Macrothink Institute™

Network Protocols and Algorithms
ISSN 1943-3581
2013, Vol. 5, No. 4

Centralized and Distributed Algorithms for Stability-based Data Gathering in Mobile Sensor Networks

Natarajan Meghanathan

Department of Computer Science, Jackson State University

1400 John R. Lynch Street, Jackson, MS 39217, USA

Tel: 1-601-979-3661 E-mail: natarajan.meghanathan@jsums.edu

Philip Mumford

Air Force Research Lab, Wright-Patterson Air Force Base, AFRL/RYWA

2241 Avionics Circle, Dayton, OH 45433, USA

Tel: 1-937-528-8553 E-mail: Philip.Mumford@wpafb.af.mil

Received: September 1, 2013 Accepted: November 15, 2013 Published: December 31, 2013

DOI: 10.5296/npa.v5i4.4208 URL: http://dx.doi.org/10.5296/ npa.v5i4.4208

Abstract

Due to the dynamic nature of the network topology in a mobile sensor network, a data gathering tree is likely to frequently break, necessitating the need for stable data gathering trees that can withstand node mobility for a reasonable amount of time. In this pursuit, we propose two algorithms: (1) a centralized algorithm that can return the sequence of longest-living stable data gathering trees such that the number of tree transitions (changes) is the global minimum; (2) a distributed algorithm that is based on the idea of finding a maximum spanning tree on a network graph whose edge weights are the predicted link expiration times (LET). While the centralized maximum stability-based data gathering (MaxS-DG) algorithm can be used to derive benchmarks for the optimal number of tree transitions (and thence the sequence of longest living stable data gathering trees) over the duration of a data gathering session, the distributed LET-based DG algorithm can be run across the sensor nodes in a network to find stable data gathering trees that have a longer lifetime bounded above by the MaxS-DG trees. In the simulations, we evaluate the tree

lifetime, delay per round, node and network lifetime incurred with the MaxS-DG and LET-DG trees and observe a stability-delay tradeoff.

Keywords: Stability, Data Gathering Tree, Tree Lifetime, Link Expiration Time, Minimum Distance Spanning Trees, Simulations, Mobile Sensor Networks

1. Introduction

A wireless sensor network is a network of several smart sensor nodes that can gather data about the ambient environment as well as intelligently process them before propagating to a control center called the sink, which is typically located far away from the field being monitored and used to remotely administer the sensor network. Even though widely used for data gathering in several real-time applications, wireless sensor networks are mostly deployed for static environments, wherein the mobility of the sensor nodes, the users and the monitored phenomenon are all totally ignored. A wireless mobile sensor network (WMSN) is the next logical evolutionary step for sensor networks in which mobility needs to be handled in all its forms. With the widespread growth of embedded systems and ubiquitous computing technologies, a mobile sensor network could be envisioned as a homogeneous or heterogeneous network of sensor-equipped computers, mobile phones and vehicles, generally referred to as nodes (having one or more sensors like a camera sensor, microphone, GPS sensor, etc) [1]. The nodes of a WMSN often move in an arbitrary fashion, independent of each other. Some of the applications [2] of WMSNs could be traffic monitoring, route planning, civil infrastructure monitoring (say, attaching vibration sensors to cars and monitoring the conditions of roads/pot holes), geo-imaging and etc. WMSNs can be used to monitor and collect data over a much larger geographical area with less number of sensor nodes compared to static sensor networks. With mobility, the entire area could be covered with fewer sensors/nodes over a period of time.

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.

Over the past few years, the sensor network research community has proposed a number of data gathering algorithms to effectively combine the data collected at these sensor nodes through a properly constructed communication topology and send only the aggregated data (that is a representative of the entire network) to the sink. However, a majority of these data

gathering algorithms are meant for static sensor networks (i.e., static sensor nodes) with either a static (e.g., [3][4]) or mobile (e.g., [5][6]) sink. Tree-based data gathering is considered to be the most energy-efficient [7] 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. Our contributions in this paper are two fold:

- First, we propose a centralized algorithm that can be used to find a sequence of longest-living stable data gathering trees for mobile sensor networks such that the number of tree discoveries is the global minimum. We present a simple but powerful polynomial-time greedy algorithm, referred to as the Maximum Stability Data Gathering (MaxS-DG) algorithm, to determine the sequence of longest-living stable data gathering trees. Given the complete knowledge of the future topology changes, the MaxS-DG algorithm operates based on the following greedy principle: Whenever a data gathering tree is required at time instant t, choose the longest-living data gathering tree from t. The above strategy is repeated over the duration of the data gathering session. The sequence of such longest-living data gathering trees is called the Stable-Mobile-DG-Tree. The worst-case run-time complexity of the MaxS-DG tree algorithm is $O(n^2Tlogn)$ and $O(n^3Tlogn)$ when operated under sufficient-energy and energy-constrained scenarios respectively, where n is the number of nodes in the network and T is the total number of rounds of data gathering; $O(n^2logn)$ is the worst-case run-time complexity of the minimum-weight spanning tree algorithm (we use Prim's algorithm [8]) used to determine the underlying spanning trees from which the data gathering trees are derived.
- Second, we propose a distributed spanning tree-data gathering algorithm for mobile sensor networks, based on the notion of link expiration time (LET) [9] that is predicted according to a model used for the highly successful Flow-Oriented Routing Protocol (FORP) [10], a stable unicast routing protocol for mobile ad hoc networks. The LET-data gathering tree (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.

The rest of the paper is organized as follows: Section 2 presents related work on data

gathering in mobile sensor networks. Section 3 reviews solutions proposed in the literature for Mobile Ad hoc Networks (MANETs) to discover stable communication topologies and also highlights the lessons learnt from our performance comparison studies (in earlier research) on these MANET protocols, all of which form the basis of the research conducted for this paper. Section 4 presents the system model, including the models for the link expiration time and energy consumption, as well as defines the various terms used and states the assumptions. Section 5 presents the MaxS-DG algorithm, analyzes its run-time complexity for both sufficient-energy and energy-constrained scenarios, and provides a formal proof of correctness of the algorithm. We also present an example to illustrate the working of the MaxS-DG algorithm. Section 6 describes the proposed algorithm to determine the LET-DG trees in a distributed fashion. Section 7 presents an exhaustive simulation study evaluating the performance of the MaxS-DG and LET-DG trees under diverse conditions of network dynamicity (node mobility and number of static nodes), network density (transmission range) and energy level at the nodes (sufficient-energy and energy-constrained scenarios). The performance metrics evaluated are the tree lifetime, delay per round, node lifetime and network lifetime. We compare the performance of the MaxS-DG trees with that of the LET-DG trees. Section 8 presents the conclusions along with a summary of the simulation results. For the rest of the paper, the terms ‘node’ and ‘vertex’, ‘edge’ and ‘link’, ‘data aggregation’ and ‘data gathering’ will be used interchangeably. They mean the same.

2. Related Work on Data Gathering in Wireless Mobile Sensor Networks

The research on mobile sensor networks started with the deployment of mobile sink nodes on a network of static sensor nodes. A common approach of data gathering in such environments is to employ a mobile data collecting agent (e.g., [11][12][13]) that goes around the network in the shortest possible path towards the location from which the desired data is perceived to originate. The mobile-agent based algorithms do not gather data from all the sensor nodes in a round; data is gathered only from the nodes in the vicinity of the anchor-points (the stops made by a mobile agent at any particular data gathering round). The focus of these algorithms is thus to choose the anchor-points in such a way that data can be gathered from all the sensor nodes within a certain time threshold and simultaneously maximize the network lifetime and utility (usefulness of the information gathered) as much as possible. Due to the work of Kurs et al [14] in the area of wireless energy transfer, researchers have recently started to envision "rechargeable sensor networks" [15] and have proposed models for joint wireless energy replenishment [16] and anchor-point based mobile data gathering [15]. However, the sensor nodes in such rechargeable sensor networks are still considered static. Our proposed LET-DG distributed data gathering algorithm can be very well adapted to a rechargeable network of mobile sensor nodes.

Very few topology-based data gathering algorithms have been proposed for mobile sensor networks where the sensor nodes actually move. Among these, 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 [17] and the mobility-level [18] of the nodes to decide on the choice of cluster heads; stability of the links between a regular node and its cluster head [19]; as well as set up a panel of cluster heads to facilitate cluster reconfiguration in the presence of node mobility [20]. In another related work [21], the authors assume the network to comprise of a mix of static and mobile nodes: A cluster is evolved within the neighborhood of every static node, with the static node as the cluster head; a mobile node chooses the closest static node as its cluster head. The static nodes directly forward the aggregated data to a leader node that does one more level of aggregation before forwarding the data to the sink node. The problem with this approach is that the static sensor nodes need to be provided with surplus of energy compared to the mobile sensor nodes to sustain their lifetime throughout the duration of the network. Also, if sufficient static nodes are not deployed, a mobile sensor node may have to transmit over a longer distance to reach out to the closest static node; in this case, the mobile node will also soon run out of energy. In [22], the authors propose a distributed cluster-head based algorithm in which cluster-heads are elected based on node IDs (0 to C-1, C to 2C-1 …, to operate with C clusters at a time) or node locations (nodes that are closest to certain landmarks with in a WMSN serve as the cluster-heads). In [23], the authors investigate the use of a directed acyclic graph as the underlying communication topology of a sensor network field, modeled according to the theory of thermal fields, to form propagation paths such that the temperature of the nodes on the path increases as data progresses towards the sink, which is considered to be the warmest.

Tree-based data gathering is considered to be the most energy-efficient [7] 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. The only tree-based data gathering algorithm we have come across for WMSNs is a shortest path tree-based algorithm [24] wherein each sensor node is constrained to have at most a certain number of child nodes. Based on the results from the literature (e.g., [25][26][27]) of mobile ad hoc networks (MANETs), 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 wireless mobile sensor networks.

Both MANETs and WMSNs exhibit mobility of nodes in an arbitrary fashion and are constrained by the limited transmission range of the nodes and network bandwidth. However, one should note that there are some subtle differences between MANETs and WMSNs. The network density in WMSNs is much larger than that seen in MANETs and the sensor nodes operate with relatively very little battery charge. MANET protocols typically try to extend the time of first node failure, as every node (could be a PDA, laptop, etc) is considered important. WMSNs typically focus on prolonging the network lifetime (the time at which the network

gets disconnected due to the failure of one or more sensor nodes). The MANET literature has several solutions to determine stable topologies for unicast, multicast and broadcast communication (e.g., [28][29][30]). However, the MANET communication protocols focus on optimizing the paths from a source node to one or more receiver nodes, depending on the nature of the communication. On the other hand, data gathering algorithms for WMSNs focus on collecting data from all of the sensor nodes and forwarding the aggregated data towards a root node. Hence, though both MANETs and WMSNs have dynamically changing communication topologies, it is not possible to directly employ any existing MANET communication algorithms for data gathering in WMSNs.

As a precursor to the proposed LET-DG algorithm, one of the authors of this paper had recently studied the different forms of spanning tree-based topologies (minimum distance-based, predicted LET-based and minimum velocity-based topologies) [31] that could be used for efficient broadcasting in MANETs. All the three broadcast topologies were determined in a centralized fashion with the complete knowledge of topology changes under idealized settings for medium access, bandwidth and energy-availability. The LET-based broadcast topology (LET-BT) was observed to be relatively more stable; while the minimum distance and minimum velocity-based broadcast topologies were observed to respectively yield a higher energy-efficiency and lower end-to-end delay. However, as noted above, MANET solutions cannot be directly extended and/or applied for WMSNs. Moreover, in a graph theoretic context, LET-BT spanning trees are not rooted and not directed; whereas, we need a data gathering tree (a rooted and directed spanning tree) in the context of WMSNs.

In this paper, we have proposed the LET-DG algorithm as a distributed data gathering algorithm for WMSNs and compared its performance with that of the MaxS-DG algorithm, a benchmarking algorithm based on the idea of graph intersections. The lifetime of the LET-DG trees is bounded above by that of the MaxS-DG trees. The protocol development and simulation framework presented in this paper can become a model for developing a benchmarking algorithm to obtain theoretical bounds on a performance metric for a topology-control problem (in this research, our focus is to arrive at the theoretically possible maximum lifetime of the data gathering trees such that the number of tree discoveries is the global minimum) and develop a distributed algorithm for the same problem as well as conduct a comparative performance evaluation of the distributed algorithm vis-a-vis the centralized algorithm under identical operating conditions. The proposed MaxS-DG algorithm can be used to validate the claims on the stability of data gathering trees determined using any distributed algorithm for WMSNs (like the LET-DG algorithm).

3. Review on MANET Solutions for Stable Communication Topologies

The MANET literature has several protocols and algorithms proposed for determining sequence of stable communication topologies for unicast (paths), multicast (trees) and broadcast (connected dominating sets) communication. The unicast solutions focus on determining a sequence of stable paths between two particular nodes (source-destination nodes) using some form of link stability metric (like the number of beacons exchanged in the

past between the end nodes of the link [32], predicted link expiration time [9], rate of change of signal strength for the messages transmitted on the link [33], etc). In an earlier work [34], it was observed that routes determined through unicast routing protocols that model link stability based on future predictions (like the LET [9]) are more stable than those determined through protocols based on the past history (like the protocols in [33] and [32]). The relatively better performance of the LET-based Flow Oriented Routing Protocol (FORP) [10] in a MANET context is a motivation for the authors to use the LET as the underlying link stability metric for the distributed data gathering algorithm proposed in this paper.

The multicast solutions (e.g., [35][36][37]) proposed for MANETs focus on determining a sequence of stable trees or meshes connecting a source node to one or more receiver nodes forming the multicast group. In prior works (e.g., [38][39]), multicast trees (e.g., [36][37]) were observed to be more energy-efficient and bandwidth-efficient; whereas, the multicast meshes (e.g., [35]) are observed to be more stable (robust to link failures) at the cost of energy and bandwidth efficiency. The MANET multicast tree protocols are designed to be either minimum-hop based (e.g., [37]) or minimum-edge based (e.g., [36]). The minimum-hop based protocols connect the source on a minimum hop path to each of the receivers; whereas, the minimum-edge based protocols aim at connecting the source to all of the receivers through as minimum links as possible. The minimum-hop based multicast trees contain more links than the minimum-edge based trees. The general trend observed in the simulation studies (e.g., [27][40]) comparing the performance of the minimum-edge based and minimum-hop based multicast trees is that larger the number of links on the tree, the lower the stability as well as higher the energy consumption. The relatively better performance of minimum-edge based multicast trees also prompted us to choose a spanning tree-based approach rather than a shortest path-tree based approach to determine the data gathering trees for the algorithms proposed in this paper.

The broadcast solutions for MANETs are typically based on connected dominating sets (CDS) - a subset of nodes in the network such that any node in the network is either in the CDS or is a neighbor of a node in the CDS. The idea behind the use of a CDS is that any broadcast message needs to be transmitted only by the nodes in the CDS within their neighborhood in order for all the nodes to receive the message. Obviously, the smaller the CDS Node Size (the number of nodes constituting the CDS and required to transmit in their respective neighborhoods), the larger is the energy-efficiency. However, from prior research (e.g., [41][42][43][44]), we observe a tradeoff between the CDS Node Size and Lifetime: Connected dominating sets with minimum CDS Node Size (e.g., [42]) do not last long and need to be frequently reconfigured; whereas, CDSs that are determined based on node mobility (e.g., [44]) or link stability (e.g., [43]) as the underlying criterion tend to exist for a longer time. The tradeoff could be minimized to a certain extent by preferring to form minimum node size-CDS based on strong neighborhoods (i.e., prefer CDS nodes that are predicted to have stable links with as many neighbors as possible) [41]. This is another observation that motivated us to propose an algorithm to determine a link stability (LET)-based data gathering tree that would be shallow (lower height to minimize the number of intermediate nodes) and at the same time have as many stable links as possible (between

two intermediate nodes as well as between an intermediate node and a leaf node).

4. System Model, Energy Consumption Model, Terminology and Assumptions

4.1 System Model

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 [45] 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 less than or equal to 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 (MaxS-DG tree or LET-DG) 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 u – v between two nodes u and v, currently at (X_u, Y_u) and (X_v, Y_v), and moving at speed s_i and s_j in directions θ_i and θ_j (with respect to the positive X-axis) is computed using the formula proposed in [9]:

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

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

4.2 Energy Consumption Model

The energy consumption model used is a first order radio model [46] that has been also used in several of the well-known previous work (e.g., [3][4]) 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 given by:

$$E_{TX}(k, d) = E_{elec}* k + \in_{amp} *k* d^2 \qquad (1)$$

(ii) The energy lost in receiving a k-bit message is given by:

$$E_{RX}(k) = E_{elec} * k \qquad (3)$$

(iii) During a network-wide flooding of a control message (for example, the tree establishment messages for LET-DG as described in Section 6), 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. Though the MaxS-DG algorithm is centralized in nature, to be fair with LET-DG, we consider the energy lost in changing from one data gathering tree to another as equal to the energy lost in the network-wide flooding of a control message (same as the size used in the LET-DG algorithm) as described above.

4.3 Terminology

We use the notions of static graphs and mobile graphs (adapted from [47]) to capture the sequence of topological changes in the network and determine a stable data gathering tree that spans over several time instants. A *static graph* is a snapshot of the network at any particular time instant and is modeled as a unit disk graph [45] wherein there exists a link between any two nodes if and only if the physical distance between the two end nodes of the link is less than or equal to the transmission range. The weight of an edge on a static graph is the Euclidean distance between the two end nodes of the edge. The Euclidean distance for a link $u - v$ between two nodes u and v, currently at (X_u, Y_u) and (X_v, Y_v) is given by: $\sqrt{(X_u - X_v)^2 + (Y_u - Y_v)^2}$.

A *mobile graph* $G(i, j)$, for $1 \leq i \leq j \leq T$ and T is the total number of rounds of the data gathering session corresponding to the network lifetime, is defined as $G_i \cap G_{i+1} \cap \ldots G_j$, where G_i, G_{i+1}, ..., G_j are the individual static graphs captured at time instants t_i, t_{i+1}, …, t_j corresponding to rounds i, i+1, ..., j. Thus, a mobile graph is a logical graph that captures the presence or absence of edges in the individual static graphs. In this research work, we sample the network topology periodically for every round of data gathering to obtain the sequence of static graphs. The weight of an edge in the mobile graph $G(i, j)$ is the geometric mean of the weights of the edge in the individual static graphs spanning G_i, …, G_j. Since there exist an edge in a mobile graph if and only if the edge exists in the corresponding individual static graphs, the geometric mean of these Euclidean distances would also be within the transmission range of the two end nodes for the entire duration spanned by the mobile graph. Note that at any time, a mobile graph includes only *live sensor nodes*, nodes that have positive available energy.

A *static spanning tree* is a minimum-weight spanning tree determined on a static graph. Since we use the Euclidean distance between the constituent nodes of an edge as the link weight, the minimum-weight spanning tree determined on a static graph will be a minimum-distance spanning tree for which the sum of the edge weights will be the minimum.

A *static data gathering tree* is a rooted form of its corresponding static spanning tree with the root node being the leader node chosen for the round corresponding to the time instant represented by the static spanning tree. A *mobile spanning tree* is a minimum-weight spanning tree determined on a mobile graph whose edge weights are the geometric mean of the corresponding edge weights in the constituent static graphs. A *mobile data gathering tree* is a rooted form of its corresponding mobile spanning tree with the root node being the leader node chosen for the round corresponding to the beginning time instant of the mobile graph. The leader node of a mobile data gathering tree remains the same until the mobile graph gets disconnected due to node mobility or a node failure occurs, whichever happens first.

4.4 Key Assumptions

The *key assumptions* behind the LET-DG algorithm 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 Section 6). 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. A sensor code could similarly maintain a Distance-table comprising of estimates of the Euclidean distance with the neighbor nodes that sent it the TREE-CONSTRUCT message (assumed for use to determine the MaxS-DG trees).

(iii) Sensor nodes are assumed to be both TDMA (Time Division Multiple Access) and CDMA (Code Division Multiple Access)-enabled [48]. 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.

A *key assumption* behind the MaxS-DG algorithm is that the entire sequence of network topology changes is known beforehand at the time of running the MaxS-DG algorithm. This is required to generate the mobile graph spanning several static graphs, each representing snapshots of the network topology at time instants corresponding to successive rounds of data gathering, on which a stable long-living data gathering tree will be determined. The above assumption may not be practical for distributed systems of sensor networks. However, note

that our goal is to develop the MaxS-DG algorithm as a *benchmarking algorithm* that can give us the sequence of long-living data gathering trees (over the duration of the data gathering session) whose *lifetime will be the upper bound for the data gathering trees* obtained using any other algorithm (like the LET-DG algorithm) developed for this problem in the area of mobile sensor networks. The sequence of such stable longest-living data gathering trees determined using the MaxS-DG algorithm will involve the minimum number of discoveries involving network-wide flooding. Thus, the number of data gathering tree discoveries incurred with the MaxS-DG algorithm *will form the lower bound for the number of data gathering tree discoveries* incurred with any other algorithm for mobile sensor networks.

5. Maximum Stability-based Data Gathering (MaxS-DG) Algorithm

The MaxS-DG algorithm is based on a greedy look-ahead principle and the intersection strategy of static graphs. When a mobile data gathering tree is required at a sampling time instant t_i, the strategy is to find a mobile graph $G(i, j) = G_i \cap G_{i+1} \cap \ldots G_j$ such that there exists a spanning tree in $G(i, j)$ and no spanning tree exists in $G(i, j+1) = G_i \cap G_{i+1} \cap \ldots G_j \cap G_{j+1}$. We find such an epoch $t_i, \ldots, t_j$ as follows: Once a mobile graph $G(i, j)$ is constructed with the edges assigned the weights corresponding to the geometric mean of the weights in the constituent static graphs $G_i, G_{i+1}, \ldots, G_j$, we run the Prim's minimum-weight spanning tree algorithm [8] on the mobile graph $G(i, j)$. If $G(i, j)$ is connected, we will be able to find a spanning tree in it. We repeat the above procedure until we reach a mobile graph $G(i, j+1)$ in which no spanning tree exists and there existed a spanning tree in $G(i, j)$. It implies that a spanning tree basically existed in each of the static graphs $G_i, G_{i+1}, ..., G_j$ and we refer to it as the mobile spanning tree for the time instants $t_i, \ldots, t_j$. To obtain the corresponding mobile data gathering tree, we choose an arbitrary root node for this mobile spanning tree and run the Breadth First Search (BFS) algorithm [8] on it starting from the root node. The direction of the edges in the spanning tree and the parent-child relationships are set as we traverse its vertices using BFS. The resulting mobile data gathering tree with the chosen root node (as the leader node) is used for every round of data gathering spanning time instants $t_i, \ldots, t_j$. We then set $i = j+1$ and repeat the above procedure to find a mobile spanning tree and its corresponding mobile data gathering tree that exists for the maximum amount of time since t_{j+1}. A sequence of such maximum lifetime (i.e., longest-living) mobile data gathering trees over the timescale T corresponding to the number of rounds of a data gathering session is referred to as the ***Stable Mobile Data Gathering Tree***. Figure 1 presents the pseudo code of the MaxS-DG algorithm that takes as input the sequence of static graphs spanning the entire duration of the data gathering session.

While operating the algorithm under energy-constrained scenarios, one or more sensor nodes may die due to exhaustion of battery charge even though the underlying spanning tree may topologically exist. For example, if we have determined a data gathering tree spanning across time instants t_i to t_j using the above approach, and we come across a time instant t_k ($i \leq k \leq j$) at which a node in the tree fails, we simply restart the Max.S-DG algorithm starting

from time instant t_k considering only the live sensor nodes (i.e., the sensor nodes that have positive available energy) and determine the longest-living data gathering tree that spans all the live sensor nodes since t_k. The pseudo code of the MaxS-DG algorithm in Figure 1 handles node failures, when run under energy-constrained scenarios, through the ***if*** block segment in statement 8. If all nodes have sufficient-energy and there are no node failures, the algorithm does not execute statement 8.

Input: Sequence of static graphs $G_1, G_2, \ldots G_T$; Total number of rounds of the data gathering session – T

Output: *Stable-Mobile-DG-Tree*

Auxiliary Variables: i, j

Initialization: i =1; j=1; *Stable-Mobile-DG-Tree* = Φ (empty set)

Begin *MaxS-DG Algorithm*

1 **while** ($i \leq T$) **do**

2 Find a mobile graph $G(i, j) = G_i \cap G_{i+1} \cap \ldots \cap G_j$ such that there exists at least one spanning tree in $G(i, j)$ and {no spanning tree exists in $G(i, j+1)$ or $j = T$}

3 *Mobile-Spanning-Tree*(i, j) = **Prim's Algorithm** ($G(i, j)$)

4 *Root*(i, j) = Choose a node randomly in $G(i, j)$

5 *Mobile-DG-Tree*(i, j) = **Breadth First Search** (*Mobile-Spanning-Tree*(i, j), *Root*(i, j))

6 *Stable-Mobile-DG-Tree* = *Stable-Mobile-DG-Tree* U { *Mobile-DG-Tree*(i, j) }

7 **for** each time instant $t_k \in \{t_i, t_{i+1}, \ldots, t_j\}$ **do**

Use the *Mobile-DG-Tree*(i, j) in t_k

8 **if** node failure occurs at t_k **then**

$j = k - 1$

break

end if

end for

9 $i = j + 1$

10 **end while**

11 **return** *Stable-Mobile-DG-Tree*

End *MaxS-DG Algorithm*

Figure 1: Pseudo Code for the Maximum Stability-based Data Gathering Tree Algorithm

5.1 Example to Illustrate the Working of the MaxS-DG Algorithm

We run the MaxS-DG algorithm on the sequence of static graphs $G_1G_2G_3G_4G_5$ (shown in the first part of Figure 2), generated by sampling the network topology for every second. For simplicity and clarity in the representation, we do not use weights for the edges. The reader could assume that the spanning trees determined on the static graphs and mobile graphs at different instances of execution of the MaxS-DG algorithm in Figure 2 are the minimum-weight spanning trees on the corresponding graphs.

In Figure 2, we could find a connected mobile graph spanning G_1, G_2 and G_3; and could not find a connected mobile graph from G_1 through G_4. A spanning tree exists for a graph if and only if the graph is connected. We determine a spanning tree on $G_1 \cap G_2 \cap G_3$ and derive a data gathering tree rooted at an arbitrarily selected node (node 3). This stable data gathering tree is to be used for the rounds corresponding to time instants of the static graphs G_1, G_2 and G_3. Similarly, we continue with the subsequent two static graphs and find a data gathering tree (with an arbitrary root node – node 6) that exists in both G_4 and G_5. Thus, we require a total of two data gathering tree discoveries for the sequence of static graphs $G_1G_2G_3G_4G_5$. There has to be at least a sequence of two spanning trees for the graph sequence $G_1G_2G_3G_4G_5$ as the intersection graph $G_1 \cap G_2 \cap G_3 \cap G_4$ was not connected.

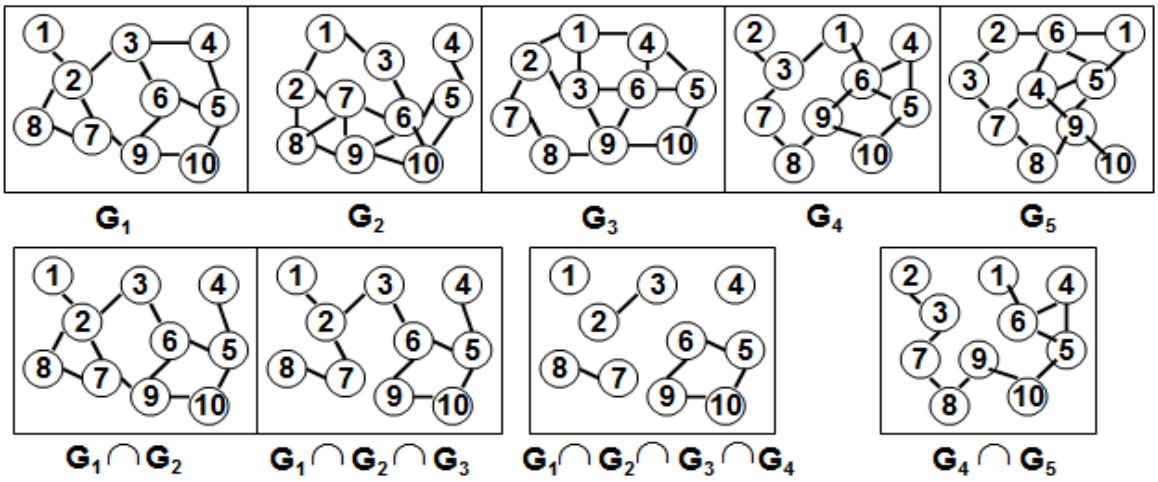


Figure 2: Example to Illustrate Execution of the Maximum Stability-based Data Gathering Algorithm

5.2 Run-time Complexity Analysis of the MaxS-DG Algorithm

To expand a mobile graph $G(i, j) = G_i \cap G_{i+1} \cap \ldots G_j$ to $G(i, j+1)$, all we had to do is to check whether each of the edges in the mobile graph $G(i, j)$ existed at time instant t_{j+1}. This can be done in $O(n^2)$ time on a mobile graph of n nodes, as there can be at most $O(n^2)$ edges on a graph of n vertices. The overall complexity of the MaxS-DG algorithm is the sum of the time complexity to construct the mobile graphs, the time complexity to run the spanning tree algorithm on these mobile graphs and the time complexity to transform these spanning trees to data gathering trees using BFS.

Sufficient-energy Scenarios: When the network operates under sufficient-energy scenarios (i.e., no node failures), for a data gathering session comprising of T rounds, we will have to construct T mobile graphs, resulting in a time complexity of $O(n^2T)$ to construct the mobile graphs. On each of these T mobile graphs, we will have to run a spanning tree algorithm. If we use the $O(n^2*\log n)$ Prim's algorithm to construct a spanning tree, the complexity to run the spanning tree algorithm on the T mobile graphs becomes $O(n^2*\log n*T)$. A spanning tree on n vertices has n–1 edges. The time-complexity of running BFS on a

spanning tree of n vertices with n–1 edges is O(n) [8]. To run BFS on the O(T) spanning trees, we incur O(nT) time. Thus, the overall complexity of the MaxS-DG algorithm under sufficient-energy scenarios is O(n^2T) + O(n^2Tlog$n$) + O(nT) = O(n^2Tlog$n$).

Energy-Constrained Scenarios: There can be at most n–1 node failures (on an n node network) that trigger the execution of statement 8 in the pseudo code of Figure 1 for the MaxS-DG algorithm. A node failure occurring at time instant t_k ($i \leq k \leq j$), while using a mobile data gathering tree that has been determined on a mobile graph for time instants t_i, …, t_j, would require us to construct a mobile graph starting from t_k and the number of mobile graphs that we have to construct and run the spanning tree algorithm increases by j–k+1. At the worst case, if there are n–1 node failures, the number of mobile graphs that we have to construct and run the spanning tree algorithm increases by (T – 1) + (T – 2) + (T – (n–1)) = (n–1)T – [1 + 2 + … + (n–1)] = O(nT) + O(n^2). Under the sufficient-energy scenarios, we had to construct T mobile graphs and run the spanning tree algorithm on each of them. In the energy-constrained scenarios, we will have to construct at most T + O(nT) + O(n^2) mobile graphs and run the spanning tree algorithm on each of them. The number of rounds of data gathering is generally far greater than the number of nodes in the network. For example, in our simulations, we use a value of T = 4000 rounds (4 rounds per second, for 1000 seconds) and n = 100 nodes. Thus, since $n << T$, we can say that $n^2 << nT$. Therefore, a total of T + O(nT) + O(n^2) = T + O(nT) = O(nT) mobile graphs are constructed. The time complexity to construct these mobile graphs is O(n^2 * nT) = O(n^3T). We run the O(n^2log$n$) Prim's spanning tree algorithm on the O(nT) mobile graphs, resulting in a time-complexity of O(n^3Tlog$n$) to determine the spanning trees. The time-complexity of running the O(n)-BFS algorithm on the O(nT) spanning trees is O(n^2T). Thus, the overall time-complexity of the MaxS-DG algorithm under the energy-constrained scenarios is O(n^3T) + O(n^3Tlog$n$) + O(n^2T) = O(n^3Tlog$n$).

5.3 Proof of Correctness of the MaxS-DG Algorithm

In this section, we prove that the MaxS-DG algorithm does determine the sequence of long-living stable mobile data gathering trees such that the number of tree discoveries is the global minimum (i.e. optimum). We use the approach of Proof by Contradiction. Let m be the number of data gathering tree discoveries incurred using the MaxS-DG algorithm on a sequence of static graphs G_1G_2 … G_T. Let there be another algorithm (a hypothetical algorithm) that returns a sequence of mobile data gathering trees for the same sequence of static graphs such that the number of tree discoveries is $n < m$. If such an algorithm exists, then without loss of generality, there has to be one mobile data gathering tree, determined using this hypothetical algorithm, existing for the entire duration of a mobile graph $G(p, s)$; whereas, the MaxS-DG algorithm had to have at least one data gathering tree transition in $G(p, s)$. However, there cannot be such a data gathering tree that spanned through the entire mobile graph $G(p, s)$ and was not discovered by the MaxS-DG algorithm. Because, the MaxS-DG algorithm takes intersection of the static graphs $G_p \cap G_{p+1} \cap \ldots G_s$ and runs a spanning tree algorithm on the intersection graph $G(p, s)$ – if at all a spanning tree existed in $G(p, s)$, then $G(p, s)$ would have to be connected. If the MaxS-DG algorithm could not determine a spanning tree/data gathering tree for the mobile graph $G(p, s)$, it implies the mobile graph $G(p, s)$ is not connected. It is not possible for any algorithm, including our

hypothetical algorithm, to find a spanning tree/data gathering tree that covers all the vertices of a disconnected graph. Thus, the hypothetical algorithm would also had to have at least one tree transition in $G(p, s)$. The above proof holds good for any value of static graph indices p and s, where $1 \leq p \leq s \leq T$, and T is the total number of rounds corresponding to the duration of the data gathering session. Thus, the number of data gathering tree discoveries incurred with using the MaxS-DG algorithm is the global minimum.

Note that in the above proof, we have implicitly assumed that all the sensor nodes are alive for the entire duration of the data gathering session. In other words, we have proved that when operated under sufficient-energy scenarios, the MaxS-DG algorithm returns the stable sequence of data gathering trees such that the number of tree discoveries is the global minimum. It is not possible to theoretically prove the optimality of the MaxS-DG algorithm under energy-constrained scenarios. One can only validate the optimality of the lifetime of the MaxS-DG trees under energy-constrained scenarios through simulations, as we do in Section 7, wherein we observe the MaxS-DG trees to yield a relatively longer lifetime compared to the LET-DG trees under energy-constrained scenarios.

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

The LET-DG algorithm is a distributed implementation of the maximum spanning tree algorithm [8] 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 6.2).

6.1 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 6.2). 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.

6.2 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 6.3) 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.

6.3 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.

6.4 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 6.2. 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 $-\infty$, 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.

6.5 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.

7. Simulations

In this section, we present an exhaustive simulation study on the performance of the MaxS-DG trees and compare them with that of the LET-DG trees under diverse conditions of network density and mobility. The simulations are conducted in a discrete-event simulator developed (in Java) by us exclusively for data gathering in mobile sensor networks. The MAC (medium access control) layer is assumed to be collision-free and considered an ideal channel without any interference. Sensor nodes are assumed to be both TDMA (Time Division Multiple Access) and CDMA (Code Division Multiple Access)-enabled. 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.

The network dimension is 100m x 100m. The number of nodes in the network is 100 and initially, the nodes are uniform-randomly distributed throughout the network. The sink is located at (50, 50), at the center of the network field. For a given simulation run, the transmission range per sensor node is fixed and is the same across all nodes. The network density is varied by varying the transmission range per sensor node from 20m to 50m, in increments of 5m. For brevity, we only present results obtained for transmission ranges per node of 25m and 30m (representative of moderate density, with connectivity of 97% and

above), and for 40m (representative of high density, with 100% connectivity).

Simulations are conducted for two kinds of energy scenarios: One scenario wherein each node is provided with abundant supply of energy (100 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 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. The energy consumption model is as described in Section 4.2.

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 for tree discoveries is assumed to be 400 bits. We assume that a tree discovery (for both MaxS-DG and LET-DG trees) requires network-wide flooding of the 400-bit control messages such that each sensor node will broadcast the message exactly once in its neighborhood. As a result, each sensor node will lose energy to transmit the 400-bit message over its entire transmission range and receive the message from each of its neighbor nodes. In high density networks, the energy lost due to receipt of the redundant copies of the tree discovery control messages dominates the energy lost at a node for tree discovery. All of these mimic the energy loss observed for flooding-based tree discovery in ad hoc and sensor networks.

The node mobility model used is the well-known Random Waypoint mobility model [49] with the maximum node velocity being 3 m/s and 10 m/s representing scenarios of low and high mobility respectively. 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 v_{max} value, 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.

7.1 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 4 through 15 is averaged over these 200 mobility profiles. The tree lifetime and delay per round are measured for both the sufficient-energy and energy-constrained scenarios (appropriately prefixed as '*EC*' next to the names of the data gathering trees). 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) *Delay per Round* – measured in terms of the number of time slots needed per round of data aggregation at the intermediate nodes, all the way to the leader node of the data gathering tree, averaged across all the rounds of the simulation. A brief description of the algorithm used to compute the delay per round is given in Section 7.2 along with an illustration in Figure 3.

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

(iv)*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), while the network would have stayed connected if all the nodes were alive at that time instant. So, before confirming whether an observed time instant is the network lifetime (at which the network of live sensor nodes is noticed to be disconnected), we test for connectivity of the underlying network if all the sensor nodes were alive.

Nodes at Level	Nodes
0	12
1	7, 9
2	15, 2, 5, 14
3	3, 0, 6, 11, 13
4	10, 1, 4
5	8

Delay for the DG Tree
= 1 + Delay for the Root
= 1 + 6 = 7 time units

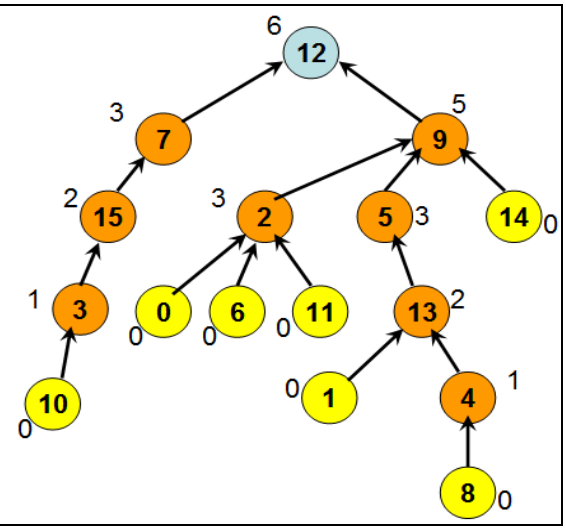


Figure 3: Example to Illustrate the Calculation of Delay per Round of Data Gathering

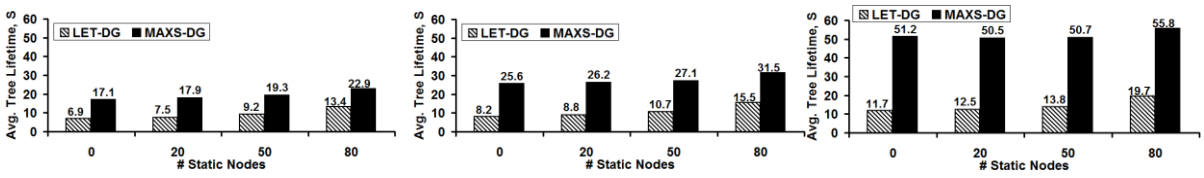
7.2 Algorithm to Compute the Delay per Round of Data Gathering

The delay incurred at a node is measured in terms of the number of time slots it takes to gather data from all of its immediate child nodes. The delay for the data gathering tree is one plus the delay incurred at the leader node (root node). We assume that it takes one time slot per child node to transfer data to its immediate predecessor node in the tree. However, a node cannot transfer the aggregated data to its parent node until it receives the data from its own child nodes. The delay calculations start from the bottom of the data gathering tree. The delay incurred at a leaf node is 0. To calculate the delay incurred at an intermediate node u, *Delay*(u), located at a particular level in the data gathering tree, we maintain a sorted list, *Child-Nodes*(u), of the delay associated with each of its immediate child nodes and use a temporary running variable *Temp-Delay*(u), initialized to zero, to explore the sorted list of the delays at the child nodes. For every child node $v \in$ *Child-Nodes*(u), *Temp-Delay*(u) =

Maximum [*Temp-Delay*(*u*) + 1, *Delay*(*v*) + 1)], as we assume it takes one time slot for a child node to transfer its aggregated data to its immediate predecessor node in the tree. The delay associated with an intermediate node *u*, *Delay*(*u*), is the final value of the *Temp-Delay*(*u*) variable, after we iterate through the sorted list of the delays associated with the list *Child-Nodes*(*u*). The above procedure is repeated at all the intermediate nodes, from levels one less than the *Height* of the tree all the way to zero (i.e., the root node). We illustrate the working of the above explained procedure for delay computation on a data gathering tree through an example presented in Figure 3. The integer inside a circle indicates the node ID and the integer outside a circle indicates the delay for data aggregation at the node.

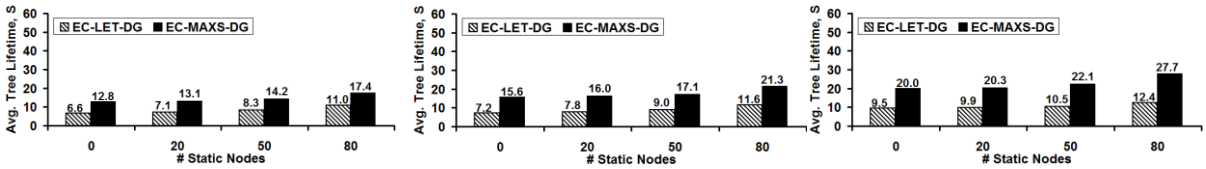
7.3 Tree Lifetime

Among the three key operating parameters (maximum node velocity, number of static nodes and transmission range per node) of the simulations, we observe the stability of the data gathering trees to be highly influenced by the maximum node velocity (v_{max}) of the nodes. When operated under sufficient-energy scenarios, for a fixed number of static nodes and transmission range per node, we observe the lifetime incurred for both the MaxS-DG trees and LET-DG trees to proportionally decrease with a corresponding increase in the v_{max} values from 3 m/s to 10 m/s. In the energy-constrained scenarios, even though a data gathering tree may topologically exist, the tree would require reconfiguration if one or more nodes in the tree fail due to exhaustion of battery charge. 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. In the case of both the MaxS-DG and LET-DG trees, for a fixed transmission range and # static nodes, we observe the largest difference between the tree lifetimes for the sufficient-energy scenarios vis-à-vis the energy-constrained scenarios to occur when the network is operated under low node mobility conditions (v_{max} = 3 m/s). This could be attributed to the significantly longer lifetime observed for the data gathering trees at low node mobility conditions when operated with sufficient-energy for the nodes.



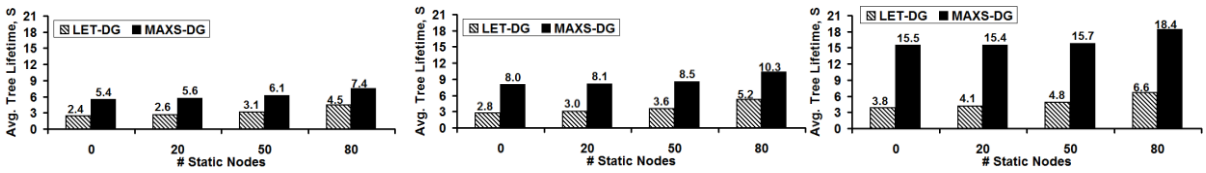
Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 4: Average Tree Lifetime under Sufficient Energy Scenario (Low Node Mobility: v_{max} = 3 m/s)



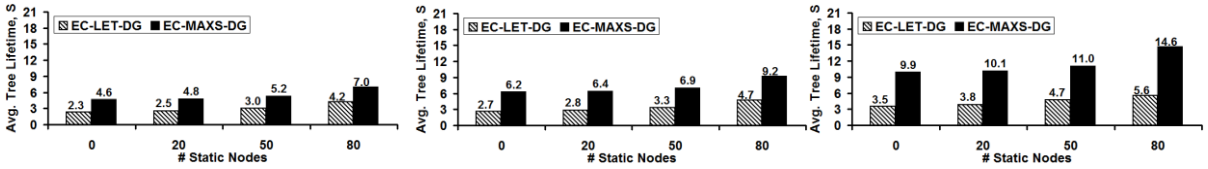
Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 5: Average Tree Lifetime under Energy Constrained Scenario (Low Node Mobility: v_{max} = 3 m/s)



Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 6: Average Tree Lifetime under Sufficient Energy Scenario (High Node Mobility: v_{max} = 10 m/s)



Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 7: Average Tree Lifetime under Energy Constrained Scenario (High Node Mobility: v_{max} =10 m/s)

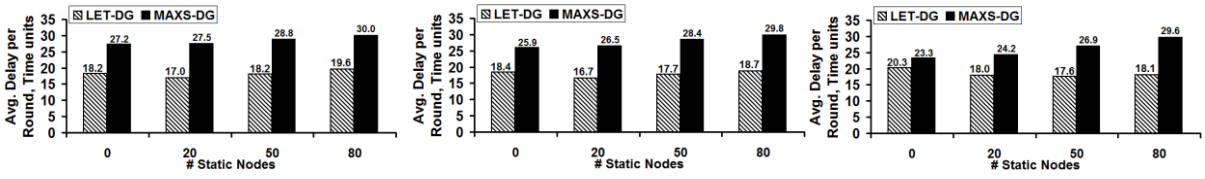
In low mobility scenarios (refer Figures 4 and 5), we also observe the difference in the tree lifetimes under sufficient-energy vs. energy-constrained scenarios to increase with increase in the transmission range per node. At higher transmission ranges, the links are more stable as nodes of a link have relatively higher freedom to move around (compared to operating at low and moderate transmission ranges) and still remain as neighbors. Hence, the data gathering trees are bound to be the most stable at low node mobility and larger transmission ranges per node. At these conditions – under sufficient-energy scenarios, we observe the MaxS-DG trees to sustain a lifetime that is larger than that of the LET-DG trees by a factor of about 3 to 4.5. However, under energy-constrained scenarios, the MaxS-DG trees are only 100-125% more stable than that of the LET-DG Trees. Nevertheless, the energy savings sustained by the MaxS-DG algorithm with respect to tree discoveries under both low and high node mobility scenarios contributes to the nodes on a MaxS-DG tree to exist for a relatively much longer time compared to that of the LET-DG trees, resulting in an increased network lifetime (refer Section 7.5).

With regards to the impact of the transmission range per node, the difference in the lifetime of the MaxS-DG trees and the LET-DG trees increases with increase in the transmission range per node, for a given level of node mobility. For a fixed v_{max} value, the lifetime of the MaxS-DG trees increases by a factor of 2 to 3 as we increase the transmission range from 25m to 40m; whereas the lifetime of the LET-DG trees increases only at most by a factor of 2. This could be again attributed to the optimal usage of the availability of stable links (facilitated by the larger transmission ranges per node) by the MaxS-DG algorithm through a centralized, look-ahead and graph intersection approach. However, as is the bane of the distributed algorithms based on the local optimum approach, the LET-DG trees are formed with links that are relatively less stable even when operated with higher transmission ranges per node.

With regards to the impact of the number of static nodes, we observe that for both the sufficient-energy and energy-constrained scenarios, the lifetime of both the MaxS-DG trees and the LET-DG trees increases by at most 50% when the number of static nodes is increased from 0 to 80 nodes. There is not much of a significant increase (only at most about 10-15% increase) in the lifetime of both the data gathering trees when we run the network with 20 and 50 static nodes instead of 0 nodes. This vindicates the impact of node mobility on the stability of the data gathering trees. Even if half of the nodes in the network are operated static, we observe the data gathering trees to have about the same vulnerability for a link failure vis-à-vis operating the network with all mobile nodes.

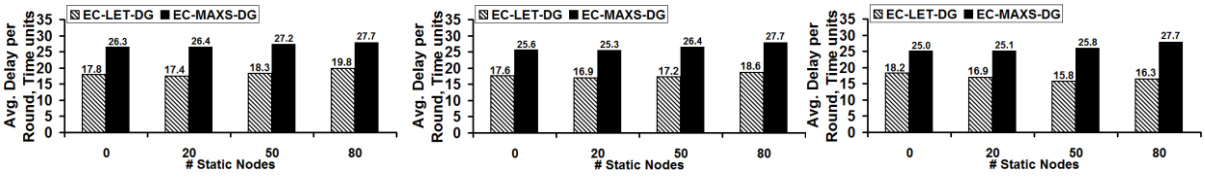
7.4 Delay per Round

We observe the LET-DG trees to incur significantly lower delay per round of data gathering compared to the MaxS-DG trees. The delay per round is not much affected by the dynamicity of the network and is more impacted by the topological structure of the two spanning trees. The MaxS-DG tree tends to have relatively fewer leaf nodes, and as a result more nodes are likely to end up as intermediate nodes – leading to a much larger depth. Note that the underlying link weight criterion used for MaxS-DG trees is the geometric Euclidean distance between the end nodes of the link. The MaxS-DG tree is also observed to be more unbalanced with respect to the distribution of the number of children per intermediate node as well as the distribution of the leaf nodes at different levels. Not all leaf nodes are located at the bottommost level of the tree. Due to all these structural complexities, the MaxS-DG trees have been observed to incur a much larger delay per round of data gathering. On the other hand, the LET-DG trees have been observed to be more shallow (i.e., lower depth) with more leaf nodes and the distribution of the number of child nodes per intermediate node is relatively more balanced. All of these factors contribute to a much lower delay per round of data gathering.



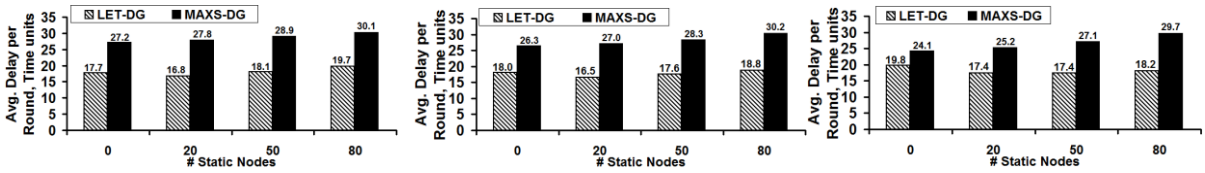
Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 8: Average Delay / Round under Sufficient Energy Scenario (Low Node Mobility: v_{max} = 3 m/s)



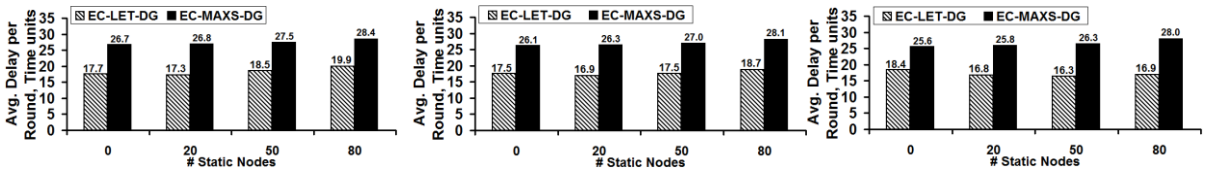
Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 9: Average Delay / Round under Energy Constrained Scenario (Low Node Mobility: v_{max} = 3 m/s)



Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 10: Average Delay / Round under Sufficient Energy Scenario (High Node Mobility: v_{max} = 10 m/s)



Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 11: Average Delay/Round under Energy Constrained Scenario (High Node Mobility: v_{max}=10 m/s)

Across all the simulations, we observe the MaxS-DG trees to incur on average a 40-65% larger delay per round of data gathering. For a given maximum node mobility, the difference in the delay per round of data gathering between the MaxS-DG and LET-DG trees decreases with increase in the transmission range per node. While operating the network at larger transmission ranges per node, it is possible to obtain a slightly better distribution of the nodes across the different levels of the MaxS-DG tree, contributing to the reduction in the delay. For a given v_{max} and transmission range per node, we also observe the difference in the magnitudes of the delay per round between the MaxS-DG and LET-DG trees to increase with increase in the number of static nodes. This can be attributed to the reduced chances of changes to the topological structure of the MaxS-DG tree in the presence of more static nodes – the unbalanced distribution of the nodes at the different levels of the tree gets to continue for a longer time – contributing to the larger delay.

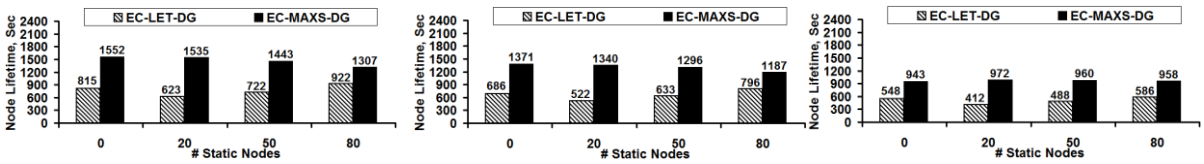
We observe the energy-constrained scenarios to have only minimal impact on the delay per round of data gathering. The two data gathering trees incur only a slightly lower delay per round of data gathering (by a factor of 5-10%) when operated in energy-constrained scenarios compared to the sufficient energy scenarios. The reduction in the delay per round of data gathering in the presence of node failures could be attributed to the overall reduction in the number of time slots needed to gather data from around the nodes in the network. The impact of node failures and the energy-constraint on the delay per round is almost equally observed for both the LET-DG and MaxS-DG trees.

7.5 Node Lifetime and Network Lifetime

The MaxS-DG trees incur a larger node lifetime and network lifetime compared to the LET-DG trees. For a given transmission range per node, the difference in the node lifetime and network lifetime increases with increase in node velocity. This could be attributed to the unstable LET-DG trees at higher node mobility levels and the increase in the number of network-wide flooding based tree discoveries. On the other hand, the MaxS-DG trees are

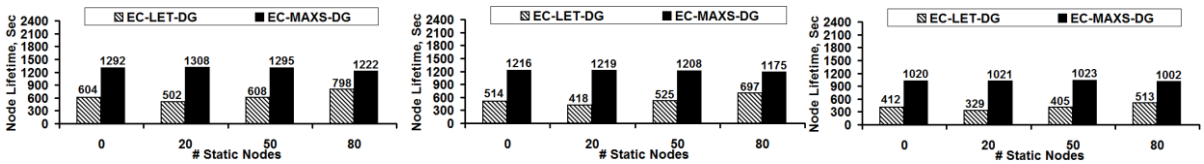
relatively much more stable than the LET-DG trees (though the absolute magnitude of the tree lifetime is lower for both the data gathering trees at high node mobility levels) and hence incur lower energy loss in flooding-based tree discoveries.

We observe the LET-DG trees to incur the first node failure (node lifetime) much earlier during the simulation (compared to that incurred with the MaxS-DG trees). The node lifetime incurred with the MaxS-DG trees is observed to be 150-250% and 175-300% larger than that incurred with the LET-DG trees at low and high node mobility levels respectively. This could be attributed to the relatively shallow structure of the LET-DG trees – only fewer nodes serve as intermediate nodes of the data gathering tree and they spend more energy in gathering data from all their children/leaf nodes and forwarding the aggregated data further upstream in the tree. Due to the stable nature of the LET-DG trees, the intermediate nodes continue to lose more energy compared to the other nodes (leaf nodes) in the trees – leading to premature node failures. However, the difference in the network lifetime between the LET-DG and MaxS-DG trees is considerably lower; the network lifetime sustained with the MaxS-DG trees is only at most 60% larger than that incurred with the LET-DG trees (much smaller difference in the network lifetime, compared to the difference in node lifetime as noted above). The difference in the network lifetime between the MaxS-DG and LET-DG trees increases primarily with increase in the node mobility; however, at low node mobility, the network lifetime of the LET-DG trees considerably increases with increase in the transmission range per node. At v_{max} = 3 m/s and 40m transmission range per node, the network lifetime incurred with the MaxS-DG trees is only at most 15% larger than that incurred with the LET-DG trees.



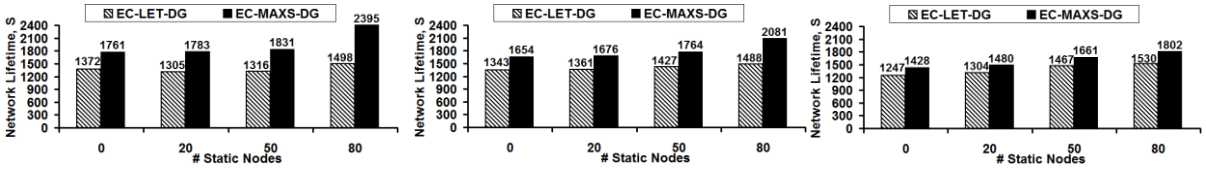
Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 12: Average Node Lifetime (Low Node Mobility: v_{max} = 3 m/s)



Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

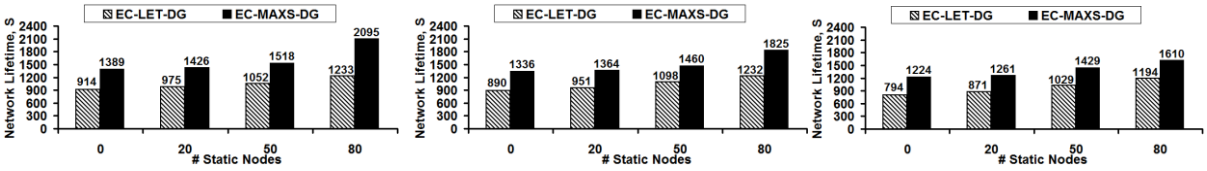
Figure 13: Average Node Lifetime (High Node Mobility: v_{max} = 10 m/s)



Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 14: Average Network Lifetime (Low Node Mobility: v_{max} = 3 m/s)

With respect to the impact of the presence of static nodes in the network, as we increase the number of static nodes from 0 to 80, we observe the node lifetime observed with the LET-DG trees to increase by about 15% and 30% at v_{max} values of 3 m/s and 10 m/s respectively. At high node mobility, the presence of more static nodes definitely helps the LET-DG trees to be more stable (note in Section 7.3, the lifetime of LET-DG trees at v_{max} = 10 m/s could be as large as double the lifetime incurred at v_{max} = 3 m/s), leading to a reduction in the energy lost due to network-wide flooding-based tree discoveries. On the other hand, the MaxS-DG trees do not sustain any significant increase in tree lifetime when the number of static nodes is increased from 0 to 80; hence, there is no significant energy savings (in flooding) with little increase in the MaxS-DG tree lifetime. As a result, owing to the inherent stable nature of the MaxS-DG trees that exist for the longest possible time (there could be an increased use of certain nodes at the cost of others), we even observe the node lifetime incurred with the MaxS-DG trees to decrease slightly increase with increase in the number of static nodes from 0 to 80 for a given v_{max} and transmission range per node.



Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 15: Average Network Lifetime (High Node Mobility: v_{max} = 10 m/s)

With respect the impact of the operating parameters on the absolute magnitude of the network lifetime, we observe the network lifetime incurred with the two data gathering trees increases with increase in the number of static nodes for a given value of v_{max} and transmission range per node. The percentage increase in the network lifetime relative to the node lifetime varies for the two data gathering trees. For a given v_{max} and transmission range per node, the network lifetime incurred with the LET-DG trees is consistently about 50% more than that of the node lifetime for all values of the number of static nodes operated. On the other hand, with the MaxS-DG trees, the network lifetime incurred when all nodes are mobile (i.e., 0 static nodes) is only about 10-20% more than that of the node lifetime; however, when we operate with 80 static nodes (out of a total of 100 nodes), the network lifetime incurred with the MaxS-DG trees increases significantly (compared to the node lifetime) by as large as 100%.

For a given level of node mobility, the network lifetime incurred for the two data gathering trees decreases with increase in transmission range per node. This could be attributed to the increased energy expenditure in the transmission of messages as well as flooding-based tree discoveries. For a given transmission range per node and number of static nodes, the network lifetime incurred for the two data gathering trees decreases with increase in the maximum node velocity, especially for the LET-DG trees due to their relative instability and energy loss incurred due to frequent tree discoveries. The network lifetime incurred with the MaxS-DG trees and MST-DG trees decreases by about 15-25% and 50-60% respectively as we increase the maximum node velocity from 3 m/s to 10 m/s for a fixed transmission range per node and number of static nodes.

8. Conclusions and Future Work

The high-level contributions of this paper in the area of mobile sensor networks are two fold: (1) Design and development of a centralized benchmarking algorithm to determine maximum stability data gathering (MaxS-DG) trees whose lifetime forms upper bound for the maximum lifetime that can be incurred with data gathering trees for mobile sensor networks; (2) Design and development of a distributed algorithm to determine stable predicted link expiration time-based data gathering (LET-DG) trees that can also incur lower delay per round.

Given the entire sequence of topology changes over the duration of the data gathering session as input, the MaxS-DG algorithm returns the sequence of longest-living stable data gathering trees such that the number of tree discoveries is the global minimum. The run-time complexity of the algorithm has been observed to be $O(n^2T\log n)$ and $O(n^3T\log n)$ when operated under sufficient-energy and energy-constrained scenarios respectively, where n is the number of nodes in the network and T is the duration of the data gathering session. Since the MaxS-DG trees are spanning tree-based and a spanning tree exists in a network if and only if the network is connected, the stability of a spanning tree or any network-wide communication topology (like a connected dominating set) discovered by an existing or prospective data gathering algorithm can be evaluated by comparing its lifetime with that obtained for the MaxS-DG trees. With a polynomial-time complexity and a much broader scope of application, as described above, the MaxS-DG algorithm has all the characteristics to become a global standard for evaluating the stability of communication topologies for data gathering in mobile sensor networks.

Table 1. Influence of the Operating Parameters on the Performance of the Data Gathering Trees

Performance Metric	**Ranking of the Operating Parameters in the Order of Influence [*1-Highest Influence*]**		
	Node Velocity	***Static Nodes***	***Transmission Range per Node***
Tree Lifetime	1	3	2
Delay per Round	3	2	1
Node Lifetime	1	3	2
Network Lifetime	1	2	3

One salient feature of the LET-DG algorithm is that it does not require the periodic beacon exchange of beacons in the neighborhood of the sensor nodes. Though the algorithm does not incur as large a lifetime as that of the benchmark values observed for the MaxS-DG algorithm, the LET-DG algorithm is the first such distributed stability-based data gathering algorithm for mobile sensor networks. The LET-DG trees also incur a significantly lower delay per round of data gathering (compared to the MaxS-DG trees) – thus, implying a stability-delay tradeoff for data gathering in mobile sensor networks. We also observe the LET-DG trees to sustain a comparable network lifetime (that is at most 60% smaller) than that incurred with the MaxS-DG trees whose larger network lifetime can be attributed to the minimal use of network-wide flooding based tree discoveries (due to the optimal tree lifetime incurred).

Table 1 ranks the three operating parameters in the decreasing order of influence on the performance of the two data gathering trees. The nature of influence is identical for both the data gathering trees.

As part of future work, we plan to compare the stability of the MaxS-DG trees under several different node mobility models [50] vis-à-vis the Random waypoint model, the mobility model used in our simulations that has been widely used in the ad hoc network literature. Also, as stable data gathering trees are likely to be used for a longer time, the trustworthiness of the data aggregated at the intermediate nodes needs to be validated and maintained through proper trust-evaluation schemes. We plan to develop and integrate a trust-evaluation model as part of stable data aggregation in mobile sensor networks.

Acknowledgment

This research was sponsored by the U. S. Air Force Office of Scientific Research (AFOSR) through the Summer Faculty Fellowship Program for the lead author (Natarajan Meghanathan) in June-July 2012. The research was conducted under the supervision of the co-author (Philip Mumford) at the U. S. Air Force Research Lab (AFRL), Wright-Patterson Air Force Base (WPAFB) Dayton, OH. The AFRL public release approval # is 88ABW-2013-2005. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agency. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] Kansal, A., Goraczko, M., and Zhao, F., “Building a Sensor Network of Mobile Phones”, International Symposium on Information Processing in Sensor Networks, pp. 547-548, Cambridge, MA, USA, April 25-27, 2007. http://dx.doi.org/10.1145/1236360.1236433

[2] Hull, B., Bychkovsky, V., Zhang, Y., Chen, K., Goraczko, M., Miu, A., Shih, E., Balakrishnan, H., and Madden, S., “CarTel: A Distributed Mobile Sensor Computing System”,

The 4th International Conference on Embedded Networked Sensor Systems, pp. 125-138, Boulder, USA, October 31-November 3, 2006. http://dx.doi.org/10.1145/1182807.1182821

[3] Heinzelman, W., Chandrakasan, A., and Balakarishnan, H., "Energy-Efficient Communication Protocols for Wireless Microsensor Networks", Hawaaian International Conference on Systems Science, pp. 1-10, Maui, HI, January 4-7, 2000. http://dx.doi.org/10.1109/HICSS.2000.926982

[4] Lindsey, S., Raghavendra, C., Sivalingam, K. M., "Data Gathering Algorithms in Sensor Networks using Energy Metrics", IEEE Transactions on Parallel and Distributed Systems, Vol. 13, Issue 9, pp. 924-935, September 2002. http://dx.doi.org/10.1109/TPDS.2002.1036066

[5] Srinivasan, A., and Wu, J., "TRACK: A Novel Connected Dominating Set based Sink Mobility Model for WSNs", *17th International Conference on Computer Communications and Networks*, pp. 1-8, St. Thomas, US Virgin Islands, August 3-7, 2008. http://dx.doi.org/10.1109/ICCCN.2008.ECP.127

[6] Vlajic, N., and Stevanovic, D., "Sink Mobility in Wireless Sensor Networks: When Theory meets Reality", IEEE Sarnoff Symposium, pp. 1-8, Princeton, NJ, USA, March 30-April 1, 2009. http://dx.doi.org/10.1109/SARNOF.2009.4850301

[7] Meghanathan, N., "A Comprehensive Review and Performance Analysis of Data Gathering Algorithms for Wireless Sensor Networks", International Journal of Interdisciplinary Telecommunications and Networking, Vol. 4, Issue 2, pp. 1-29, April-June 2012. http://dx.doi.org/10.4018/jitn.2012040101

[8] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., "Graph Algorithms" *in Introduction to Algorithms*, 3rd ed. MIT Press, 2009, pp. 587-671.

[9] Su, W., and Gerla, M., "IPv6 Flow Handoff in Ad hoc Wireless Networks using Mobility Prediction", IEEE Global Telecommunications Conference, pp. 271-275, Rio de Janeiro, Brazil, December 5-9, 1999. http://dx.doi.org/10.1109/GLOCOM.1999.831647

[10] Su, W., Lee, S-J., Gerla, M., "Mobility Prediction and Routing in Ad Hoc Wireless Networks", International Journal of Network Management, Vol. 11, Issue 1, pp. 3-30, January-February, 2001. http://dx.doi.org/10.1002/nem.386

[11] Wu, W., Beng Lim, H., and Tan, K-L., "Query-driven Data Collection and Data Forwarding in Intermittently Connected Mobile Sensor Networks", The 7th International Workshop on Data Management for Sensor Networks, pp. 20-25, Singapore, September 13, 2010. http://dx.doi.org/10.1145/1858158.1858166

[12] Xing, G., Wang, T., Jia, W., and Li, M., "Rendezvous Design Algorithms for Wireless Sensor Networks with a Mobile Base Station", The 9th ACM International Symposium on Mobile Ad hoc Networking and Computing, pp. 231-240, Hong Kong SAR, China, May 27-30, 2008. http://dx.doi.org/10.1145/1374618.1374650

[13] Zhao, M., and Yang, Y., "Bounded Relay Hop Mobile Data Gathering in Wireless Sensor Networks", The 6th IEEE International Conference on Mobile Ad hoc and Sensor Systems, pp. 373-382, October 12-15, 2009. http://dx.doi.org/10.1109/MOBHOC.2009.5336976

[14] Kurs, A., Karalis, A., Moffatt, R., Joannopoulos, J. D., Fisher, P., Soljacic, M., "Wireless Power Transfer via Strongly Coupled Magnetic Resonances", Science, Vol. 347, Issue 5834, pp. 83-86, July 2007. http://dx.doi.org/10.1126/science.1143254

[15] Guo, S., Wang, C., and Yang, Y., "Mobile Data Gathering with Wireless Energy

Replenishment in Rechargeable Sensor Networks", IEEE International Conference on Computer Communications (INFOCOM), pp. 1932-1940, Turin, Italy, April 2013. http://dx.doi.org/10.1109/INFCOM.2013.6566993

[16] Angelopoulos, C. M., Nikoletseas,S., Raptis, T. P., "Efficient Wireless Recharging in Sensor Networks", The 2013 IEEE International Conference on Distributed Computing in Sensor Systems, pp. 298-300, Cambridge, MA, USA, May 20-23, 2013. http://dx.doi.org/10.1109/DCOSS.2013.10

[17] Banerjee, T., Xie, B., Jun, J. H. and Agarwal, D. P., "LIMOC: Enhancing the Lifetime of a Sensor Network with Mobile Clusterheads", The Vehicular Technology Conference Fall, pp. 133-137, Baltimore, USA, October 1-3, 2007. http://dx.doi.org/10.1109/VETECF.2007.43

[18] Santhosh Kumar, G., Vinu Paul, M. V., and Jacob Poulose, K., "Mobility Metric based LEACH-Mobile Protocol", 16th International Conference on Advanced Computing and Communications, pp. 248-253, Chennai, India, December 14-17, 2008. http://dx.doi.org/10.1109/ADCOM.2008.4760456

[19] Deng, S., Li, J., Shen, L., "Mobility-based Clustering Protocol for Wireless Sensor Networks with Mobile Nodes", IET Wireless Sensor Systems, Vol. 1, Issue 1, pp. 39-47, March 2011. http://dx.doi.org/10.1049/iet-wss.2010.0084

[20] Sarma, H. K. D., Kar, A., and Mall, R., "Energy Efficient and Reliable Routing for Mobile Wireless Sensor Networks", 6th IEEE International Conference on Distributed Computing in Sensor Systems Workshops, pp. 1-6, Santa Barbara, CA, USA, June 21-23, 2010. http://dx.doi.org/10.1109/DCOSSW.2010.5593277

[21] Li, P., and Jian-bo, X., "ECDGA: An Energy-Efficient Cluster-Based Data Gathering Algorithm for Mobile Wireless Sensor Networks", International Conference on Computational Intelligence and Software Engineering, pp. 1-4, Wuhan, China, December 11-13, 2009. http://dx.doi.org/10.1109/CISE.2009.5366773

[22] Liu, C-M., Lee, C-H., and Wang, L-C., "Distributed Clustering Algorithms for Data Gathering in Wireless Mobile Sensor Networks", Journal of Parallel and Distributed Computing, Vol. 67, Issue 11, pp. 1187-1200, November 2007. http://dx.doi.org/10.1016/j.jpdc.2007.06.010

[23] Macuha, M., Tariq, M., and Sato, T., "Data Collection Method for Mobile Sensor Networks based on the Theory of Thermal Fields," Sensors, Vol. 11, Issue 7, pp. 7188-7203, July 2011. http://dx.doi.org/10.3390/s110707188

[24] Singh, M., Sethi, M., Lal, N., Poonia, S., "A Tree Based Routing Protocol for Mobile Sensor Networks (MSNs)", International Journal on Computer Science and Engineering, Vol. 2, Issue 1S, pp. 55-60, 2010.

[25] Meghanathan, N., "A Simulation-based Performance Analysis of Multicast Routing in Mobile Ad hoc Networks", International Journal of Information Processing and Management, Vol. 1, Issue 1, pp. 4-14, July 2010. http://dx.doi.org/10.4156/ijipm.vol1.issue1.1

[26] Meghanathan, N., and Farago, A., "On the Stability of Paths, Steiner Trees and Connected Dominating Sets in Mobile Ad Hoc Networks", Ad hoc Networks, Vol. 6, Issue 5, pp. 744-769, July 2008. http://dx.doi.org/10.1016/j.adhoc.2007.06.005

[27] Meghanathan, N., "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, Issue 3, pp. 1-14, June 2011. http://dx.doi.org/10.5121/ijwmn.2011.3301

[28] Meghanathan, N., "Routing Protocols to Determine Stable Paths and Trees using the Inverse of Predicted Link Expiration times for Mobile Ad hoc Networks", International Journal of Mobile Network Design and Innovation, Vol. 4, Issue 4, pp. 214-234, December 2012. http://dx.doi.org/10.1504/IJMNDI.2012.054463

[29] Meghanathan, N., "A Link Distance Ratio based Stable Multicast Routing Protocol for Mobile Ad hoc Networks", Springer-Verlag Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Series, LNICST 84, pp. 253-262, January 2012. http://dx.doi.org/10.1007/978-3-642-27299-8_27

[30] Meghanathan, N., "Node Stability Index: A Stability Metric and an Algorithm to Determine Long-Living Connected Dominating Sets for Mobile Ad hoc Networks", International Journal of Interdisciplinary Telecommunications and Networking, Vol. 4, Issue 1, pp. 31-46, January-March 2012. http://dx.doi.org/10.4018/jitn.2012010102

[31] Meghanathan, N., and Thompson, J. A., "On the Different Forms of Spanning Tree-based Broadcast Topologies for Mobile Ad hoc Networks", International Journal of Combinatorial Optimization Problems and Informatics, Vol. 4, Issue 1, pp. 3-11, January-April 2013.

[32] Toh, C-K., "Associativity-Based Routing for Ad hoc Mobile Networks," IEEE Personal Communications, Vol. 4, Issue 2, pp. 103-109, March 1997. http://dx.doi.org/10.1023/A:1008812928561

[33] Agarwal, S., Ahuja, A., Singh, J. P. and Shorey, R., "Route-Life Time Assessment Based Routing Protocol for Mobile Ad hoc Networks", The IEEE International Conference on Communications, vol. 3, pp. 1697-1701, New Orleans, LA, USA, June 18-22, 2000. http://dx.doi.org/10.1109/ICC.2000.853783

[34] Meghanathan, N., "Exploring the Performance Tradeoffs among Stability-Oriented Routing Protocols for Mobile Ad hoc Networks", Network Protocols and Algorithms - Special Issue on Data Dissemination for Large scale Complex Critical Infrastructures, Vol. 2, Issue 3, pp. 18-36, November 2010. http://dx.doi.org/10.5296/npa.v2i3.436

[35] Lee, S-J., and Gerla, M., "On-demand Multicast Routing Protocol", IEEE International Wireless Communications and Networking Conference, vol. 3, pp. 1298-1302, New Orleans, USA, September 21-24, 1999. http://dx.doi.org/10.1109/WCNC.1999.796947

[36] Ozaki, T., Kim, J-B., and Suda, T., "Bandwidth-Efficient Multicast Routing for Multihop, Ad hoc Wireless Networks", IEEE International Conference on Computer Communications, vol. 2, pp. 1182-1192, Anchorage, AK, USA, April 22-26, 2001. http://dx.doi.org/10.1109/INFCOM.2001.916313

[37] Royer, E., and Perkins, C. E., "Multicast Operation of the Ad-hoc On-demand Distance Vector Routing Protocol", 5th ACM/IEEE Annual Conference on Mobile Computing and Networking, pp. 207-218, Seattle, WA, USA, August 15-20, 1999. http://dx.doi.org/10.1145/313451.313538

[38] Meghanathan, N., Vavilala, S. R., "Impact of Route Selection Metrics on the Performance of On-Demand Mesh-based Multicast Ad hoc Routing Protocols", Journal of Computer and Information Science, Vol. 3, Issue 2, pp. 3-18, May 2010.

[39] Sims, J., Meghanathan, N., "Construction and Evaluation of Meshes based on Shortest Path Tree vs. Steiner Tree for Multicast Routing in Mobile Ad hoc Networks", International Journal of Theoretical and Applied Information Technology, Vol. 19, Issue 2, pp. 134-142, September 2010.

[40] Meghanathan, N., "Performance Comparison Study of Multicast Routing Protocols for Mobile Ad hoc Networks under Default Flooding and Density and Mobility Aware Energy-Efficient (DMEF) Broadcast Strategies," Informatica - An International Journal of Computing and Informatics, Vol. 35, Issue 2, pp. 165-184, June 2011.

[41] Meghanathan, N., and Terrell, M., "An Algorithm to Determine Stable Connected Dominating Sets for Mobile Ad hoc Networks using Strong Neighborhoods", International Journal of Combinatorial Optimization Problems and Informatics (IJCOPI), Vol. 3, Issue 2, pp. 79-92, May - August 2012.

[42] Meghanathan, N., "A Simulation-based Performance Comparison of the Minimum Node Size and Stability-based Connected Dominating Sets for Mobile Ad hoc Networks", International Journal of Computers and Network Communications, Vol. 4, Issue 2, pp. 169-184, March 2012. http://dx.doi.org/10.5121/ijcnc.2012.4211

[43] Meghanathan, N., "Node Stability Index: A Stability Metric and an Algorithm to Determine Long-Living Connected Dominating Sets for Mobile Ad hoc Networks", International Journal of Interdisciplinary Telecommunications and Networking, Vol. 4, Issue 1, pp. 31-46, January-March 2012. http://dx.doi.org/10.4018/jitn.2012010102

[44] Meghanathan, N., "Use of Minimum Node Velocity Based Stable Connected Dominating Sets for Mobile Ad hoc Networks", International Journal of Computer Applications: Special Issue on Recent Advancements in Mobile Ad hoc Networks, Vol. 2, pp. 89-96, September 2010. http://dx.doi.org/10.5120/1016-52

[45] Kuhn, F., Moscibroda, T., and Wattenhofer, R., "Unit Disk Graph Approximation", Workshop on the Foundations of Mobile Computing (DIALM-POMC), pp. 17-23, Philadelphia, PA, USA, October 1, 2004. http://dx.doi.org/10.1145/1022630.1022634

[46] Rappaport, T. S. "Mobile Radio Propagation: Large-Scale Path Loss", in *Wireless Communications*: *Principles and Practice*, 2nd ed., Prentice Hall, January 2002.

[47] Farago, A., Syrotiuk, V. R., "MERIT: A Scalable Approach for Protocol Assessment", Mobile Networks and Applications, Vol. 8, Issue 5, pp. 567-577, October 2003. http://dx.doi.org/10.1023/A:1025193929081

[48] Viterbi, A. J., "Coding and Interleaving", in *CDMA*: *Principles of Spread Spectrum Communication*, 1st ed., Prentice Hall, April 1995.

[49] Bettstetter, C., Hartenstein, H., Perez-Costa, X., "Stochastic Properties of the Random-Way Point Mobility Model", Wireless Networks, Vol. 10, Issue 5, pp. 555-567, September 2004. http://dx.doi.org/10.1023/B:WINE.0000036458.88990.e5

[50] Camp, T., Boleng, J., Davies, V., "A Survey of Mobility Models for Ad Hoc Network Research", Wireless Communication and Mobile Computing, Vol. 2, Issue 5, pp. 483-502, September 2002. http://dx.doi.org/10.1002/wcm.72