International Journal of Communication Networks and Information Security (IJCNIS) Vol. 4, No. 3, December 2012

Link Expiration Time and Minimum Distance Spanning Trees based Distributed Data Gathering Algorithms for Wireless Mobile Sensor Networks

Natarajan Meghanathan

Department of Computer Science, Jackson State University,
Mailbox 18839, 1400 John R. Lynch Street, Jackson, MS 39217, USA
natarajan.meghanathan@jsums.edu

Abstract: The high-level contributions of this paper are the design and development of two distributed spanning tree-based data gathering algorithms for wireless mobile sensor networks and their exhaustive simulation study to investigate a complex stability vs. node-network lifetime tradeoff that has been hitherto not explored in the literature. The topology of the mobile sensor networks changes dynamically with time due to random movement of the sensor nodes. Our first data gathering algorithm is stability-oriented and it is based on the idea of finding a maximum spanning tree on a network graph whose edge weights are predicted link expiration times (LET). Referred to as the LET-DG tree, the data gathering tree has been observed to be more stable in the presence of node mobility. However, stability-based data gathering coupled with more leaf nodes has been observed to result in unfair use of certain nodes (the intermediate nodes spend more energy compared to leaf nodes), triggering pre-mature node failures eventually leading to network failure (disconnection of the network of live nodes). As an alternative, we propose an algorithm to determine a minimum-distance spanning tree (MST) based data gathering tree that is more energy-efficient and prolongs the node and network lifetimes, at the cost frequent tree reconfigurations.

Keywords: Stability, Node Lifetime, Network Lifetime, Tradeoff, Sensor Networks, Data Gathering Algorithms, Link Expiration Time, Spanning Trees, Simulations

1. Introduction

A wireless sensor network comprises of several smart sensor nodes that can gather data about the surrounding environment as well as process them before propagating to a control center called the sink, from which the end user typically operates to administer the network and access the nodes. Wireless sensor networks have been considered to give unprecedented levels of access to real-time information about the physical world, and the benefits of deploying such networks are widely seen these days. However, in almost all cases, the sensor networks are statically deployed and evaluated, wherein the mobility of the sensor nodes, the users and the monitored phenomenon are all totally ignored. Wireless mobile sensor networks (WMSNs) are the next logical evolutionary step for sensor networks in which mobility needs to be handled in all its forms. A motivating example could be a network of environmental monitoring sensors, mounted on vehicles, used to monitor pollution levels in a city. In this example, all the entities involved (i.e. the sensors, the users, and the sensed phenomenon as well) are moving. Likewise, one can conceptualize many such real-time scenarios to deploy sensor networks in which one or more of the participating entities move.

Like their static counterparts, the mobile sensor nodes are likely to be constrained with limited battery charge, memory and processing capability as well as operate under a limited transmission range. Two sensor nodes that are outside the transmission range of each other cannot communicate directly. The bandwidth of a WMSN is also expected to be as constrained as that of a static sensor network. Due to all of the above resource and operating constraints, it will not be a viable solution to require every sensor node to directly transmit their data to the sink over a longer distance. Also, if several signals are transmitted at the same time over a longer distance, it could lead to lot of interference and collisions. Thus, there is a need for employing energy-efficient data gathering algorithms that can effectively combine the data collected at these sensor nodes and send only the aggregated data (that is a representative of the entire network) to the sink.

Tree-based data gathering is considered to be the most energy-efficient [23] in terms of the number of link transmissions; however, almost all of the tree-based data gathering algorithms have been proposed for static sensor networks without taking the mobility of the sensor nodes into consideration. In the presence of node mobility, the network topology changes dynamically with time – leading to frequent tree reconfigurations. Thus, mobility brings in an extra dimension of constraint to a WMSN and we need algorithms that can determine stable long-living data gathering trees that do not require frequent reconfigurations. To the best of our knowledge, we have not come across any work on stable data gathering trees for mobile sensor networks. The only tree-based data gathering algorithm we have come across for WMSNs is a shortest path-based spanning tree algorithm [9] wherein each sensor node is constrained to have at most a certain number of child nodes. Based on the results from the literature of mobile ad hoc networks (e.g., [10][11]), minimum hop shortest paths and trees in mobile network topologies are quite unstable and need to be frequently reconfigured. We could not find any other related work on tree-based data gathering for WMSNs.

Most of the work on data gathering algorithms for WMSNs is focused around the use of clusters wherein researchers have tried to extend the classical LEACH (Low Energy Adaptive Clustering Hierarchy) [3] algorithm for dynamically changing network topologies. Variants of LEACH for WMSNs that have been proposed in the literature include those that take into consideration the available energy level [12] and the mobility-level [2] of the nodes to decide on the choice of cluster heads; stability of the links between a regular node and its cluster head [4]; as well as set up a panel of cluster heads to facilitate cluster reconfiguration in the presence of node mobility [1]. Another category of research in WMSNs is to employ a mobile data collecting agent (e.g., [5][6][8]) that goes around the network in the shortest possible path towards the location from which the desired data is perceived to originate.

In this research, we propose two distributed spanning tree-based data gathering algorithms for WMSNs. One of these data gathering algorithms is based on the notion of link expiration time (LET) that is predicted according to a model used for the highly successful Flow-Oriented Routing Protocol (FORP) [13], a stable unicast routing protocol for mobile ad hoc networks. The LET-DG tree is a rooted directed spanning tree determined in a distributed fashion on a network graph comprising of links whose weights are the predicted expiration time. The LET-DG tree has been observed to yield long-living stable trees that exist for a longer time. As observed in the simulation studies of this paper, the drawback of using stable trees is that they tend to overuse certain nodes (especially the intermediate nodes of the data gathering tree) and lead to their premature failure. As sensor networks are often deployed with higher density, one or more node failures do not immediately bring the network to a halt. The live sensor nodes (the nodes that still have a positive available energy) maintain the coverage and connectivity of the underlying network for a longer time. Nevertheless, the unfairness of node usage persists with stable data gathering trees. As an alternative, we propose a second data gathering algorithm that is based on a distributed implementation of the minimum-distance spanning tree (MST) algorithm run on a network graph comprising of links whose weights are the Euclidean distance between the constituent end nodes. The MST-DG trees have been observed to yield a much longer node and network lifetimes, at the cost of frequent tree reconfigurations.

The rest of the paper is organized as follows: Section 2 presents the system model, including the models for the link expiration time and energy consumption, as well as states the assumptions. Section 3 describes the proposed algorithm to determine the LET-DG trees in a distributed fashion. Section 4 presents a variation of the LET-DG algorithm to determine minimum-distance based MST-DG trees. Section 5 presents an exhaustive simulation-based comparison of the LET-DG and MST-DG trees with respect to performance metrics such as the tree lifetime and the node and network lifetimes (due to disconnection) along with a distribution of the probability of node failures. Section 6 concludes the paper. Note that most of the performance comparison studies in the sensor network literature stop their simulations with the first node failure. In this paper, we continue beyond the first node failure and keep track of the time and distribution of the subsequent node failures. Throughout the paper, the terms ‘data aggregation’ and ‘data gathering’, ‘edge’ and ‘link’ are used interchangeably. They mean the same.

2. System Model, Energy Consumption Model and Assumptions

The *system model* adopted for the data gathering algorithms presented in this paper can be summarized as follows:

(i) The underlying network graph considered in the construction of the communication topology used for data gathering is a unit disk graph [20] constructed assuming each sensor node has a fixed transmission range, *R*. There exists a link between any two nodes in a unit disk graph if and only if the physical distance between the two end nodes of the link is ≤ the transmission range, *R*.

(ii) The data gathering algorithms operate in several rounds, and during each round, data from the sensor nodes are collected, aggregated and forwarded to the sink through the data gathering tree (LET-DG or MST-DG tree) rooted at a leader node.

(iii) The leader node of a data gathering tree remains the same as long as the tree exists and is randomly chosen by the sink every time a new tree needs to be determined.

(iv) *LET-DG Tree:* The predicted link expiration time (LET) of a link $i - j$ between two nodes i and j, currently at (X_i, Y_i) and (X_j, Y_j), and moving with velocities v_i and v_j in directions θ_i and θ_j (with respect to the positive X-axis) is computed using the formula proposed in [3]:

$$LET(i, j) = \frac{-(ab+cd)+\sqrt{(a^2+c^2)R^2-(ad-bc)^2}}{a^2+c^2} \quad ...(1)$$

where $a = v_i*\cos\theta_i - v_j*\cos\theta_j$; $b = X_i - X_j$; $c = v_i*\sin\theta_i - v_j*\sin\theta_j$; $d = Y_i - Y_j$

(v) *MST-DG* ***Tree***: The Euclidean distance for a link $i - j$ between two nodes i and j, currently at (X_i, Y_i) and (X_j, Y_j) is given by: $\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$ ………….. (2)

The *energy consumption model* used is a first order radio model [15] that has been also used in several of the well-known previous work (e.g., [3][16]) in the literature. According to this model, the energy expended by a radio to run the transmitter or receiver circuitry is E_{elec} = 50 nJ/bit and $\in_{amp}$ = 100 pJ/bit/m^2 for the transmitter amplifier. The radios are turned off when a node wants to avoid receiving unintended transmissions.

(i) The energy lost in transmitting a k-bit message over a distance d is: $E_{TX}(k, d) = E_{elec}* k + \in_{amp} *k* d^2$.

(ii) The energy lost in receiving a k-bit message is given by: $E_{RX}(k) = E_{elec}* k$.

(iii) During a network-wide flooding of a control message (for example, the tree establishment messages as described in Sections 3 and 4), each node is assumed to lose energy corresponding to transmission over the entire transmission range of the node and to receive the message from each of its neighbors. In networks of high density, the sum of the energy lost at a node due to reception of the broadcast message from all of its neighbors is often more than the energy lost due to transmitting the message.

The *key assumptions* behind the two data gathering algorithms are as follows:

(i) A sensor node is able to obtain its current location, velocity and direction of motion (with respect to the positive X-axis) at any point of time and also includes the same as a Location Update Vector (LUV) in the TREE-CONSTRUCT message broadcast to its neighborhood at the time of constructing the data gathering trees (refer Sections 3 and 4). With the inclusion of a LUV in the TREE-CONSTRUCT message, we avoid the need to periodically exchange beacons in the neighborhood.

(ii) For the LET-DG trees, a sensor node maintains a LET-table comprising of the estimates of the LET values to each of its neighbor nodes based on the latest TREE-CONSTRUCT messages received from them. For the MST-DG trees, a sensor node maintains a Distance-table comprising of estimates of the Euclidean distance with the neighbor nodes that sent it the TREE-CONSTRUCT message.

(iii) Sensor nodes are assumed to be both TDMA (Time Division Multiple Access) and CDMA (Code Division Multiple Access)-enabled [14]. Every upstream node broadcasts a time schedule (for data gathering) to its immediate downstream nodes; a downstream node transmits its data to the upstream node according to this schedule. Such a TDMA-based communication between every upstream node and its immediate downstream child nodes can occur in parallel, with each upstream node using a unique CDMA code.

(iv) We assume the size of the aggregated data packet to be the same as the size of the individual data packets sent by the sensor nodes. In other words, aggregation at any node does not result in increase in the size of the data packets transmitted from the sensor nodes towards the sink.

3. Link Expiration Time-based Data Gathering (LET-DG) Algorithm

The LET-DG algorithm is a distributed implementation of the maximum spanning tree algorithm [21] on a weighted network graph with the edge weights modeled as the predicted link expiration time (LET) of the constituent end nodes. The objective of a maximum spanning tree algorithm is to determine a spanning tree such that the sum of the edge weights is the maximum. Our aim is to determine a maximum-LET spanning tree for mobile sensor networks such that the sum of the LETs of the constituent links of the spanning tree is the maximum. The LET-DG tree is a rooted maximum-LET spanning tree with the root being the leader node chosen by the sink (as explained in Section 3.2).

3.1 Initializations of State Information on Data Gathering Tree-Configuration

Each sensor node locally maintains its best known state information regarding the data gathering tree-configuration, containing the following fields: *estimated node weight*, *upstream node id*, *tree level*, *LEADER node id*, and *sequence number*. The *LEADER node id* corresponds to the id of the root node of the data gathering tree. The *sequence number* field is the latest known sequence number for a data gathering tree involving the sensor node. The sequence number of a data gathering tree is set during the tree construction process (as explained in Section 3.2). The *upstream node id* is the id of the immediate parent node for the sensor node in the tree. If a sensor node is the LEADER node (i.e., the root), then its upstream node id is set to NULL. The *estimated node weight* is the best known weight corresponding to the position of the sensor node in the tree. The *tree level* field is a measure of the distance of the sensor node from the root node of the tree. When a new data gathering tree needs to be configured (either initially at network startup or when the last known tree is broken), the values to the fields of the tree-configuration state information are set as follows, indicated in parenthesis next to the field name:

- At the LEADER node: *estimated node weight* (+∞), *upstream node id* (NULL), *tree level* (0), *LEADER node id* (self) and *sequence number* (the latest sequence number informed by the sink node through the TREE-INITIATE message for the new tree to be configured).
- At a regular sensor node (i.e., a non-LEADER node): *estimated node weight* (-∞), *upstream node id* (NULL), *tree level* (+∞), *LEADER node id* (NULL) and *sequence number* (the sequence number of the last known tree if one existed; otherwise, set to -1).

In the simulations, the Positive Infinity (+∞) and Negative Infinity (-∞) will be represented respectively as very large positive and very small negative values that fall outside the range of the possible values for the link weight.

3.2 Sink – Selection of the Leader Node

Whenever a sink node fails to receive aggregated data from the leader node of the LET-DG tree, the sink randomly chooses a new leader node from the *list of available nodes* currently perceived to exist with a positive residual energy, and sends it a TREE-INITIATE message to start constructing a tree rooted at the chosen leader node (LEADER). The sink includes a sequence number (a monotonically increasing value maintained at the sink, starting from 0) for the tree construction process in the TREE-INITIATE message, and the leader node includes it in its tree construction message (see Section 3.3) to avoid replay errors involving outdated links. If the leader node is alive (i.e., it has positive available energy), then it responds back with a TREE-INITIATE-ACK message acknowledging that it will start the flooding-based tree discovery. If the TREE-INITIATE-ACK message is not received within a certain time, the sink considers the chosen sensor node to be not alive, removes it from the *list of available nodes*, and sends the TREE-INITIATE message (with a higher sequence number, to avoid any parallel tree construction occurring in the network) to another randomly chosen sensor node from the *list of available nodes*. The above procedure is repeated until the sink successfully finds a leader node that accepts to initiate the tree construction process.

3.3 Initiation of the TREE-CONSTRUCT Message

The leader node broadcasts a TREE-CONSTRUCT message containing a 6-element Tree-Configuration tuple <*sequence number*, *LEADER node id*, *sender node id*, *tree level*, sender's *estimated weight*, *upstream node id*> as well as a location update vector (LUV) comprising of the 4-element tuple <*X-coordinate*, *Y-coordinate*, *Velocity*, *Direction of motion - Angle with respect to the positive X-axis*> to its neighbor nodes. The sequence number is the value sent by the sink to the leader node for the specific tree construction process. If the sender node is the LEADER, it sets the *upstream node id* to its own id; while the other nodes set the *upstream node id* to be the id of the node that they perceive to be their best choice for the upstream node that can connect them to the tree. In the TREE-CONSTRUCT message, the leader node sets the sender's *estimated weight* value to +∞ and the value of the *tree level* field to 0.

3.4 Propagation of the TREE-CONSTRUCT Message and Tree Establishment

When a node receives the TREE-CONSTRUCT message with a higher sequence number for the first time, it treats it as a sign of tree reconfiguration and resets, if it has not already done so, the different fields of the local tree-configuration state information to their initial values as listed in Section 3.2. The receiving node then calculates the weight of the link to the neighbor node from which the message was received. A TREE-CONSTRUCT message is accepted at a node for a weight/tree configuration update and rebroadcast (in the neighborhood of the node) if the following conditions are met:

(i) The upstream node id is not equal to the id of the node itself.
(ii) The value of the *tree level* field in the message is lower than or equal to the current *tree level* field value at the node.
(iii) The estimated weight at the node is ***lower than*** the sender's estimated weight.
(iv) The estimated weight at the node is ***lower than*** the predicted expiration time of the link (LET, calculated according to equation 1) on which the TREE-CONSTRUCT message was received.

If all the above conditions are true, then a node receiving the TREE-CONSTRUCT message accepts the message to update its position in the tree. Note that conditions (i) and (ii) are included to ensure there is no looping. The receiver node selects the sender node as its upstream node for joining/connecting to the tree, sets its estimated weight in the tree as the ***minimum of the sender node's estimated weight for the tree and LET of the link*** through which the TREE-CONSTRUCT message was received, and also sets the value of its *tree level* local state information to one more than the value of the *tree level* field in the TREE-CONSTRUCT message. If its weight is updated, the receiver node sends a TREE-JOIN-CHILD message to the upstream sender node indicating the decision to connect to the tree by becoming its child node. The receiver node also decides to further broadcast the TREE-CONSTRUCT message to its neighbors by replacing the LUV of the sender node with its own LUV, the *sender node id* with its own id, the *sender's estimated weight* with its recently updated weight in the tree, the *upstream node id* set to the id of the node through which it has decided to join/connect to the tree, and the *tree level* value in the message incremented by one (matching to the updated value of the *tree level* local state information at the node). The LEADER node id and the sequence number fields are retained as it is in the TREE-CONSTRUCT message.

A node follows the same procedure as explained above when it receives a TREE-CONSTRUCT message with the highest known sequence number from any other neighbor node. In other words, a TREE-CONSTRUCT message corresponding to the latest broadcast process (decided using the sequence number) is accepted for an update and re-broadcast only if it can ***increase*** the estimated weight of the node to connect to the tree without introducing any looping. The algorithm executes as the TREE-CONSTRUCT message propagates around the sensor network reaching every sensor node. As part of this flooding process, each sensor node is guaranteed to accept the TREE-CONSTRUCT message for a weight/tree-configuration update at least once and broadcast the message in its neighborhood. This is because, the initial estimated weight of a sensor node to join the tree is –∞, and the leader node starts with a positive ∞ value and the LET values for the links are always positive. The objective of the LET-DG algorithm is to connect each node with the largest possible weight value in the tree – a measure of the estimated lifetime of the tree.

3.5 Propagation of the TREE-LINK-FAILURE Message

When an upstream sensor node finds out that a link to one of its downstream child nodes is broken due to failure to receive aggregated data packets, the upstream node initiates a TREE-LINK-FAILURE message and includes in it the sequence number that was used in the TREE-CONSTRUCT message corresponding to the most recently used flooding process. The TREE-LINK-FAILURE message is essentially reverse broadcast along the edges of the sub tree proceeding towards the leader node, starting from the upstream node of the broken link. Similarly, the downstream node detects the link failure when it fails to receive a TDMA-schedule from its upstream node for the next round of data aggregation and initiates a TREE-LINK-FAILURE message to inform about the tree failure to the nodes in the sub tree rooted at it. If an intermediate node and/or leaf node does not receive the TREE-LINK-FAILURE message, it continues to wait for the aggregated data packets from its perceived downstream nodes or the TDMA-schedule from its upstream node until it learns about the tree failure through the broadcast of a new TREE-CONSTRUCT message with a sequence number greater than that of the most recently used tree.

4. Minimum-distance Spanning Tree-based Data Gathering (MST-DG) Algorithm

The MST-DG algorithm is a distributed implementation of the minimum spanning tree algorithm [21] on a weighted network graph with the edge weights modeled as the Euclidean distance between the constituent end nodes. Our aim is to determine a minimum-distance spanning tree for wireless mobile sensor networks, such that the sum of the distances of the constituent links of the spanning tree is the minimum. Since a sensor node loses more energy to transmit over a larger distance, we reduce the transmission energy loss across the whole spanning tree by setting the edge weight to be the Euclidean distance between the constituent end nodes. The MST-DG tree is a rooted minimum-distance spanning tree with the root being the leader node chosen by the sink (as explained in Section 3.2). The overall procedure to construct the MST-DG tree is the same as that of the LET-DG tree, except the differences in the criteria used for selecting the links that form part of the tree. In this section, we only highlight these differences in detail and provide a brief outline of the entire algorithm for the sake of completeness.

The constituent fields of the MST data gathering tree-configuration state information are the same as those mentioned in Section 3.1. The initial values for the *upstream node id*, *sequence number*, *LEADER node id* and the *tree level* fields at the LEADER node and the regular sensor nodes are the same as those listed for these nodes in Section 3.1. To begin the process of constructing the MST-DG tree, the sink node randomly chooses a leader node (as is done in the case of LET-DG tree, see Section 3.2) and sends it a unique sequence number, greater than the value used for the previous tree. The initial estimates of the weight at a node to connect to the tree are rather different though for the MST-DG tree. The weight estimate at the LEADER node is 0 and the estimate at every other node is $+\infty$. The LEADER node broadcasts a TREE-CONSTRUCT message in its neighborhood; the message contains the same 6-element *Tree-Configuration* tuple and a 4-element *LUV* as indicated in Section 3.3. A node only processes TREE-CONSTRUCT messages that are received with the largest known sequence number so far or higher. The criteria used at a receiving node to accept the message for a weight/ tree configuration update is as detailed below:

(iv) The upstream node id is not equal to the id of the node itself (same as in Section 3.4)

(v) The value of the *tree level* field in the message is lower than or equal to the current *tree level* field value at the node (same as in Section 3.4).

(vi) The estimated weight at the node is ***greater than*** the sender's estimated weight.

(vii) The estimated weight at the node is ***greater than*** the predicted Euclidean distance of the link (calculated according to equation 2) on which the TREE-CONSTRUCT message was received.

If all the above four conditions are met, then a receiving node decides to join or update its connection to the tree by becoming a child node of the node that sent the TREE-CONSTRUCT message, and sends to it the TREE-JOIN-CHILD message. It sets its estimated weight in the tree to be the ***maximum of the sender's estimated weight and the Euclidean distance*** of the link on which the TREE-CONSTRCT message was received and sets its *tree level* value to one more than the value in the TREE-CONSTRUCT message. The receiving node also rebroadcasts the TREE-CONSTRUCT message in its neighborhood by replacing the LUV of the sender node with its own LUV, the *tree-level* value to one more than the current value in the message, the *sender node id* with its own id, the *estimated sender's weight* with its own recently updated weight in the tree, and the *upstream node id* with the id of the node to which it sent the TREE-JOIN-CHILD message. Any subsequently received TREE-CONSTRUCT message is accepted at a node for an update and rebroadcast only if it can ***decrease*** the estimated weight of the node in the tree. The rest of the procedure in the propagation of the TREE-CONSTRUCT message is the same as that explained in Section 3.4.

Note that every sensor node is expected to update its estimated weight in the tree at least once because the initial estimated weight at a sensor node is $+\infty$, and the values for the sender's weight in the TREE-CONSTRUCT message broadcast by the LEADER node is 0, and the Euclidean distance values are always greater than 0. The objective of the MST-DG algorithm is to connect each node with the lowest possible weight value in the tree – a measure of the energy consumption and fairness of node usage. The procedure to detect a link failure and propagate the TREE-LINK-FAILURE messages initiated by the upstream and downstream nodes of the broken link is the same as explained in Section 3.5.

5. Simulations

In this section, we present the results from simulation studies evaluating the performance of the LET-DG and MST-DG data gathering trees under diverse conditions of network density and mobility. The simulations were conducted in ns-2 (version 2.31) [18]. The Medium Access Control (MAC) layer model is the IEEE 802.11 [19] model. The network dimension is 100m x 100m. The number of nodes in the network is 100 and the nodes are uniform-randomly distributed throughout the network. The sink is located at (50, 50), the center of the network field. The transmission range per sensor node is varied from 20m to 50m, in increments of 5m. For brevity, we present only results obtained for transmission ranges of 25m, 30m (representative of moderate density, with connectivity of 96% and above) and 40 m (high density, with 100% connectivity).

Simulations are conducted for two kinds of energy scenarios: One scenario wherein each node is supplied with abundant supply of energy (50 J per node) and there are no node failures due to exhaustion of battery charge; the simulations in these ***sufficient energy scenarios*** are conducted for 1000 seconds. The second scenario is an ***energy-constrained scenario*** in which each node is supplied with a limited initial energy (2 J per node) and the simulations are conducted until the network of live sensor nodes gets disconnected due to the failures of one or more nodes. We conduct constant-bit rate data gathering at the rate of 4 rounds per second (one round for every 0.25 seconds). The size of the data packet is 2000 bits; the size of the control messages used in the tree formation phase is assumed to be 400 bits, which is sufficiently large enough to accommodate the 6-element Tree-Configuration tuple and the 4-element Location Update Vector (LUV) tuple of the TREE-CONSTRUCT message, allocated as follows:

- Tree-Configuration tuple: *sequence number* (int, 2 bytes); *LEADER node id* (int, 2 bytes); *sender node id* (int, 2 bytes); *tree level* (int, 2 bytes); *sender's estimated weight* (double, 8 bytes); *upstream node id* (int, 2 bytes)
- LUV tuple: *X-coordinate* (double, 8 bytes); *Y-coordinate* (double, 8 bytes); *Velocity* (double, 8 bytes); *Direction of motion – Angle with respect to the positive X-axis* (double, 8 bytes)

The node mobility model used is the well-known Random Waypoint mobility model [17] with the maximum node velocity being 3 m/s (for low mobility scenarios) and 10 m/s (for high mobility scenarios). According to this model, each node chooses a random target location to move with a velocity uniform-randomly chosen from [0,…, v_{max}], and after moving to the chosen destination location, the node continues to move by randomly choosing another new location and a new velocity. Each node continues to move like this, independent of the other nodes and also independent of its mobility history, until the end of the simulation. For a given value of v_{max}, we also vary the dynamicity of the network by conducting the simulations with a variable number of static nodes (out of the 100 nodes) in the network. The values for the number of static nodes used are: 0 (all nodes are mobile), 20, 50 and 80.

5.1 Performance Metrics

We generated 200 mobility profiles of the network for a total duration of 6000 seconds, for every combination of the maximum node velocity and the number of static nodes. Every data point in the results presented in Figures 1 through 6 is averaged over these 200 mobility profiles. While the tree lifetime is measured for both the sufficient energy and energy-constrained (appropriately prefixed as '*EC*' next to the algorithm names) scenarios, the node and network lifetimes are measured only for the energy-constrained scenarios.

The performance metrics measured in the simulations are:

(i) ***Tree Lifetime*** – the duration for which a data gathering tree existed, averaged over the entire simulation time period.

(ii) ***Node Lifetime*** – measured as the time of first node failure due to exhaustion of battery charge.

(iii) ***Network Lifetime*** – measured as the time of disconnection of the network of live sensor nodes (i.e., the sensor nodes that have positive available battery charge).

To obtain the distribution of node failure times, we counted the frequency of the number of node failures, ranging from 1 to 100, in each of the 200 mobility profile files for every combination of transmission range, maximum node velocity and number of static nodes. The probability for '*x*' number of node failures (*x* from ranging from 1 to 100 as we have a total of 100 nodes in our network for all the simulations) for a given combination of the operating conditions is measured as the number of mobility profile files that reported *x* number of node failures divided by 200, which is the total number of mobility profiles used for every combination of maximum node velocity and number of static nodes. Similarly, we keep track of the time at which '*x*' (*x* ranging from 1 to 100) number of node failures occurred in each of the 200 mobility profiles for a given combination of operating conditions and the values for the time of node failures reported in Figures 5 and 6 is an average of these data collected over all the mobility profile files. We discuss the results for distribution of the time and probability of node failures along with the node and network failure times in Section 5.3.

5.2 Tree Lifetime

We measure the tree lifetime for both the sufficient energy scenarios (to capture the impact of network dynamicity – i.e., variations in node velocity and the number of static nodes) and the energy-constrained scenarios (to capture the impact of tree reconfigurations induced by node failures, in addition to network dynamicity). We say a tree exists topologically if the physical Euclidean distance between the end nodes of the links constituting the tree is within the transmission range of the nodes. In the energy-constrained scenarios, even though a data gathering tree may topologically exist, the tree would require reconfiguration (i.e., a new discovery through network-wide flooding of the TREE-CONSTRUCT messages) if one or more nodes in the tree fail due to exhaustion of battery charge. This has an impact on the lifetime of the data gathering trees observed in the simulations, and this is what we capture by measuring the tree lifetime under energy-constrained scenarios. Since a tree also needs to be reconfigured due to node mobility, the lifetime of the data gathering trees observed for energy-constrained scenarios is always less than or equal to that observed for sufficient energy scenarios. This statement holds true for both the LET-DG and MST-DG trees.

As the LET-DG trees are inherently more topologically stable than the MST-DG trees, we observe the difference in the absolute magnitudes of the tree lifetimes for the sufficient energy and energy-constrained scenarios to be relatively larger in the case of the LET-DG trees, especially for moderate transmission range per node. However, at larger transmission ranges per node, the LET-DG trees sustain premature node failures due to continued use of certain intermediate nodes for stable data gathering; as a result, the MST-DG trees – with their tendency to more fairly use the nodes – show a relatively larger difference in the tree lifetime at sufficient energy scenarios compared to energy-constrained scenarios.

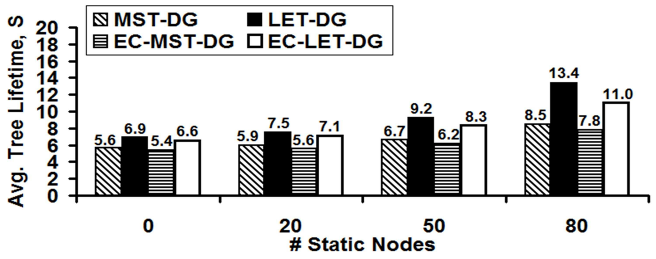

Figure 1(a). Transmission Range = 25 m

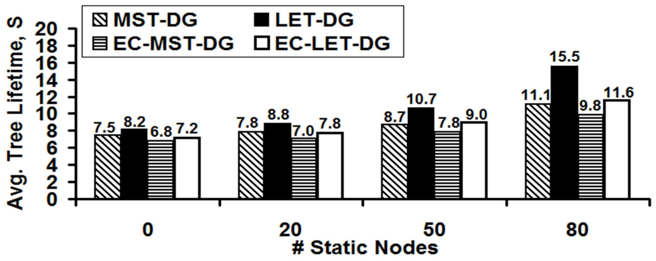

Figure 1(b). Transmission Range = 30 m

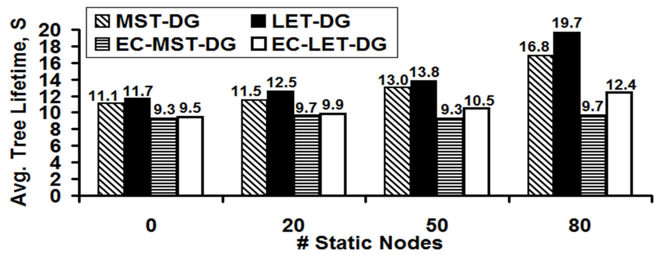

Figure 1(c). Transmission Range = 40 m

Figure 1. Average Tree Lifetime
(Low Node Mobility, v_{max} = 3 m/s)

For fixed node mobility, the magnitude difference in the tree lifetime between the sufficient energy and energy-constrained scenarios for both the data gathering trees increases with increase in the transmission range per node. This can be attributed to the increased energy expenditure incurred at the nodes at larger transmission ranges, leading to certain premature node failures, as well as to a relatively longer link lifetime (i.e., the trees tend to topologically exist for a longer time at larger transmission ranges). At larger transmission ranges, the constituent end nodes of a link have more degrees of freedom to move around and still be within the transmission range of each other for a longer time. On the other hand, for fixed transmission range per node, the difference in the tree lifetime between the sufficient and energy-constrained scenarios decreases with increase in v_{max}. This is as expected because the trees have a relatively lower topological lifetime at high node velocities. For a given v_{max} and transmission range per node, the difference in the magnitude for the lifetime of the data gathering trees for the sufficient energy and energy-constrained scenarios increases with increase in the number of static nodes. This can be attributed to increase in the topological lifetime of the trees as the number of static nodes increases.

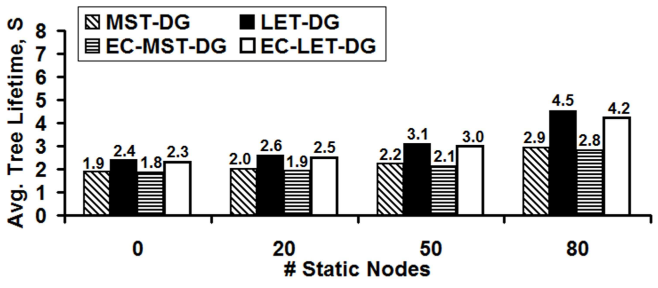

Figure 2(a). Transmission Range = 25 m

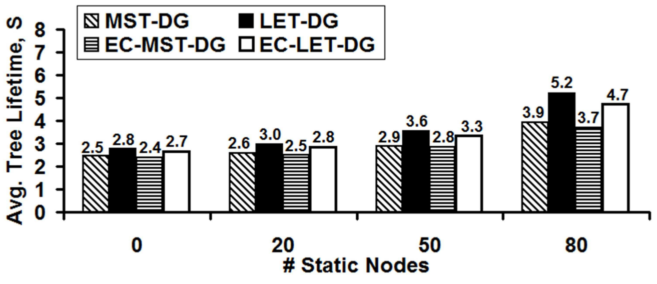

Figure 2(b). Transmission Range = 30 m

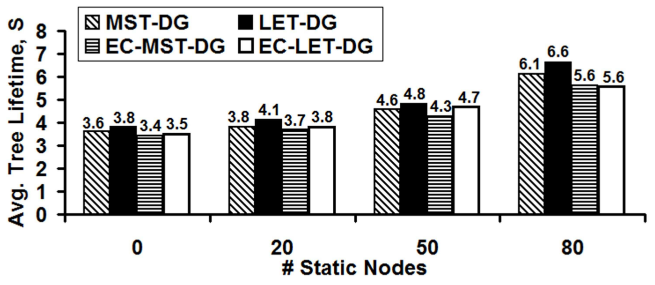

Figure 2(c). Transmission Range = 40 m

Figure 2. Average Tree Lifetime
(Moderate-High Node Mobility, v_{max} = 10 m/s)

While comparing the magnitude difference between the lifetime of the LET-DG and MST-DG trees, we observe that the difference in magnitude decreases with increase in the transmission range per node as well as with increase in network dynamicity (i.e. as more nodes are mobile). For a given maximum node velocity, the LET-DG trees incur a much longer lifetime compared to the MST-DG trees when operated at moderate transmission ranges per node and a larger proportion of static nodes. In the sufficient energy scenarios, when operated under moderate transmission ranges per node, the difference in the tree lifetime can be as large as 25% when all the 100 nodes are mobile and as large as 55-60% when operated with 80 static and 20 mobile nodes. Under energy-constrained scenarios, especially at larger node mobility, as we increase the transmission range per node, the lifetime of the LET-DG trees converge to that of the MST-DG trees. This can be attributed to the premature node failures in the LET-DG trees. For a given maximum node velocity and transmission range per node, as we increase the number of static nodes from 0 to 80 (out of a total of 100 nodes) the LET-DG trees incur about 60-90% larger lifetime and the MST-DG trees incur about 50-60% larger lifetime.

For both the sufficient energy and energy-constrained scenarios, for each data gathering tree, for a fixed transmission range per node, as we increase the maximum node velocity by more than 3 times (i.e., from 3 m/s to 10 m/s), we observe a more or less proportional decrease in the tree lifetime (i.e., the lifetime of the trees decreases by about 1/3rd). For a fixed maximum node velocity, as we increase the transmission range per node from 25m to 40m, we observe more than a proportional increase in the lifetime for both data gathering trees. This can be attributed to the significantly high network connectivity (more than a linear increase) obtained at larger transmission ranges per node.

5.3 Node Lifetime and Network Lifetime

We observe a stability-node/network lifetime tradeoff between the LET-DG and MST-DG trees. While the LET-DG trees have been credited for higher stability, they are unfair with respect to node usage. An intermediate node that lies on a stable tree tends to get used for a longer time and ends up spending more energy to receive data from all of its child nodes, aggregate them and transmit to an upstream node; whereas, the leaf node of a data gathering tree only spends energy to transmit its data to the upstream node.

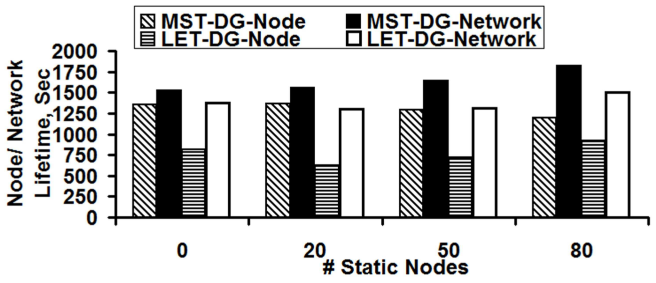

Figure 3(a). Transmission Range = 25 m

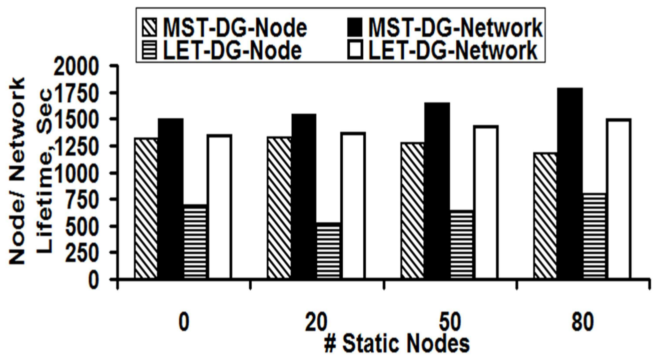

Figure 3(b). Transmission Range = 30 m

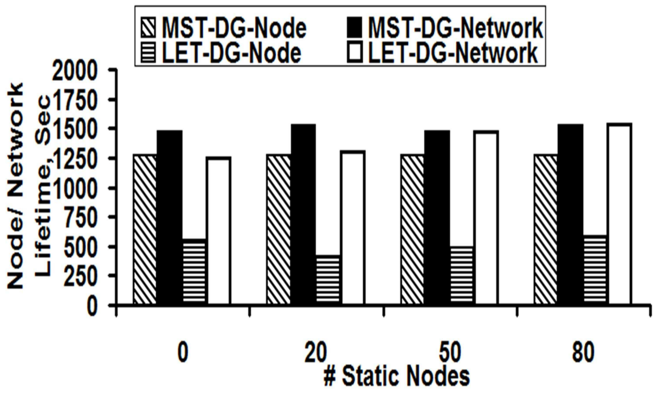

Figure 3(c). Transmission Range = 40 m

Figure 3. Average Node Lifetime and Network Lifetime (Low Node Mobility, v_{max} = 3 m/s)

With a shallow structure and more leaf nodes, the LET-DG trees are vulnerable for premature node failures, as observed in Figures 3 – 6. However, the LET-DG trees have been observed to significantly offset the early node failures with a much better network lifetime, attributed to the lower energy spent to reconfigure the trees and the possibility of the energy-rich leaf nodes (that were lightly used before and during the first few node failures) becoming intermediate nodes in the subsequently reconfigured trees after the initial set of node failures. The impact of the latter factor could be especially observed in Figures 5 and 6 wherein we show the distribution of the node failure times and the probability of node failures.

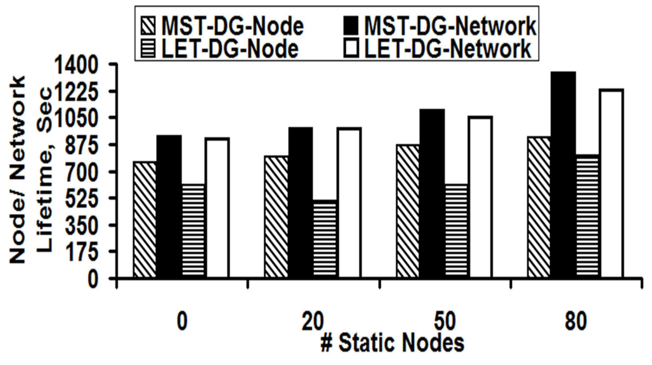

Figure 4(a). Transmission Range = 25 m

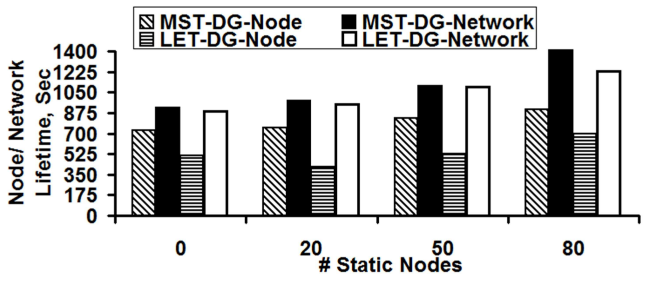

Figure 4(b). Transmission Range = 30 m

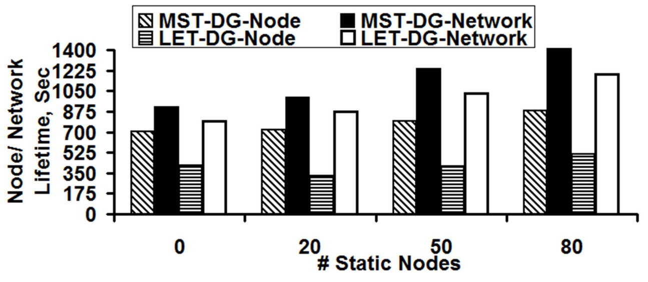

Figure 4(c). Transmission Range = 40 m

Figure 4. Average Node Lifetime and Network Lifetime (Moderate-High Node Mobility, v_{max} = 10 m/s)

After the initial set of node failures, attributed to the excessive use of certain nodes as part of the stable trees, we observe the LET-DG trees to have a much lower probability of node failure for much of the network's lifetime compared to the MST-DG trees. This could be attributed to the relatively equal expenditure of energy across all the nodes of a MST-DG tree. With fewer leaf nodes and relatively more frequent tree reconfigurations, we expect almost all of the nodes in a MST-DG tree to lose about the same amount of energy during the network lifetime. This could be confirmed by observing a much flatter curve for the probability of node failure (closer to 1) for a sufficiently larger number of node failures. From figures 3 and 4, we observe the network lifetime incurred for the MST-DG trees to be mostly about 15-30% more than that of the node lifetime (and at best, 70% larger when operated with 80 static nodes at v_{max} of 10 m/s and transmission range per node of 40m); whereas, the network lifetime for the LET-DG trees to be mostly 50-125% more than that of the node lifetime (and at best, can be as large as 200% more when operated with 80 static nodes at v_{max} of 10 m/s and transmission range per node of 40m).

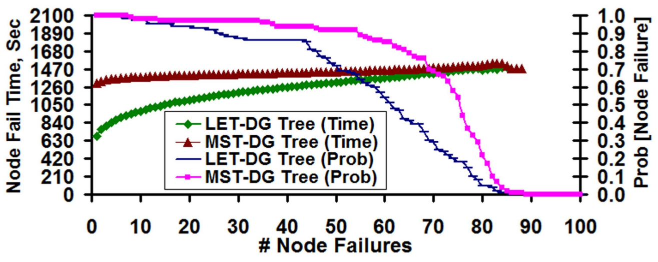

Figure 5(a). Transmission Range = 30 m, 0 Static Nodes

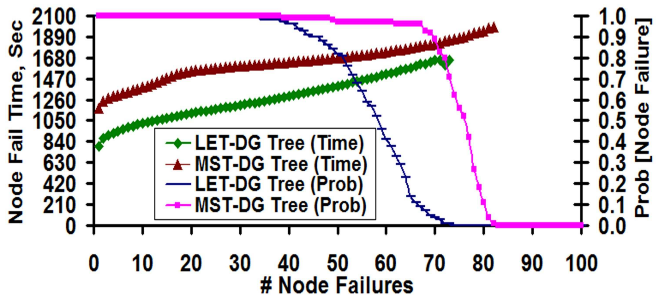

Figure 5(b). Transmission Range = 30 m, 80 Static Nodes

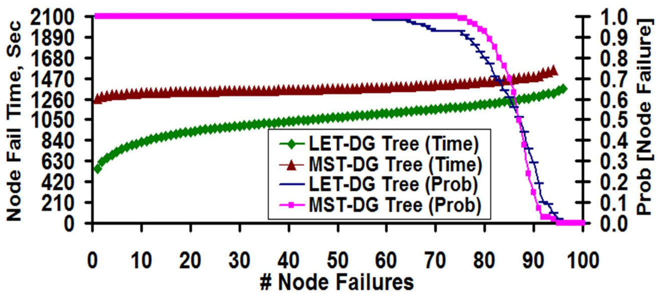

Figure 5(c). Transmission Range = 40 m, 0 Static Nodes

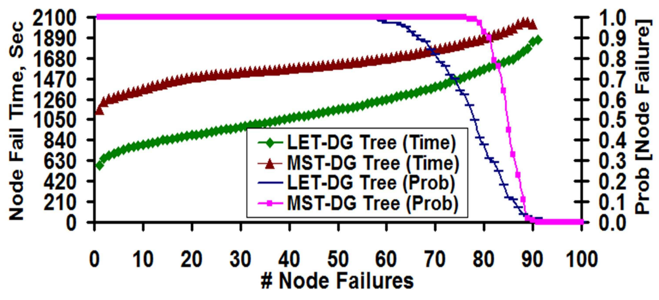

Figure 5(d). Transmission Range = 40 m, 80 Static Nodes

Figure 5. Distribution of Node Failure Times and Probability of Node Failures [v_{max} = 3 m/s]

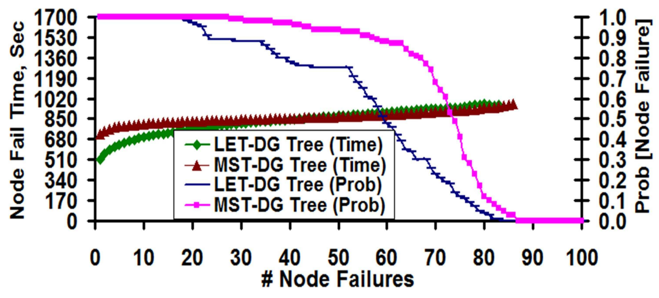

Figure 6(a). Transmission Range = 30 m, 0 Static Nodes

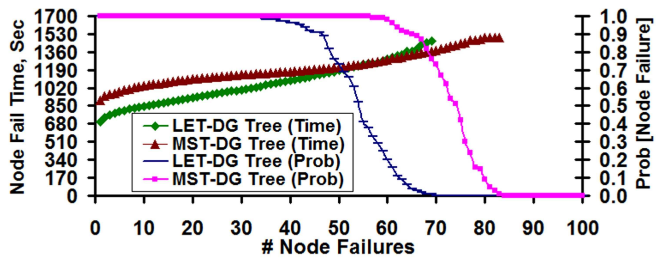

Figure 6(b). Transmission Range = 30 m, 80 Static Nodes

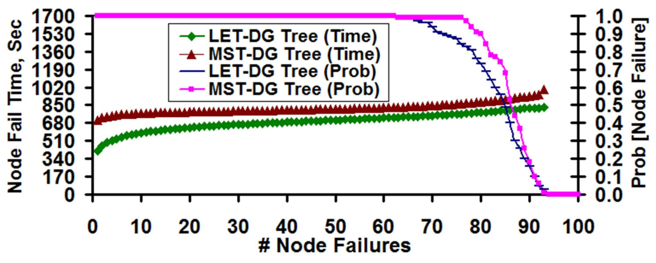

Figure 6(c). Transmission Range = 40 m, 0 Static Nodes

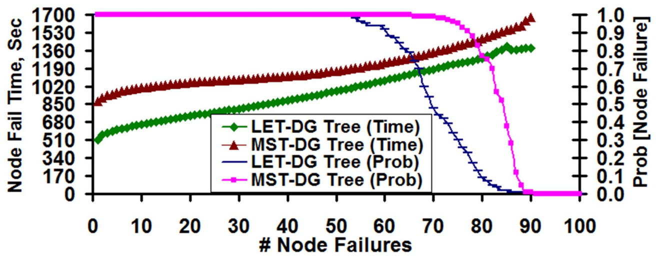

Figure 6(d). Transmission Range = 40 m, 80 Static Nodes

Figure 6. Distribution of Node Failure Times and Probability of Node Failures [v_{max} = 10 m/s]

The impact of the difference in the node usage policies of the two data gathering trees can be observed in the difference in the magnitudes for the node lifetimes (the time of first node failure) and the network lifetimes (the time by which the network of live nodes, nodes with positive available energy, getting disconnected due to failure of peer nodes) for the two data gathering trees. We observe the MST-DG trees to incur about 85-150% larger node lifetime than the LET-DG trees for different combinations of node mobility, # static nodes and transmission ranges per node. On the other hand, the network lifetime sustained for the MST-DG trees is hardly 10-15% more than that of the LET-DG trees and is only at most 25% larger.

In the case of LET-DG trees, one can observe from Figures 3 and 4 that for a given level of network dynamicity (v_{max} and the number of static nodes), the node lifetime substantially decreases (as large as by 25%) with increase in the transmission range per node, whereas the network lifetime decreases only marginally (by at most 10%) with increase in the transmission range per node. The decrease in the node lifetime with increase in transmission range can be attributed to the increase in the transmission energy loss and receipt of data from several downstream nodes when operated at higher transmission range. However, when operated at a higher transmission range, LET-DG trees discover more stable routes as well as balance the distribution of the role of the intermediate nodes and leaf nodes more evenly, resulting in a significant increase in the time of node failures, beyond the first node failure. In the case of MST-DG trees, we observe a very slight decrease in node lifetime (at most 5%) with increase in transmission range per node. For the network lifetime, we observe a decrease of at most 15% at v_{max} = 3 m/s and an increase of at most 15% at v_{max} = 10 m/s. We attribute the better performance of MST-DG trees with respect to network lifetime at higher node velocities and transmission range per node to the increase in the fairness of node usage, and the possibility of the role of "intermediate node" to be rotated among the nodes with regular reconfiguration of the data gathering tree at high node mobility. The energy-efficiency associated with lower Euclidean distance of the links also helps to contain the transmission energy loss and aid in increasing the network lifetime.

Both the LET-DG and MST-DG trees are observed to demonstrate larger node and network lifetimes when operated in networks that have a mix of both static and mobile nodes vis-à-vis a network comprising of only mobile nodes. For a given value of v_{max} and transmission range per node, both the data gathering trees have been observed to sustain about 20-25% larger node lifetime when operated in a network that is a mix of 80 static and 20 mobile nodes compared to operating in a network of 100 mobile nodes. Under similar conditions, the network lifetimes incurred with both the data gathering trees have been observed to be about 50-70% larger. The MST-DG trees, with their vulnerability to break quickly with time in a network replete with mobile nodes, are observed to be the most benefitted with respect to node and network lifetimes when operated in networks that are a mix of static and mobile nodes.

6. Conclusions

We have proposed two distributed algorithms to construct (i) stable predicted link expiration time-based data gathering (LET-DG) trees and (ii) energy-efficient minimum-distance spanning tree based data gathering (MST-DG) trees that incur larger node and network lifetimes. The two algorithms *do not require* any periodic beacon exchange in the neighborhood of the sensor nodes. Performance comparison studies of the two data gathering trees under diverse simulation conditions of network dynamicity (variable node velocity and number of static nodes) and density (variable transmission range per node) illustrate a complex stability vs. node-network lifetime tradeoff that has not been explored so far in the context of data gathering in wireless mobile sensor networks. The LET-DG trees sustain for a longer time; however due to repeated use of certain nodes as part of stable data gathering, the LET-DG trees suffer from pre-mature node failures. Still, the network lifetime observed with the LET-DG trees is only at most 25% less than that observed with the MST-DG trees. The MST-DG trees incur a significantly longer node lifetime (time of first node failure) than that of the LET-DG trees by as large as 85-150%; however, due to frequent tree reconfigurations and larger depth of the tree, almost all the nodes in a MST-DG tree lose about the same amount of energy. As a result, even though the first node failure occurs after a prolonged time, the subsequent node failures occur more quickly, within a short span of time, contributing to only about 15-30% additional network lifetime beyond the time of first node failure.

With respect to the impact of the operating conditions on the different performance metrics, we observe the lifetime of the LET-DG trees to be significantly larger than that of the MST-DG trees at low node mobility and moderate transmission ranges per node and converge to that of the MST-DG trees at higher node mobility and large transmission ranges per node. As we increase the number of static nodes, we observe the lifetime of LET-DG trees to increase at a relatively faster rate. The lifetime of both the LET-DG and MST-DG trees are observed to be lower at energy-constrained scenarios compared to those incurred at the sufficient energy scenarios. For each data gathering tree, the difference in the magnitude of lifetime (in energy-constrained vs. sufficient energy scenarios) increases with increase in transmission range per node (for fixed node mobility) and decreases with increase in node mobility (for a fixed transmission range per node). For a fixed maximum node velocity and transmission range, the node and network lifetimes incurred with both the MST-DG and LET-DG trees substantially increase with increase in the number of static nodes.

References

[1] H. K. D. Sarma, A. Kar and R. Mall, "Energy Efficient and Reliable Routing for Mobile Wireless Sensor Networks," Proceedings of the 6th IEEE International Conference on Distributed Computing in Sensor Systems Workshops, June 2010.

[2] G. Santhosh Kumar, M. V. Vinu Paul and K. Jacob Poulose, "Mobility Metric based LEACH-Mobile Protocol," Proceedings of the 16th International Confernece on Advanced Computing and Communications, pp. 248-253, December 2008.

[3] W. Heinzelman, A. Chandrakasan and H. Balakarishnan, "Energy-Efficient Communication Protocols for Wireless Microsensor Networks," Proceedings of the Hawaaian International Conference on Systems Science, January 2000.

[4] S. Deng, J. Li and L. Shen, "Mobility-based Clustering Protocol for Wireless Sensor Networks with Mobile Nodes," IET Wireless Sensor Systems, vol. 1, no. 1, pp. 39-47, 2011.

[5] M. Zhao and Y. Yang, "Bounded Relay Hop Mobile Data Gathering in Wireless Sensor Networks," Proceedings of the 6th IEEE International Conference on Mobile Ad hoc and Sensor Systems, pp. 373-382, October 2009.

[6] G. Xing, T. Wang, W. Jia and M. Li, "Rendezvous Design Algorithms for Wireless Sensor Networks with a Mobile Base Station," Proceedings of the 9th ACM International Symposium on Mobile Ad hoc Networking and Computing, pp. 231-240, 2008.

[7] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann and F. Silva, "Directed Diffusion for Wireless Sensor Networking," IEEE/ACM Transactions on Networking, vol. 11, no. 1, pp. 2-16, February 2003.

[8] W. Wu, H. Beng Lim and K-L. Tan, "Query-driven Data Collection and Data Forwarding in Intermittently Connected Mobile Sensor Networks," Proceedings of the 7th International Workshop on Data Management for Sensor Networks, pp. 20-25, 2010.

[9] M. Singh, M. Sethi, N. Lal and S. Poonia, "A Tree Based Routing Protocol for Mobile Sensor Networks (MSNs)," International Journal on Computer Science and Engineering, vol. 2, no. 1S, pp. 55-60, 2010.

[10] N. Meghanathan, "Performance Comparison of Minimum Hop and Minimum Edge Based Multicast Routing Under Different Mobility Models for Mobile Ad Hoc Networks," International Journal of Wireless and Mobile Networks, vol. 3, no. 3, pp. 1-14, June 2011.

[11] N. Meghanathan and A. Farago, “On the Stability of Paths, Steiner Trees and Connected Dominating Sets in Mobile Ad Hoc Networks,” Ad Hoc Networks, vol. 6, no. 5, pp. 744 - 769, July 2008.
[12] T. Banerjee, B. Xie, J. H. Jun, and D. P. Agarwal, “LIMOC: Enhancing the Lifetime of a Sensor Network with Mobile Clusterheads,” Proceedings of the Vehicular Technology Conference Fall, pp. 133-137, 2007.
[13] W. Su and M. Gerla, “IPv6 Flow Handoff in Ad hoc Wireless Networks using Mobility Prediction,” Proceedings of the IEEE Global Telecommunications Conference, pp. 271-275, December 1999.
[14] A. J. Viterbi, “CDMA: Principles of Spread Spectrum Communication,” 1st edition, Prentice Hall, April 1995.
[15] T. S. Rappaport, “Wireless Communications: Principles and Practice,” 2nd edition, Prentice Hall, January 2002.
[16] S. Lindsey, C. Raghavendra and K. M. Sivalingam, “Data Gathering Algorithms in Sensor Networks using Energy Metrics,” IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 9, pp. 924-935, September 2002.
[17] C. Bettstetter, H. Hartenstein and X. Perez-Costa, “Stochastic Properties of the Random-Way Point Mobility Model,” Wireless Networks, pp. 555 – 567, vol. 10, no. 5, Sept. 2004.
[18] K. Fall and K. Varadhan, ns-2 Notes and Documentation, The VINT Project at LBL, Xerox PARC, UCB, and USC/ISI, Retrieved from http://www.isi.edu/nsnam/ns/, Last accessed: July 15, 2012.
[19] G. Bianchi, “Performance Analysis of the IEEE 802.11 Distributed Coordinated Function,” IEEE Journal on Selected Areas in Communications, vol. 18, no. 3, pp. 535-547, March 2000.
[20] F. Kuhn, T. Moscibroda and R. Wattenhofer, “Unit Disk Graph Approximation,” Proceedings of the Workshop on the Foundations of Mobile Computing (DIALM-POMC), pp. 17-23, October 2004.
[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms, The MIT Press, 3rd edition, July 2009.
[22] H. Zhang and J. C. Hou, “Maintaining Sensing Coverage and Connectivity in Large Sensor Networks,” Wireless Ad hoc and Sensor Networks: An International Journal, vol. 1, no. 1-2, pp. 89-123, January 2005.
[23] N. Meghanathan, “A Comprehensive Review and Performance Analysis of Data Gathering Algorithms for Wireless Sensor Networks,” International Journal of Interdisciplinary Telecommunications and Networking (IJITN), vol. 4, no. 2, pp. 1-29, April-June 2012.