a non-leaf node increases beyond a certain level, the possibility of converting it to a leaf-node to suppress its energy drainage should be considered. In case such a decision is made, all its children in the query tree have to be detached and switched to new parents. In the next section, we devise a protocol to achieve this objective.

III. Dynamic Query-tree Energy Balancing Algorithm

A. *Algorithm-related Definitions and Design Overview*

As discussed earlier, energy of non-leaf nodes drains faster than that of leaf nodes. Thus, according to Equation (1), the weight of non-leaf nodes tends to become higher than leaf nodes' as time elapses. This leads to unbalanced energy distribution across the sensor network, eventually causing some nodes to expend their energy before others thereby making the sensor network disconnected. Take Fig. 1 as an example. Here solid lines represent edges of the query tree, while dashed lines indicate that the two nodes are within each other's transmission range. The arrow points from child to parent. The label attached on each node includes the node ID and its associated weight (number in the parenthesis). Nodes 4 and 5 are within the transmission range of both node 3 and 6. Initially, nodes 4 and 5 are children of node 3. After forwarding some broadcast queries, node 3's weight is increased by a certain amount (0.2 in our example). According to cost function of Equation (2), if nodes 4 and 5 are now switched to node 6 as their new parent, the overall cost of a broadcast will be decreased by 0.3λ as compared to the original tree. To exploit such energy savings, our algorithm mainly focuses on how to identify an "aging" non-leaf node, find new parents for its children which decreases the overall cost and finally switch the aging node to a leaf node so that both the node and sensor network's lifetime are prolonged. This strategy also adds robustness to the whole tree in that the disconnection of the tree could be largely prevented. Recall that in Equation (1) a "dying" node has a considerably higher weight over other nodes. With energy balancing, it has a much higher chance to stay as a leaf node when it dies. Thus the risk of tree disconnection can be minimized to a considerable extent. All these decisions should be taken distributively and are accomplished by re-arranging the query tree structure in a local area to lower the overall cost. The query tree's updating procedure is triggered when the weight of a non-leaf node increases by a certain amount. To facilitate our presentation, we first define some terminology:

- **Update Coordinator (UC):** the node in charge of re-arranging the tree structure in its neighborhood. It is responsible for collecting information from its children, executing the update scheme and dispatching update instructions to the affected nodes. In our protocol, an aging non-leaf node whose weight has recently increased by a certain amount serves as the UC and triggers the algorithm described later in this section.
- **Alternative Parent (AP):** a node qualified to be the new parent of UC's children. A node may have more than one AP's.
- **Designated Parent (DP):** the node which is designated by the UC to be the new parent of UC's children.

Following are necessary procedures to dynamically update the query tree:

1) Collection of information regarding the weights of its children and their neighbors by the UC
2) Execution of the update algorithm by the UC for modifications to the query tree which reduce the overall cost
3) If the step above succeeds, UC informs its children to switch to their respective DPs. If the update algorithm fails, no actions are taken by the UC and it stays as a non-leaf node.

The first procedure is accomplished by the algorithm described in Section III-B, while the other two procedures are accomplished by the algorithm described in Section III-C. Both of the two algorithms are designed to operate distributively, which makes them highly scalable.

B. *Neighborhood Information Synchronization Algorithm*

In this section we describe our "Neighborhood Information Synchronization Algorithm" (NIS), which is designed to update and synchronize state information among neighboring nodes. This information will be used by our Dynamic Query-tree Energy Balancing Algorithm in the next section.

When the query tree is first set up and stabilized, each node i broadcasts its state information to its neighbors. This information includes the node i's ID, weight and route to the root. Each node also maintains a "Neighborhood Information Table" (NIT) where it stores the state information obtained from each of its neighbors. In order to detect node failures, NIS also requires the exchange of periodic "hello" messages between a non-leaf node and its children. If a node's state information changes in the interval between hello messages, the hello message is substituted by the changed information. These messages also serve the purpose of notifying that the node is alive and avoids the need to transmit additional, explicit hello messages.

As mentioned earlier, a non-leaf node will trigger a tree re-arranging procedure when its weight increases by a certain amount. The node now becomes a UC and invokes NIS to do following:

1) Send a request message to all its children to ask them for all their APs' information
2) Collect responses from its children and save them in UC's NIT

The NIS algorithm tries to achieve a compromise between energy consumption and information collection latency. Each node maintains its one hop neighbors' information and updates them if anything changes. Information about an AP is sent only on demand, which is after the UC triggers the re-arranging procedure. Thus, compared to sending such information periodically, the proposed mechanism consumes lower energy.

C. Dynamic Query-tree Energy Balancing Algorithm

The state information of the neighbors collected using the information synchronization algorithm is used by the DQEB algorithm to dynamically update the query tree. The DQEB algorithm has two components, the first of which tries to update the query tree to minimize the cost function and the second part which informs the UC's children about any such updates. The second step of DEQB is trivial in the sense that it can be achieved simply by broadcasting any changes to the tree which might have occured. The core part of the DEQB algorithm is to find DPs for each of the UC's children and evaluate the cost function of Equation (2). This is done using a greedy algorithm which we now describe.

1) A Greedy Algorithm for Parent Selection: The parent selection algorithm developed in this section is analogous to a set-covering problem [2]. Our algorithm involves three types of nodes: the UC, UC's children and children's alternative parents. The algorithm is a local algorithm which is run at each UC with its local information. The basic aim of the algorithm is for the UC to select the appropriate APs for its children from all its non-child neighbors. However, the algorithms has other considerations also which are described and addressed next. Consider the network shown in Figure 2 where we assume node i's weight increases beyond the threshold amount and thus triggers the DQEB algorithm. Thus node i tries to detach all its children and switch to being a leaf node in order to reduce its battery depletion rate. As shown in the figure, all solid nodes are involved in this decision process at node i, while others are not (node 2 is not involved since it is the parent of node i). Thus, the problem is how to find DPs for all i's children such that the cost function of Equation (2) is minimized. Each node is associated with a weight value. Also, we use $C_1, C_2, C_3, \cdots, C_n$ to denote UC's children and $A_1, A_2, A_3, \cdots, A_m$ to denote their APs. We also define the sets $C = \cup_{i=1}^{n} C_i$ and $A = \cup_{i=1}^{m} A_i$.

Weights of nodes in set A are denoted by $\omega_1, \omega_2, \omega_3, \cdots, \omega_m$. The weights of non-leaf nodes are scaled by multiplying a small number ϵ to their original weights defined by Equation (1), while weights of leaf nodes are kept at their original value. This ensures that non-leaf nodes have a higher likelihood of being selected as a DP while choosing DPs for the children of node i. Non-leaf nodes are preferred as a DP because children nodes can be switched to it without introducing any extra cost. We also denote the degree of node A_j by d_j, $1 \leq j \leq m$, which is the number of neighbors of C_j that are in set C. Our algorithm chooses a subset of nodes from A as the DPs that serve as the new parents for all of the UC's children, with the cost function minimized. Intuitively, the desirability of choosing node A_j as a DP is inversely proportional to the ratio ω_j/d_j since it gives preference to nodes with lower weights and those with larger degrees. This suggests a recursive algorithm as follows:

1) Set $K = \phi$
2) If $C = \phi$ or $A = \phi$, then stop. Otherwise find A_j with the least ratio ω_j/d_j and add it to K.
3) Remove A_j from A. Remove all of A_j's neighbors that are still in set C, add them to set A and return to the step above. Update d_1 through d_m.

This algorithm returns a set K, which is the set of all DPs. After the algorithm returns, if $C \neq \phi$ and $A = \phi$, or the overall cost as computed by Equation (2) is not reduced, it implies that we could not find a DP, or DPs good enough for all of UC's children. Thus, the UC will do nothing and stay as the non-leaf node. If $C = \phi$ and the overall cost is decreased, UC will become a leaf node and all its children are switched to new parents given by set K.

The above algorithm falls in the class of weighted greedy algorithms and it is clear that the algorithm is distributed and does not require any global information. However, it cannot always match the performance of centralized algorithms which are guaranteed to give the optimal solution, though at the cost of additional computational and communication complexity. The proposed algorithm tries to achieve a tradeoff between the optimality of the solution and the scalability, energy consumption and energy balance of the nodes.

2) Choosing and Differentiating APs: The greedy algorithm described in the previous section utilizes only local information in its decision process. This can lead to some unforeseen consequences which might hinder the desired working of the DEQB algorith. For example, consider the network in Figure 2 with node i as the UC. Now, node 5 reports node 6 as one of its APs but if node 6 is chosen as node 5's DP by the DEQB algorithm, it is possible that node 5 will become disconnected from the network after node i (the UC) becomes a leaf node. This is due to the fact that node 5's parent, node 6, is actually connected the the root *via* node i. This problem suggests that we have to differentiate between APs and deal with them accordingly. Now, APs of each i's children can be classified into the following 3 categories:

- **Sibling AP:** Both the node and its AP are UC's children. In this case, the AP is called the node's sibling AP; e.g., in Fig. 2 node 4 and node 3 are each other's sibling APs;
- **Offspring AP:** An AP that is the corresponding UC's offspring, but not its child; e.g., in Figure 2 node 6 is 5's

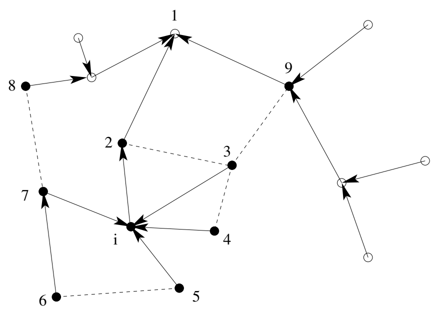

Fig. 2. Example topology showing three types of APs.

offspring AP since node 6 is UC node i's grandchild.

- **Independent AP**: An AP that is neither a sibling AP nor an offspring AP is defined as an independent AP; e.g., in Figure 2 node 9 is node 3's independent AP;

The major difference between these APs is that for independent APs the UC is not included in their route to the sink node which is not the case for the sibling and offspring APs. Thus independent APs are the most desirable nodes from the set of APs to be switched to DPs. In our greedy algorithm we incorporate this by simply adding the independent APs to set A during initialization. For the sibling APs, both the node and its corresponding sibling APs start off in set C. It is easy to verify that our algorithm guarantees that a sibling AP is connected to the query tree before it can be selected as a DP. This prevents a node and its sibling AP from being designated as each other's DP. The selection of offspring APs is based on similar considerations as in the case of sibling APs.

3) Resolving Update Conflicts: The distributed nature of the algorithm described above and its dependence on only local information makes it highly scalable. On the other hand, it also brings up a potential problem: what happens if a node is concurrently involved in more than one update procedure? This can easily lead to conflicting assignments and many of the UCs might base their decisions on stale information. Consequently, it is also possible that the network gets disconnected or its overall cost increases because of the use of stale information.

In order to resolve these conflicts, we introduce a "lock" mechanism. With this procedure, once the UC triggers the updating algorithm and its NIC algorithm requests all its children for their APs' information, all children will reply with the required information. However, once this information is transmitted, the nodes *freeze* themselves and do not respond to similar requests from other UCs while they stay in this state. This guarantees that one node will not be involved into more than one updates at the same time. The nodes stay in the frozen state till they either receive the present UC's DP information or they time out. Any UC that does not get a response from some of its children waits for a given time and starts over.

IV. Simulation Results

To evaluate the performance of the proposed algorithm, we use a sensor network with 1000 nodes, uniformly distributed in an area of 600×600 meters. We assume that a node's power is within the range [0,1] and they initially have full power. Since the network is connected, we can construct an initial query tree, with the sink node as the root. In our simulations $\lambda = 3$ and $\beta = 3$. The root generates the broadcast traffic (queries) at fixed intervals. The following metrics and definitions are used to evaluate the performance of the DQEB algorithm:

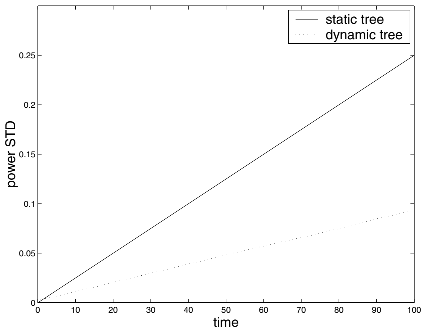

Fig. 3. power balancing of nodes across the network

- **Death Rate:** is defined as the percentage of dead nodes across the whole sensor network. A node is considered dead when its remaining power drops below some predefined threshold. The sensor network is considered dead when nodes' death rate reaches a predefined threshold.
- **Lifetime:** indicates the time it takes for the death rate to reaches a given threshold. It is a non-decreasing function of the predefined death rate threshold.
- **Static Tree:** refers to the case when the broadcast tree is static and our algorithm is NOT applied.
- **Dynamic Tree:** refers to the case when our DQEB protocol is applied and the query tree structure changes dynamically.

A. Results and Discussion

Since our algorithm is mainly designed to balance energy distribution across the sensor network and to prolong the lifetime of the whole network, our simulations focus on the performance improvement regarding two aspects: the sensor network's lifetime and energy distribution evenness. For a given sensor network topology, the performance of a dynamic query tree is compared with that of a static tree. We also investigate network connectivity's influence on the lifetime. When calculating the lifetime, we also take into account the energy consumed by the protocol itself.

1) Power Distribution Balance vs. Time: In Fig. 3 we show how the remaining power at each node is balanced for the static and dynamic trees as the network keeps broadcasting queries. The figure plots the standard deviation for all nodes' remaining power as a function of time, reflecting how evenly the power is distributed among the nodes. From the figure it is evident that the dynamic tree outperforms the static tree and has consistently lower values for the deviation in the power. It implies that with our algorithm nodes' power are consumed more evenly, which is instrumental in prolonging the lifetime of the network.

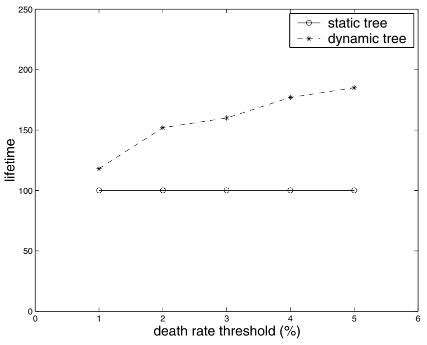

Fig. 4. Lifetime vs. death rate threshold

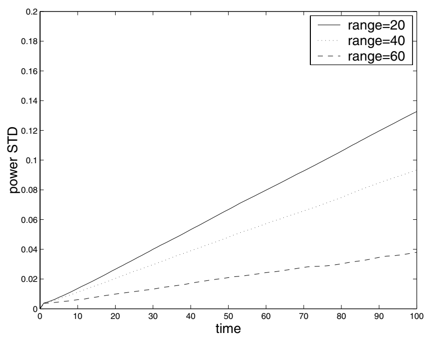

Fig. 5. Power STD vs. connectivity

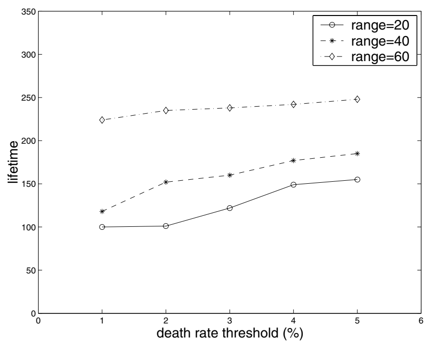

Fig. 6. Lifetime vs. connectivity

2) Lifetime vs. Death Rate Threshold: As mentioned earlier, the lifetime is a non-decreasing function of the death rate threshold. In Fig. 4 we plot the lifetime of the sensor network as a function of the the death rate threshold. As can be seen, when the death rate threshold is set as 5%, the lifetime almost doubles with our protocol applied. With a static tree, all non-leaf nodes die at the same time since they have the same initial battery power and broadcast load. This explains why in Fig. 4 the network lifetime with a static tree does not change with the death rate threshold. With a dynamic tree, the broadcast load is balanced among all nodes rather than excessively exploiting certain nodes thereby prolonging the network's lifetime.

3) Impact of Network Connectivity: We now investigate the influence of network connectivity on power balancing and network lifetime. In general, there are two ways of increasing connectivity in a sensor network: increase nodes' transmission range or increase the node density. These two methods are equivalent while comparing the performance our protocol. For our simulations, we chose the method of increasing the nodes' transmission range and in Fig. 5 we show the standard deviation of the battery power at each node with the transmission range as 20, 40 and 60 meters (equal to 1/30, 1/15, 1/10 of the region size, respectively) for our protocol. We note that better the connectivity, the more balanced is the power distribution. This is also intuitive since better connectivity implies statistically more nodes are available to share the broadcast burden in a given sub-area. Finally, in Fig. 6 we show the change in the network's lifetime as a function of the connectivity. We note that higher connectivity increases the network's lifetime, again owing to the fact that more nodes are available for sharing the broadcast burden in a given region.

V. Conclusion

In this paper, we proposed an energy-aware, distributed protocol to dynamically update query tree structures in sensor networks. The protocol aims at balancing the broadcast load and achieving even battery power distribution across all nodes to maximize the sensor network's lifetime. Based on the observation that the energy of non-leaf nodes drains faster than that of leaf nodes, we propose a weighted greedy algorithm that updates the broadcast tree taking the remaining battery power at each node into account. Decisions are taken at each non-leaf node to locally minimize the cost of the broadcast tree by switching a non-leaf node with low remaining power to a leaf node so that its energy depletion rate is decreased. The children of this node are then assigned new parents found by the proposed algorithm. Our simulations show that with the proposed protocol, the network lifetime is greatly improved considerably while achieving a considerably higher level of balance in the energy distribution when compared with that obtained by a static tree.

References

[1] Guha, S., Khuller, S. "Approximation Algorithm for Connected Dominating Sets," *Algorithmica, 20(4)*, pp374-387, 1998;

[2] Chvatal, V. "A Greedy Heuristic for the Set-covering Problem," *Mathematics of Operations Research,* Vol. 4, No. 3, Aug. 1979

[3] Li, F., Nikolaidis, I., "On Minimum-Energy Broadcasting in All-Wireless Networks," *Proceedings of the 26th Annual IEEE Conference on Local Computer Networks* Nov. 2001, Tampa, USA.

[4] Li, F., Wu, K, "Reliable, distributed and energy-efficient broadcasting in multi-hop mobile ad hoc networks," *Local Computer Networks Proceedings of LCN 2002, 27th Annual IEEE Conference on,* pp. 761-769, Nov. 2001.

[5] Wieselthier, J., G. Nguyen, A. Ephremides, "On the Construction of Energy-Efficient Broadcast and Multicast Trees in Wireless Networks,"*Porceedings of IEEE INFOCOM 2000.*

[6] Stemm, M., Katz R., "Measuring and Reducing Energy Consumption of Network Interfaces in Hand-held Devices," *IEICE Transactions on Communications,* vol. E80-B, no.8, pp.1125-1131, Aug.1997

[7] Kasten O.,"Energy Consumption", *http://www.inf.ethz.ch/k̃asten/research/bathtub/energy_consumption.html,* Eldgenossische Technische Hochschule Zurich.

[8] Chang, J., Tassiulas, L., "Energy Conserving Routing in Wireless Ad-hoc Network," *Proceedings of IEEE INFOCOM 2000* ACM, 2000

[9] Heinzelman, W., Chandrakasan, A., and Balakrishnan, H., "Energy-efficient Communication Protocol for Wireless Microsensor Networks. *In Proceedings of the 33rd Hawaii International Conference on System Sciences,* Maui, Hawaii, January 2000

[10] Li, L., Bahl, V., Wang, Y., Wattenhofer, R., "Distributed Topology Control for Power Efficient Operation in Multihop Wireless ad-hoc Networks," *Proceedings of IEEE INFOCOM 2001,* April 2001

[11] Singh, S., Raghwvendra, C., and Stepanek, J., "Power-aware Broadcasting in Mobile ad hoc Networks," *Proceedings of IEEE PIMRC'99,* Osaka, Japan, Set. 1999

[12] P.-J. Wan, G. Calinescu, and O. F. X.-Y. Li., "Minimum-energy Broadcast Routing in Static ad hoc Wireless Networks," *In IEEE INFOCOM 2001,* Anchorage, Alaska, April 2001.

[13] O. Egecioglu and T. Gonzalez, "Minimum-energy Broadcast in Simple Graphs with Limited Node Power," *In Proceedings of IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2001),* pages 334-338, Anaheim, CA, August 2001.

[14] R. J. Marks, A. K. Das, M. El-Sharkawi, P. Arabshahi and A. Cray, "Minimum power broadcast trees for wireless networks: optimizing using the viability lemma,*In Proceedings of IEEE International Symposium on Circuits and Systems,* pp. 273-276, Scottsdale, Arizona, May 2002.

[15] W. Liang, "Constructing minimum-energy broadcast trees in wireless ad hoc networks, *In Proceedings of ACM MOBIHOC'02* pp. 194-205, Lausanne, Switzerland, June 2002.