**RANK-BASED DATA GATHERING IN WIRELESS SENSOR NETWORKS**

Jeremy Bouldin & Natarajan Meghanathan

Department of Computer Science, Jackson State University, Jackson, MS 39217, USA.

**ABSTRACT**

We propose a Rank-Based Data Gathering (RBDG) algorithm for wireless sensor networks with randomly distributed sensor nodes. For every round of data communication, the algorithm functions as follows: A set of sensor nodes or vertices are assigned a random rank between 0 and 1. A link is formed between any two nodes if they are within each other’s transmission range. If a sensor node has the highest rank among it neighbors, then it is considered an associate node, else it is categorized as a leaf node. Next, we form a complete graph among the associate nodes with edge weights representing the physical Euclidean distance between the nodes. We run a minimum spanning tree algorithm on this complete graph and transform it to a rooted directed data gathering tree with the root being the sensor node with the highest residual energy. We also developed an energy-efficient version of the rank-based data gathering algorithm (EE-RBDG) wherein the rank of a node is the sum of the random number (between 0 to 1) assigned for the node and the fraction (also from 0 to 1) of the initial energy currently available at the node. Simulation results indicate that the E-RBDG algorithm performs better than the RBDG and other well-known data gathering algorithms with respect to network lifetime, delay and energy*delay per round.

**Keywords:** Rank, Data gathering, Energy-awareness, Network lifetime, Delay per round

1. **INTRODUCTION**

A wireless sensor network is an aggregation of nodes, equipped with sensors, randomly or evenly distributed across a vast area used to monitor disaster areas, terrorist attack areas, forest fires and etc. The sensor nodes could be located at random locations and relay their information to a central base station (i.e., a sink node) that is usually far from the region of sensor nodes. Sensor nodes usually have a few basic commodities that come along with them: one or more sensors, a radio transceiver for communication, a microcontroller for computation and decision making and battery for energy.

Data gathering algorithms are usually run in several rounds. In each round, data from all the sensor nodes are gathered and then forwarded to the sink. Data gathering algorithms are categorized based on the type of communication structure they use: clusters, grid, chain, connected dominating sets and trees. Different types of clustering and grid algorithms have been proposed in the literature. In this paper, we compared our proposed rank-based data gathering algorithms with two well-examined and widely studied data gathering algorithms in the literature: Low Energy Adaptive Clustering Hierarchy (LEACH) and Power-Efficient Gathering in Sensor Information Systems (PEGASIS).

In this paper, we propose a Rank Based Data Gathering algorithm (RBDG) for data collaboration in wireless sensor networks and also propose an energy-efficient version (EE-RBDG) of the same. The RBDG algorithm works as follows: We randomly distribute a set of sensor nodes within the given wireless network. For every round of data gathering, each sensor node is assigned a rank between 0 and 1. The data gathering tree (DG-tree) is formed through a sequence of steps: (i) A node becomes an associate node if it has the highest rank among all its neighbors; (ii) If a non-associate node has a neighbor that has been elected as an associate node during the first step, then the non-associate node becomes a leaf node for that neighbor; (iii) If a node is neither selected as an associate nor as a leaf node in steps 1 and 2 respectively, then the node becomes an associate node by default. This usually occurs for a very small number of nodes in an extremely large sensor network field. The data aggregation tree is comprised of leaf nodes—nodes whose rank is lower than a neighbor node, associate nodes — nodes whose rank is highest among neighbors or chosen by default, and root node — node with the highest available energy at for the particular round.

During data communication, leaf nodes forward their data to their associate node. The associate nodes then send their data to an upstream associate node according to a data gathering tree formed based on the complete graph of the associate nodes with the edge weights being the physical Euclidean distance of the constituent nodes. The root node gathers all the aggregated data from the downstream associate nodes and then sends the data to the sink, where the data is further processed depending on the application requirements. Data communication occurs in rounds, and a RBDG is executed for subsequent rounds if every node in the network still remains alive (with a positive available energy level). We also propose an energy-aware version of the rank-based data gathering algorithm (EE-RBDG) wherein for every round, the rank of a node is the sum of the random number (from 0 to 1) assigned for the node as 

---

**Institution:** Department of Computer Science, Jackson State University, Jackson, MS 39217, USA

**Keywords:** Rank, Data gathering, Energy-awareness, Network lifetime, Delay per round
in RBDG and the fraction (from 0 to 1) of the initial energy currently available at the node during the beginning of the round. Once the rank of a node is determined, the rest of the EE-RBDG algorithm to determine the associate nodes, leaf nodes and the data gathering tree is the same as that of the RBDG algorithm. We compare RBDG and EE-RBDG with LEACH and PEGASIS through extensive simulations.

The rest of the paper is organized as follows: Section 2 reviews the literature on LEACH and PEGASIS algorithms for data gathering in wireless sensor networks. Section 3 describes the proposed RBDG algorithm in detail and also discusses the energy-efficient variant of the same. Section 4 presents the simulation results of the RBDG and EE-RBDG algorithms compared along with LEACH and PEGASIS. Section 5 concludes the paper. Throughout the paper, the terms ‘data gathering’, ‘data aggregation’ and ‘data fusion’ are used interchangeably. Similarly, the terms ‘node’ and ‘vertex’, ‘edge’ and ‘link’ are used interchangeably. They mean the same.

2. LITERATURE REVIEW

As the sensor nodes are energy-constrained (have limited energy), data gathering are often needed to forward the sensed data to the sink. Many approaches have been taken to solve this problem, but nearly all of them have unwanted drawbacks. Some of the more recent and commonly used data gathering algorithms are based on clustering. These algorithms group the sensor nodes into different clusters and allow them to transmit their data to a cluster head, which communicates to the sink. The LEACH algorithm [1] is one of the widely known and well-studied clustering-based data gathering algorithms. In LEACH, each round has the following two phases: a set-up phase, when clusters of sensor nodes are formed with the assignment of a cluster head for each cluster and a steady-state phase, when the data collected from the sensor nodes in each cluster is transmitted to the sink through direct transmission from the cluster heads. In PEGASIS, each round involves the formation of a chain of the sensor nodes in the field, starting from the sensor node that is farthest from the sink. The chain of sensor nodes is formed using a greedy heuristic based on the distance between the sensor nodes. The original PEGASIS algorithm [2] resulted in huge delay as data moves across the complete chain of sensor nodes before getting transmitted to the sink. For CDMA (Code Division Multiple Access) systems [3], PEGASIS has been later improved using a chain-based binary scheme to minimize the delay incurred and to reduce the energy*delay metric [4]. In the chain-based binary approach, a round comprises of \( \log N \) levels, where \( N \) is the number of nodes in the network. For data gathering in each round, each node transmits to a close neighbor in a given level of the hierarchy. Nodes that receive data at a given level are the only nodes that rise to the next level. At the top level, there will be only one node that will remain as the leader and it will transmit the aggregated message to the sink.

The distance-based chain formation heuristic of PEGASIS is prone to an increase in the physical distance between successive nodes as the chain progresses away from the starting node. This can lead to higher energy consumption per round. The chain-based binary version of PEGASIS consumes more energy per round compared to its original version [4]. The energy consumed per round of LEACH is significantly more than that consumed in PEGASIS and its chain-based binary scheme [4]. If multiple cluster-heads are selected, even though the delay per round would be low because of reduced size of a cluster, several cluster-heads would be transferring data over long-distances to the sink. On the other hand, if few cluster-heads are selected, the sensor nodes may have to transmit over long-distances to reach the nearest cluster-head. Due to competition in each cluster, the delay might also increase. The above qualitative analysis of the energy consumption per round for the LEACH and PEGASIS algorithms suggests that the number of rounds sustained by the sensor network running these algorithms before the failure of the first sensor node (due to the exhaustion of battery charge) need not be maximum. There is a possibility of increasing the number of rounds before the first node failure by reducing the energy consumed per round and choosing only the nodes with a relatively higher energy level for transmission over long-distances. This observation formed the motivation of our work in this paper.

3. -BASED DATA GATHERING ALGORITHM

The system model comprises of a network of nodes that are uniform-randomly distributed over a region. Each node is assigned a unique identifier. For every round of data gathering, each node is assigned a random number (from 0 to 1) and that decides the rank of the node; the larger the value of the random number – the higher the rank of the node. There exists an edge between two nodes in the network graph if and only if the physical Euclidean distance between the two nodes is less than or equal to the transmission range of the nodes. As the network is static in nature, the topology of the network remains the same across several rounds of data gathering before the failure of a sensor node due to exhaustion of battery charge. We refer to the round of first node failure as the network lifetime. Figure 1 represents a snapshot of a network topology of 16 sensor nodes (the identifier is a unique character label inside the circle) and their rank values (indicated outside the circle).
The sequence of steps to be followed for the proposed rank-based data gathering algorithm is illustrated in Figure 5 and we also show the execution of this algorithm (in Figures 2, 3 and 4) on the sample network graph of Figure 1. In Figures 2, 3 and 4, the nodes with a dark-shaded circle are associate nodes and the nodes with dotted circle and a white background are the leaf nodes.

Step 1: A node becomes an associate node if it has the highest rank among all its neighbors.
Step 2: For each node \(v\) that has not been selected as an associate node in Step 1, if there exists a neighbor node \(u\) that has been selected as an associate node in Step 1, then node \(v\) becomes a leaf node for node \(u\). In case of a tie, the neighbor node that has the highest rank is chosen as the upstream associate node.
Step 3: If a node cannot be assigned as a leaf node for any associate node selected in Step 1, then the node selects itself to be an associate node and is added to the list of associate nodes.
Step 4: We form a complete graph (link between every pair of nodes) involving the associate nodes formed from Steps 1 and 3.
Step 5: Execute Kruskal’s minimum spanning tree algorithm on the complete graph formed in Step 4.
Step 6: Transform the minimum spanning tree formed in Step 5 to a rooted directed data gathering tree with the root being the node with the largest available energy.

3.1. Pseudo Codes for the Implementation of the Rank-Based Data Gathering Algorithm

Figure 6 illustrates the pseudo code for the steps (1, 2 and 3) involving the selection of the associate and leaf nodes. Figure 7 is the pseudo code for the minimum spanning tree algorithm executed on a complete network graph of associate nodes (Step 5). Figure 8 illustrates the pseudo code of the Breadth First Search algorithm used to transform the undirected minimum spanning tree of associate nodes to a rooted directed tree (Step 6) with the root being the node with the highest energy. In our example in Figure 4, the root node is also incidentally chosen as the node with the highest rank.
Input: Snapshot of the Network Graph \( G = (V, E) \); where \( V \) is the set of vertices and \( E \) is the set of edges

Auxiliary Variables and Initialization:
- Node-List, Adjacency-List of neighbors for each node,
- \( \text{canBeAssociate} = \text{true} \)
- \( \text{Associate-Node-List} = \emptyset \)
- \( \text{Leaf-Node-List} = \emptyset \)

Begin Selection – Associate and Leaf Nodes

for every vertex \( u \in \text{Node-List} \) do
  rank\( (u) \) = random number assigned from 0 to 1
end for

for every vertex \( u \in \text{Node-List} \) do
  \( \text{canBeAssociate} = \text{true} \)
  for every vertex \( v \in \text{Adjacency-List}(u) \) do
    if (rank\( (v) \) > rank\( (u) \)) then
      \( \text{canBeAssociate} = \text{false} \)
    end if
  end for
  if (\( \text{canBeAssociate} = \text{true} \)) then
    \( \text{Associate-Node-List} = \text{Associate-Node-List} U \{u\} \)
  end if
end for

for every vertex \( u \in \text{Node-List} \) and \( u \notin \text{Associate-Node-List} \) do
  candidateAssociateNode = NULL
  candidateRank = rank\( (u) \)
  for every vertex \( v \in \text{Adjacency-List}(u) \) and \( v \in \text{Associate-Node-List} \) do
    if (rank\( (v) \) > candidateRank) then
      candidateRank = rank\( (v) \)
      candidateAssociateNode = \( v \)
    end if
  end for
  if (candidateAssociateNode != NULL) then
    \( \text{Leaf-Node-List} = \text{Leaf-Node-List} U \{u\} \)
  end if
end for

for every vertex \( u \notin \text{Associate-Node-List} \) and \( u \notin \text{Leaf-Node-List} \) do
  \( \text{Associate-Node-List} = \text{Associate-Node-List} U \{u\} \)
end for

return \( \text{Associate-Node-List}, \text{Leaf-Node-List} \)

End Selection – Associate and Leaf Nodes

Figure 6. Pseudo Code for the Selection of Associate Nodes and Leaf Nodes

Input: Complete Graph of Associate Nodes \( G_A = (V_A, E_A) \); where \( V_A \) is the \( \text{Associate-Node-List} \) and \( E_A \) is the set of edges between any two nodes on the \( \text{Associate-Node-List} \)

Begin Kruskal Algorithm

\( E_{A, \text{MST}} \leftarrow \emptyset \) // Initialize the set of edges (that form the minimum spanning tree) to null set

for each vertex \( v_i \in V_A \) do
  Component \( (v_i) \leftarrow i \)
end for

Sort the edges of \( E_A \) in the non-decreasing (increasing) order of weights

for each edge \( (v_i, v_j) \in E_A \), in order by non-decreasing weight do
  if (Component \( (v_i) \) \( \neq \) Component \( (v_j) \)) then
    \( E_{A, \text{MST}} \leftarrow E_{A, \text{MST}} U (v_i, v_j) \)
  end if
end for

141
if \( \text{Component}(v_i) < \text{Component}(v_j) \) then
    for each vertex \( v_k \) in the same component as of \( v_j \) do
        Component\((v_k) \leftarrow \text{Component}(v_i)\)
    end for
else
    for each vertex \( v_k \) in the same component as of \( v_i \) do
        Component\((v_k) \leftarrow \text{Component}(v_j)\)
    end for
end if
end if
end for
return \( E_{A, \text{MST}} \)
End \textit{Kruskal} Algorithm

\textbf{Figure 7.} Pseudo Code for the Kruskal Algorithm for Minimum Spanning Tree

\textbf{Input:} Minimum spanning tree (Undirected) \( G_{A, \text{MST}} = (V_A, E_{A, \text{MST}}) \)
\textbf{Output:} A rooted directed minimum spanning tree, \( G_{r, A, \text{MST}} = (V_A, E_{r, A, \text{MST}}) \)
\textbf{Auxiliary Variables:} \textit{Queue-to-Visit}, \textit{tempQueue}
\textbf{Initialization:} \textit{Queue-to-Visit} = \( \Phi \); \( E_{r, A, \text{MST}} = \Phi \)
\textbf{Begin} Breadth First Search
    \textbf{Root} \( r\text{-node} = \{ v \mid \text{Max}_v[\text{energy}(v)] \} \)
    \textit{Queue-to-Visit} = \{ \textit{r\text{-node}} \}
    \textbf{while} (|\( E_{r, A, \text{MST}} | < |V_A| - 1| ) \textbf{do}
        \textit{tempQueue} = \( \Phi \)
        \textbf{for} every vertex \( u \in \textit{Queue-to-Visit} \) do
            \textbf{for} every edge \((u, v) \in E_{A, \text{MST}} \) do
                \textit{tempQueue} = \textit{tempQueue} U \{ v \}
                \( E_{r, A, \text{MST}} = E_{r, A, \text{MST}} U \{ u \rightarrow v \} \)
            end for
        end for
        \textit{Queue-to-Visit} = \textit{tempQueue}
    \textbf{end While}
    return \( G_{r, A, \text{MST}} = (V_A, E_{r, A, \text{MST}}) \)
\textbf{End} Breadth First Search

\textbf{Figure 8.} Breadth First Search Algorithm to Transform the Minimum Spanning Tree of Associate Nodes to a Rooted Directed Tree for Data Gathering

\subsection*{3.2. Energy-Efficient Rank-based Data Gathering}

We introduce energy awareness to the RBDG algorithm by computing the rank of a node at the beginning of a data gathering round to be the sum of the random number (from 0 to 1) assigned for that node at that round and the fraction (0 to 1) of the initial energy currently available at the node. The sum of the two values would now define the rank of a node and the rank would lie in the range \( [0 \ldots 2] \). The larger the rank of a node, the larger is its probability of being chosen as an associate node. The rest of the energy-efficient rank-based data gathering (EE-RBDG) algorithm is similar to that of the RBDG algorithm. For networks of moderate and high density, it is very rare that Step 3 of the RBDG algorithm (pseudo code in Figure 5) is executed to promote a stand-alone node, with none of its neighbors chosen as associate node, to an associate node.

\subsection*{3.3. Delay per Round of Data Gathering}

The delay at a node indicates the number of time slots it takes for the node to receive the aggregated data from all of its immediate child nodes. The delay associated with each of the leaf nodes is 0. We assign one time slot per child node to transfer data to its immediate predecessor intermediate node. We start processing the intermediate nodes from the bottom of the DG-tree. Note that the intermediate nodes at a particular level in the DG-tree are independent of each other and their delay can be computed in parallel. Also, a node is the child node for only one intermediate node in the DG tree. The delay of the DG-tree is one time slot more than the delay at the root node.
Input: Intermediate-Nodes-List, Leaf-Nodes-List, Nodes-All-Levels, rootNode
Output: Delay-per-Round

Auxiliary Variables:
 Delay(u) // Number of time slots it takes for the aggregated data to reach node u
 Node-Level // level of a particular node in the DG tree
 Nodes-At-Level // the list of nodes at a particular level
 Sorted-Delay-Child-Nodes // sorted list (in the increasing order) of the delay of the child nodes
 Temp-Delay // temporary variable to process the delay at a node

Initialization: Delay-per-Round = 0

Begin Computation-Delay-RBDG-Tree

for (every vertex v ∈ Leaf-Nodes-List) do
   Delay(v) = 0
end for

for (Node-Level = Height-DG-Tree-1 to 0) do
   Nodes-At-Level = Nodes-All-Levels(Node-Level)
   for (every vertex u ∈ Nodes-At-Level) do
      Temp-Delay = 0
      Sorted-Delay-Child-Nodes = Φ
      for (every vertex v ∈ Child-Nodes(u)) do
         Insert the tuple {v, Delay(v)} at an appropriate entry in Sorted-Delay-Child-Nodes
      end for
      for (every tuple {v, Delay(v)} in the Sorted-Delay-Child-Nodes list) do
         Temp-Delay = Maximum (Temp-Delay + 1, Delay(v) + 1)
      end for
      Delay(u) = Temp-Delay
   end for
end for

return Delay(rootNode)

End Computation-Delay-RBDG-Tree

Figure 9. Algorithm to Compute the Delay per Round for the RBDG and EE-RBDG Trees

Figure 9 illustrates the pseudo code for computing the delay of the DG-tree. For each intermediate node u at a particular level, we prepare a sorted list of the delay associated with each of its immediate child nodes. The delay associated with the intermediate node is computed through a temporary running variable, Temp-Delay (initialized to zero), as we explore the delay associated with each of the child nodes in the sorted list. For every child node v in the sorted list of the delay, Temp-Delay is set to the maximum of Temp-Delay + 1 and Delay(v) + 1, as we assume it takes one time slot for a child node to transfer its aggregated data to the immediate predecessor intermediate node. The delay associated with the intermediate node u, Delay(u), is the final value of Temp-Delay after we run through the sorted list of the delays associated with the Child-Nodes(u). The above procedure is repeated for all intermediate nodes, from levels one less than the Height of the tree all the way to zero (i.e. the root node). Figure 10 illustrates the execution of the delay computation algorithm on a sample data gathering tree.

Figure 10. Example to Illustrate the Execution of the Algorithm to Compute the Delay per Round

Level

Delay of the DG tree (time slots) = Delay (rootNode) + 1 = 7 + 1 = 8
4. SIMULATION RESULTS

The simulations of the RBDG and EE-RBDG algorithms were conducted in a discrete-event simulator developed by the first author and their performance results were recorded after an execution for several trials. The network size is 100m x 100m. There are 100 sensor nodes that are randomly distributed throughout the network. The sink node is located outside of the sensor network at (50, 300). Each node is assumed to be able to send data to the other node in its immediate downstream nodes in the network. The energy consumption model used is the first-order radio model [5]. The energy expended by a radio to run the transmitter or receiver circuitry is \(E_{\text{elec}} = 50\) nJ/bit and \(E_{\text{amp}} = 100\) pJ/bit/m\(^2\) for the transmitter amplifier. The radios are turned off when a node wants to avoid receiving unintended transmissions. The energy lost in transmitting a \(k\)-bit message over a distance \(d\) is given by: \(E_{\text{TX}} (k, d) = E_{\text{elec}} * k + E_{\text{amp}} * k^2 \cdot d^2\). The energy lost in receiving a \(k\)-bit message is \(E_{\text{RX}}(k) = E_{\text{elec}} * k\) [5]. The above energy consumption model has been earlier used in several simulations of data gathering algorithms (e.g., [1], [2], [4], [6]).

4.1. Performance Metrics

The performance metrics considered are: (i) Network lifetime, measured as the number of rounds the network sustains before the first sensor node dies due to exhaustion of battery charge, (ii) Energy consumed per round, (iii) Delay (in terms of the number of time slots) per round of data aggregation and transmission to the sink, and (iv) energy*delay value per round. For the RBDG and EE-RBDG trees, we also evaluate the impact of the transmission range on the (i) Network lifetime, (ii) Energy consumed per round, (iii) Height of the DG-tree, (iv) Delay per round for the DG-tree, (v) Energy*delay per round and (vi) Number of leaf nodes per round. The results reported in Figure 11 for RBDG and EE-RBDG trees are obtained after 1000 trials of the algorithms for each value of the transmission range per node. The results reported in Figure 12 for RBDG and EE-RBDG correspond to a transmission range of 25m as the algorithms appear to give the best performance with respect to several metrics at this transmission range value. The results reported in Figure 12 for LEACH and PEGASIS do not assume a particular transmission range per node as these two algorithms assume that the nodes can do transmission power control as and when needed. For each trial, the initial energy supplied to every sensor node is 1J.

4.2. Impact of Transmission Range on the Performance of RBDG and EE-RBDG Algorithms

For simulations conducted with higher transmission range per sensor node, we noticed the network lifetime decreased drastically. This could be attributed to the relatively smaller number of associate nodes and a large amount of in-network data aggregation and transmission done by the associate nodes. When simulated for smaller transmission ranges per node, the network lifetime for both RBDG and EE-RBDG trees increases significantly. This could be attributed to the larger number of leaf nodes per associate node and a relatively lower load of data aggregation per associate node. The EE-RBDG trees had a significantly longer lifetime compared to the RBDG trees and this could be attributed to the consideration of the available energy at the nodes as a criterion for their selection as an associate node. Based on the simulation results, the RBDG algorithm could be used for transmission range values less than 30 m and the EE-RBDG algorithm could be used for larger transmission ranges as the energy-awareness of this algorithm could help to extend the network lifetime, compared to RBDG.

When simulated with transmission range of 15 m to 30 m, the network lifetime was higher than other simulations. The network lifetime of the algorithms gets lower for simulations ran with a transmission range of 35 m or more. As we simulated a transmission range of 15 m and 20 m, the average number of leaf nodes reduced, which also leads to a reduced energy consumed per round of data communication. The height of the DG-tree is dependent on the number of leaf nodes that are selected during each round. For simulations of transmission range from 15 m to 25 m, the height decreased slightly which could be attributed to the presence of more leaf nodes at a particular level. The
delay of the DG-tree is also dependent on the transmission range per node as well as the number of leaf nodes, as also the height of the tree.

Figure 11. Impact of the Transmission Range per Sensor Node on the Performance of the RBDG and EE-RBDG Algorithms

4.3. Comparison of RBDG and EE-RBDG with PEGASIS and LEACH

With RBDG, during the course of the simulation, as we randomly assign the rank of the nodes for every round of data gathering, any node has an equal chance of becoming an associate node or a leaf node during a node. With EE-RBDG, we couple this randomness factor with the residual energy available at a node and try to be fair among all the nodes that have higher larger residual energy vis-à-vis nodes with a lower residual energy. The results in Figure 12 are for a sensor network of 100 nodes, in a field size of 100 m by 100 m, transmission range of 25 m and simulated over 100,000 trials. The results greatly show how such a small difference on choosing leader nodes can reflect a network lifetime and delay of a tree. Similar to the LEACH approach on the strategy for choosing leader nodes, EE-RBDG chooses the associate node with the most residual energy and rank combined. The added residual energy factor allows for the node with the highest amount of energy to become an associate node. The network lifetime improved greatly for EE-RBDG compared to LEACH and PEGASIS. The data reflects a difference of over 70% between EE-RBDG and LEACH, and just over 40% difference between RBDG and PEGASIS.

The simulation results also demonstrate the weakness behind the LEACH algorithm. The data also shows how the strategy for LEACH is a failing one when compared to other data gathering algorithms. RBDG, EE-RBDG, and PEGASIS have an almost equal value for the energy consumption per round, whereas LEACH incurs a significantly larger value. However, PEGASIS incurs the largest delay per round of data gathering due to the usage of a chain of sensor nodes for data gathering. Also, the chain is formed using a greedy strategy that could lead to a larger physical Euclidean distance between two successive nodes in the chain as we progress from the origin of the chain to the end of the chain. As a result, the lifetime of the sensor nodes in the PEGASIS chain could be limited, as observed with the simulation results. RBDG and EE-RBDG can yield a relatively larger network lifetime (more than 50% in the case of EE-RBDG) and also incur a lower value for the delay and energy consumed per round. This also leads to a lower energy*delay per round—a metric that represents the tradeoff between the energy and delay per round of data gathering.
Figure 12. Performance Comparison of RBDG and EE-RBDG with LEACH and PEGASIS

4.4. Impact of the Sink Location on the Performance of the RBDG and EE-RBDG Trees

The sink of a wireless sensor network is static and is usually placed outside of the network. We studied the performance of both the RBDG and EE-RBDG trees metrics for different locations of the sink node for a 100 m x 100 m network of 100 nodes and a transmission range per node of 25 m. Simulation results indicated that the significant difference in the network lifetime incurred between the RBDG and EE-RBDG algorithms occurs only when the sink is located at (50, 300) – i.e., outside the network field. When the sink is located inside the network field, either at the center – (50, 50) or at the origin (0, 0), the difference in the network lifetime between the two algorithms is relatively lower. There was no difference in the energy consumed per round when simulated with sink locations of (0, 0) and (50, 50). However, as expected the energy consumed per round when the sink is located inside the network is lower than the energy consumed per round when the sink is located outside the network, i.e., at (50, 300). The latter sink location results in several long-distance communications that could drain the energy reserves at the nodes. The sink locations do not influence the height and delay consumed per round of data gathering as these two metrics are only dependent on the transmission range per node and the density of the network. A lower energy consumed per round for sink locations inside the network also leads to a lower energy*delay per round.

Figure 13. Impact of the Sink Locations on the Performance of the RBDG and EE-RBDG Trees

5. CONCLUSIONS

The high-level contribution of this paper is the development of a rank based data gathering algorithm and the development of an energy-efficient version (EE-RBDG) of the same. After conducting exhaustive simulations, we observe that the network lifetime with EE-RBDG is 3.2 and 2.6 times more than that incurred for LEACH and
PEGASIS respectively. The delay per round of data gathering is significantly lower compared to that of PEGASIS and LEACH. The energy*delay consumed per round of data gathering for both RBDG and EE-RBDG is less than half of that incurred with PEGASIS and LEACH. Compared with LEACH and PEGASIS, both RBDG and EE-RBDG are fair with respect to the usage of the nodes and this reflects in the relatively larger value for the network lifetime, measured as the round of first node failure due to exhaustion of energy reserves. The EE-RBDG algorithm yielded a significantly larger value for the network lifetime compared to the RBDG algorithm when the sink is located outside the network; whereas when the sink is located inside the network or at the origin, the difference in the network lifetime between the two proposed algorithms is relatively less. Overall, the proposed rank-based data gathering algorithm and its energy-efficient can be a significant addition to the list of data gathering algorithms that can simultaneously maximize the network lifetime as well as minimize the energy*delay per round of data gathering.

6. ACKNOWLEDGEMENTS

This research is funded by the U.S. National Science Foundation through grant (CNS-0851646) entitled: “REU Site: Undergraduate Research Program in Wireless Ad hoc Networks and Sensor Networks,” hosted by the Department of Computer Science at Jackson State University (JSU), MS, USA. The authors also acknowledge Dr. Loretta Moore, Dr. Xuejun Liang, Ms. Ilin Dasari and Mrs. Brenda Johnson (all at JSU) for their services to this program conducted during Summer 2010.

7. REFERENCES


