GIST: Group-Independent Spanning Tree for Data Aggregation in Dense Sensor Networks

Lujun Jia[1], Guevara Noubir[1], Rajmohan Rajaraman[1], and Ravi Sundaram[1]

College of Computer and Information Science
Northeastern University, Boston, MA 02115, USA
{lujunjia, noubir, rraj, koods}@ccs.neu.edu

Abstract. Today, there exist many algorithms and protocols for constructing agregation or dissemination trees for wireless sensor networks that are optimal (for different notions of optimal, i.e. under different cost metrics). However, all these schemes differ from one common failing - they construct an optimal tree for a given *fixed* subset of the sensors. In most practical scenarios, the sensor group is continuously and dynamically *varying* - consider for example the set of sensors scattered in a forest that are sensing temperatures above some specified threshold, during a wildfire. Given the limited computational and energy resources of sensor nodes it is impossible to either prestore the optimal tree for every conceivable group or to dynamically generate them on the fly.
In this paper we propose the novel approach of constructing a *single* group-independent spanning tree (GIST) T for the network and then letting any sensor group S use the subtree induced by S on T, T_S as its group aggregation tree. The important question is, how does the quality of the subtree T_S compare to the optimal tree, OPT_S, across different groups S. We consider two well-accepted measures - aggregation cost (sum over all links) and delay (diameter). We show that in polynomial time we can construct a GIST that simultaneously achieves $O(\log n)$-approximate aggregation cost and $O(1)$-approximate delay, for all groups S.
To the best of our knowledge GIST is the first construction with a non-trivial and provable performance guarantee that works for all groups. We provide a practical and distributed protocol for realizing GIST that requires only local knowledge. We show an $\Omega(n)$ lower bound for commonly accepted solutions such as MST and SPT (i.e. there exists a group for which the induced subtree performs poorly) and demonstrate by simulation that GIST is good not just in the worst case - it outperforms SPT and MST by between 30 and 60 per cent in realistic random scenarios. GIST is an overlay construction and for the special case of grids we present GRID-GIST, a physical tree that uses only grid edges and achieves the same performance bounds.

1 Introduction

Wireless sensor networks have emerged at the forefront of applications involving the measurement of physical phenomena, environmental monitoring, medical

instrumentation, and building monitoring in warehouses and homes. A wireless sensor network is comprised of hundreds or thousands of sensor nodes networked through wireless links for collecting and processing environmental data. Sensor networks have stringent energy restrictions, and are typically deployed in high density to ensure coverage and fault tolerance.

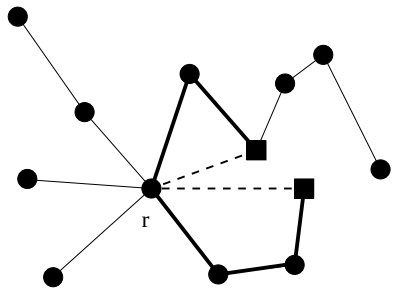

Fig. 1. In the above network, a spanning tree T is depicted by solid edges. A subset S consists of the two square nodes. A subtree of T induced by S is depicted by the thick edges. Clearly, the subtree is far from optimal, which includes the two dotted edges.

With the advent of large-scale sensor network applications, there is considerable interest in a database abstraction for sensor networks, in which users program the sensors using a high-level declarative language [16, 31, 38]. A user issues a query to the network, and every sensor node that meets the criteria defined in the query replies with the desired data reading. In the reply phase, data can be aggregated in-network to reduce communication complexity, and hence energy consumption [28, 31]. Data aggregation is usually performed using a reverse multicast tree, in which each intermediate node receives packets from its children, aggregates the information, and sends one packet to its parent. As shown in multiple studies, this can lead to considerable savings in energy consumption over an approach that does not use in-network aggregation [16, 28, 31].

Consider a sensor network deployed in a large forest for monitoring forest fires. A user may issue the following query (written in a variant of SQL) to the whole network:

```
SELECT MAX(temperature), location FROM sensors
WHERE temperature > threshold
DURATION (now,now+3600s)
EVERY 30s
```

In the above declarative query, the user asks for the maximum temperature and its location, every 30 seconds for a duration of an hour, if this maximum temperature is above a certain threshold. Due to temperature changes in different areas of the forest, the group of sensors satisfying the threshold criteria may

change continually. Therefore, the aggregation group (the group of sensors that need to send readings to the user) changes in an unpredictable manner over time, and the total number of such groups is large. In such a scenario, the following dilemma seems unavoidable:

- Since the number of different groups is large – exponential in the number of sensors, in the worst case – it is prohibitively expensive for the energy-constrained, distributed and multi-hop sensor nodes to compute efficient aggregation trees for each group.
- On the other hand, if a communication-optimized data aggregation structure is not used, the aggregation itself can be expensive.

Most existing algorithms and protocols for constructing data aggregation trees have drawbacks in that one aggregation tree has to be constructed and maintained for one group of sensors [3, 19, 27, 31]. Such aggregation protocols are not suitable for large-scale sensor networks serving aggregation queries at high frequencies. We refer to these protocols as group-dependent data aggregation protocols. Group-dependence, or group-awareness, makes the optimization of communication cost possible, however only for a single group. Therefore, in sensor applications where the sensor groups of interest evolve constantly or different queries are issued on a frequent basis, new solutions are desired to support effective in-network aggregation.

We propose GIST (Group-Independent Spanning Tree) for data aggregation in sensor networks. A GIST, T is a single spanning tree that is "oblivious" to any aggregation group in the sense that the aggregation structure adopted for any group is simply the subtree of T induced by the group. The performance of GIST for a given group is measured by comparing the cost of this induced subtree with the cost of an optimal aggregation tree for that group. Figure 1 illustrates a group-independent spanning tree and its subtree induced by a subset of nodes. It is not clear a priori that it is possible to find such a group-independent tree with good guarantee. Two natural candidates for GIST are the minimum spanning tree (MST) and the shortest-paths tree (SPT). However as shown in [22], they both have $\Omega(n)$ worst case performance.

1.1 Main results

- We propose the group-independent tree paradigm for providing effective underlying structures to support data aggregation in wireless sensor networks that performs well in terms of aggregation cost and delay. The worst-case performance of common tree structures such as MST and SPT is shown to be $\Omega(n)$ times the optimal in terms of aggregation cost.
- We propose an algorithm for constructing GIST in sensor networks, such that the cost of the tree induced by any group is within $O(\log n)$ factor of the cost of the optimal cost for that group, and delay is within $O(1)$ factor of the optimal delay. We also present a distributed protocol for constructing our GIST, performing data aggregation, and for maintaining GIST as nodes join or leave the network.

- We prove that our upper bound of $O(\log n)$ is nearly tight by presenting a lower bound of $O(\log n/\log\log n)$ for aggregation cost, for any polynomial time algorithm that approximates the problem.
- Through extensive simulations, we show that our GIST algorithm outperforms both MST and SPT by $30-60\%$ in terms of both the communication cost as well as average delay.

Our GIST protocol reduces the tree construction overhead, and each node in GIST only needs to memorize a single parent. Also, our GIST protocol has a provable $O(\log n)$ performance guarantee for aggregation cost and the delay is within a constant factor of all induced subtrees.

The remainder of this paper is organized as follows. In Section 2, we survey related work. Section 3 presents our models and basic assumptions. In Section 4, we give the formal definition of GIST and present the GIST algorithm. In Section 5, we give a distributed implementation of our GIST algorithm. In Section 6, we evaluate the performance of our GIST algorithm by simulations. In Section 7, we discuss future work and some limitations of our scheme, and propose an algorithm for constructing physical GIST on grid networks.

2 Related work

A number of data aggregation algorithms and protocols have been proposed for wireless sensor networks over the past several years. Directed diffusion [20] is proposed as a data centric communication paradigm for sensor networks. In directed diffusion, subscriptions use flooding to spread interest. Initially, the data is sent to the sink along multiple paths; however, better aggregation paths are gradually enforced. SAFE [25] uses geographically limited flooding to forward query to nodes. Due to expensive flooding operations, it is not suitable for large sensor networks. TTDD [39] exploits local flooding within a local cell of a grid to facilitate large scale data dissemination. However, when the sink moves out of the cell, the dissemination path has to be reconstructed. In [18], a regional-flooding based multicast scheme for the problem of mobicast is proposed, where high sensor network density is exploited for delivery guarantee as well as satisfying certain temporal requirements.

In [28], data-centric routing protocols were compared with traditional address-centric protocols, and the authors showed that data-centric routing offers significant performance gains in a wide range of operational scenarios. In [2, 24], the authors consider the problem of data dissemination from a source node to multiple mobile sinks. In their algorithm, a mobile sink is attached to a static sensor access point close to it, and a multicast group consists of a set of such access points that request the same information from a sensor region. Their algorithm is based on the construction of minimum Steiner trees. In [27, 31], data aggregation algorithms are designed to reduce the rounds of transmitted data from sensors to sink. In-network data aggregation has also been considered in [3, 27, 31]. We however note that none of the proposed schemes is able to achieve a provable performance guarantee.

In [21], the authors developed a polynomial time algorithm for computing a spanning tree with $O(\log^4 n/\log\log n)$ stretch on arbitrary metrics. For Euclidean metrics, an $O(\log n)$-stretch spanning tree is presented. The $O(\log n)$-stretch spanning tree construction of [21] requires the whole network to be known *a priori*, and the algorithm is centralized and not suitable for a distributed network. In addition, the algorithm only specifies the initial construction of the tree, and does not consider the maintenance of the tree in presence of node failures. Note that the scheme of dividing the Euclidean plane into recursively small regions is a commonly used technique, and has been considered in [30]. Schemes that take into account both edge cost and network diameter are also considered in [32].

Multicast communication algorithms are also considered in the context of ad hoc networks. The authors in [19] construct a publish/subscribe spanning tree across the network. A location-aware Steiner tree based algorithm is proposed in [10] for multicast in ad hoc networks. Geocasting [26], where multicast algorithms are tailored for sending information to a geographical area, is studied in [26]. In [14], hierarchical multicast routing protocols are proposed. The authors adopted an overlay-driven approach for supporting hierarchical routing.

A number of overlay multicast protocols have been proposed in the context of IP multicast. In [37], a proactive approach is considered for reconstructing overlay multicast trees. In [4], the authors consider the trade-off between path length and the load on nodes in overlay multicasting, and proposed an application-layer mininum delay multicast algorithm. Related studies identify network structures that optimize other metrics. In [8, 9], the authors proposed algorithms for constructing routing structures that minimize network congestion at nodes. In [7], the problem of routing for minimizing transmission energy under the relay model is studied.

3 Models and assumptions

3.1 Network model

Unit disk graphs. We model a wireless sensor network as an undirected unit disk graph $G = (V, E)$ on the Euclidean plane, where V is the set of nodes, and E is a set of edges on V. (We remark that our GIST algorithm can be easily extended to 3-dimensional space.) A edge (u, v) exists between $u, v \in V$ if both nodes are capable of exchanging messages over the distance $|uv|$. We assume that each node is aware of its own geographical location. This can be achieved by either a GPS device, or a location service such as the position estimator in [1, 5].

Node density. The focus of this paper is on large-scale dense sensor networks, where scalability is the main challenge for data aggregation. We assume a deployment of n sensors randomly placed in the 2-D plane, with a density of $\Omega(\log n)$ nodes per unit area, assuming unit transmission range for all nodes. It is well-known that an n-node network is disconnected (assuming unit transmission range) with high probability if the density is $O(\log n)$ [15, 33].

As discussed in Section 1, our main performance measures are the total communication cost and the average delay. The total communication cost for an aggregation is measured by total number of hops used in the aggregation, while the delay is the number of hops between the root and the farthest node being aggregated. Through standard probabilistic analysis, it has been shown that if the density is $\Omega(\log n)$ (for a suitably chosen constant hidden in the Ω notation), each cell of a sufficiently small constant area is occupied [15]. In such a dense network, the network hop distance is well-approximated by the Euclidean distance (within a small constant factor). This correlation has also been established by means of an in-depth experimental study in [18]. In our performance analysis, we adopt the Euclidean distance as our measure of the hop metric since these are equivalent up to small constant factors and the Euclidean metric is more conducive to our analysis for establishing provable bounds.

We emphasize that the correctness of our protocol does not rely on any assumptions about the sensor node density; our protocol works for *any* connected network.

Routing. Our general GIST algorithm constructs an overlay tree on the underlying physical graph $G = (V, E)$. An edge in the overlay tree corresponds to a path in the underlying graph. Therefore, an underlying routing service from a source to a destination is required. Note that the idea of using overlay trees for data dissemination and aggregation is not new and has been considered in [2, 6, 24] for wireless sensor networks and in [13, 36] in the context of ad hoc networks. Such underlying paths can be shortest paths, or paths computed by geographic routing protocols [23, 29], etc.

In Section 7, we propose an algorithm, GRID_GIST, which is not dependent on any underlying routing service. In a GRID_GIST, all edges are physical edges in the underlying graph G. In location-based services, the deployment regions can be divided into a number of small geographic areas such that two nodes in neighboring areas can communicate with each other. In each area, a leader can be selected, and a GRID_GIST can be constructed for all the leaders. Whenever a node want to participate in the data aggregation process, it first sends messages to its leader in the area, then the GRID_GIST is employed to transport and aggregate data to the sink.

3.2 Aggregation functions

In this study, we assume distributive aggregation functions [28, 31], where intermediate nodes compute and transmit one single output packet as the result of aggregating over multiple input packets. In such aggregates, the size of the partial state record is the same as the final aggregate. Thus, each transmitted packet is of the same size. Computing *maximum, minimum, average, sum, count* are examples of this class of aggregation functions. Assuming distributive aggregates, it can be easily seen that given a group S of sensor nodes and a root node (information sink), the minimum cost (total number of transmissions) data aggregation tree is a minimum Steiner tree, where the Steiner set consists of the information sink and all the data sources involved [2, 10, 24]. Therefore, the cost

of using an aggregation tree can be measured by simply counting the number of edges on the tree, since each node transmits only once.

3.3 Simulation model

In our simulation, we first evaluate our algorithm on dense sensor networks where all nodes in the network are capable of sensing, aggregating and transmitting data. We refer to this model as SNM (Sensor Network Model). We compare the performance of GIST, MST and SPT under two metrics: aggregation cost and aggregation delay. As discussed in the preceeding section, aggregation cost can be measured by the number of edges on the tree, assuming distributive aggregation functions. We measure aggregation delay by the maximum tree depth rooted at the sink node, since this represents the maximum number of hops for a packet to reach the sink. Note that delays caused by packet collision are not considered.

In addition to the above model, we consider sensor networks consisting of regular sensor nodes, as well as a dense collection of cooperative "relay" nodes [17, 34]. Relay nodes are only capable of routing and forwarding; they do not participate in data aggregation and other higher-layer sensor network applications. In [17], the authors show that given a certain energy budget, it is more efficient to deploy additional relay nodes than increase energy in existing nodes in order to extend the life time of the network. Our bound on the performance of GIST requires a dense network of the combined sensor and relay deployment. We refer to this model as SRNM (Sensor and Relay Network Model).

4 A provably near-optimal group-independent spanning tree

The optimal data aggregation tree problem can be reduced to the classic minimum Steiner tree problem, assuming distributive aggregate functions. The minimum Steiner tree problem is NP-complete [12], but can be easily approximated to within a constant factor in polynomial time, given the fixed group of vertices that need to be connected [35]. As discussed in Section 1, however, in many sensor network applications, the group of relevant sensors that needs to be aggregated may evolve constantly. Thus, it is infeasible for the resource-constrained sensor nodes to compute efficient multicast trees for the many groups (potentially, exponential in the number of sensors) on-line. Motivated by this [21], one can consider the following natural variation of the Steiner tree problem, *universal Steiner tree* problem: Given a root node $r \in V$, is there a spanning tree T connecting r to all nodes in $V - \{r\}$, such that for any subset S of V the cost of tree induced by $S + \{r\}$ on T is "close" to that of an optimal Steiner tree for the set $S + \{r\}$? If a "good" universal Steiner tree exists and can be computed efficiently, it is an excellent candidate for group-independent data aggregation, with the information sink as the root. We now present a formal definition of the universal Steiner tree problem [21].

Definition 3.1 An instance of the universal Steiner tree problem is a triple $\langle V, d, r\rangle$ where (V, d) forms a metric space, and r is a distinguished vertex in V that we refer to as the *root*. Let $||T||$ denote the cost of tree T. For any spanning tree T of V, define the *stretch* of T as $\max_{S\subseteq V} ||T_{S+\{r\}}||/||Opt_{S+\{r\}}||$, where $T_{S+\{r\}}$ denote the tree induced by $S + \{r\}$ on T and $Opt_{S+\{r\}}$ is a minimum Steiner tree for subset $S + \{r\}$. The goal is to determine a spanning tree with minimum stretch.

Two natural candidates for a universal Steiner tree are the minimum spanning tree (MST) and the shortest-paths tree (SPT). In Section 4.1, we show that both MST and SPT have $\Omega(n)$ stretch, thus making them poor choices for group-independent aggregation in the worst-case. The main focus of this section is a new algorithm for constructing GIST that achieves $O(\log n)$ stretch in Euclidean metrics, and can be efficiently implemented in wireless sensor networks.

4.1 Lower bounds for MST and SPT

MST and SPT [11] are polynomial time computable structures that are often used for optimizing the overall tree cost or individual cost of each node communicating with the root. However, when used as a GIST, MST and SPT can perform arbitrarily bad. We have the following,

Theorem 1 *The worst-case stretch of both SPT and MST in Euclidean plane is $\Omega(n)$, where n is the number of vertices in the metric.*

Due to space constraints, the proof of the above theorem is included in the full version [22].

4.2 GIST for wireless sensor networks

Our GIST algorithm will adopt the following approaches to address the issues discussed in the preceding section. First, we assume that each sensor node is aware of its geographical location. This can be achieved by adopting a location service such as the position estimators in [1, 5]. We also assume that each node also knows the location of the root node (information sink). Given such location information, our GIST algorithm divides the whole deployment region into recursively small regions (*levels*), and tree construction computation is limited to the subset of nodes in each such region. Second, our GIST algorithm constructs an *overlay* tree, i.e., an edge in this overlay tree can be a path in the original unit disk graph. Each node in the overlay tree represents a geographical region it resides in, and is referred to as a *leader* of this region. This enables GIST to be constructed without full knowledge of the network. Finally, leader election, as well as node joining/leaving the tree will be implemented on the overlay tree structure, which will also be the basis for handling communication failures in unreliable sensor networks.

The value $\sqrt{5}$ is used to ensure communication between two nodes at the opposite end of a 2×1 rectangle (consisting of two neighboring squares). Let

Algorithm 1 G$IST(v, (x_1, y_1), (x_2, y_2))$

1: **if** $|x_2 - x_1| > R/\sqrt{5}$ **then**
2: Divide the square region marked by (x_1, y_1) and (x_2, y_2) into 9 equally-sized smaller square regions (Figure 2);
3: **else**
4: return;
5: **end if**
6: Within each square region, select any node not selected before to be a leader, except for the square region where v resides (for which v shall still be the leader);
7: Output an edge between each selected leader ℓ and v;
8: For each square region, invoke G$IST(\ell, (x'_1, y'_1), (x'_2, y'_2))$, where ℓ is the leader of the square region, (x'_1, y'_1) and (x'_2, y'_2) are the coordinates marking the square region;

R be the transmission range of sensor nodes, and let D be the maximum distance between any node in $V - \{r\}$ and r. (Recall that r is the root node.) Without loss of generality, let the coordinate of r be $(0, 0)$. Algorithm 1 presents the formal definition of our GIST algorithm. It takes as input a root node and a square region specified by two diagonally-opposite locations, and computes a spanning tree connecting the root node to nodes within the square. To compute a final GIST tree, G$IST(r, (-D, -D), (D, D))$ is invoked. Our algorithm adopts a top-down approach: In each recursion, a set of new leaders representing certain geographical regions are selected and connected to their parent; each new invocation of the algorithm divides a square region into smaller pieces and repeats the same leader selection process. Note that a selected leader for a square region will also be the leader for one of the 9 smaller square regions. This applies to root node r. Let T be the GIST computed by our algorithm. For any subset $S \subseteq V$, we use *aggregation cost* to refer to the total edge distance of the tree induced by $S + \{r\}$ on T, and use *aggregation delay* to denote the length of the path from a node to the root. We have the following theorem for the aggregation delay,

Theorem 2 *(Aggregation Delay) Let T be a GIST generated by Algorithm* 1. *The distance from any node $v \in V$ to r on T is within a constant factor of the Euclidean distance between v and r.*

For the aggregation cost, we have the following.

Theorem 3 *(Aggregation Cost) Under the Euclidean metric, Algorithm 1 computes an GIST with stretch $O(\log n)$ in polynomial time.*

The proof of the above two theorems is included in the full version [22].

One interesting property of Algorithm 1 is that it eliminates dependence on knowledge of the network topology, since each node in the overlay tree represents a geographical region. Nodes can join (e.g., powered on) or leave (e.g., due to failure) the network in a dynamic fashion, while geographical regions are relatively stable. This forms the basis for our tree maintenance and fault tolerance

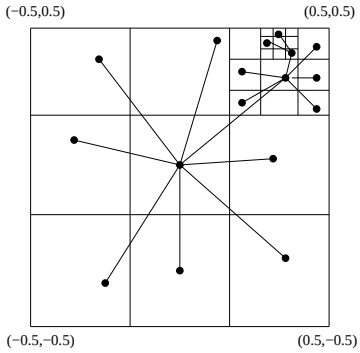

Fig. 2. Algorithm 1 divides the network into recursively small regions, and constructs an overlay tree with stretch $O(\log n)$.

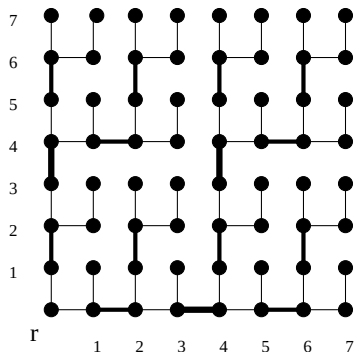

Fig. 3. Algorithm 3 construct a GRID_GIST on grid networks. The stretch is $O(\log n)$.

mechanisms (Section 5). Another property of Algorithm 1 is that the distance between any node $v \in V$ and root r on the computed GIST is within a constant factor of the minimum distance between v and r. This implies that the aggregation delay using the induced tree of GIST is within a constant factor of the minimum aggregation delay.

4.3 Lower bound for GIST

Our bound on the aggregation cost of GIST is almost tight. We can establish the following lower bound on GIST(please refer to [22] for the proof).

Theorem 4 *No algorithm can build a GIST with stretch better than* $\Omega(\frac{\lg n}{\lg \lg n})$ *for any sensor network on Euclidean plane.*

5 A GIST based data aggregation protocol

In this section, we present a distributed implementation of Algorithm 1 for constructing GIST. Our protocol proceeds in rounds by selecting leader among nodes in small regions in the first round, and electing leaders among leaders selected in the previous round in larger regions, and so on. We also define tree maintenance and fault tolerance mechanisms, as well as a brief description on data aggregation using GIST. Our protocol is a bottom-up implementation of Algorithm 1.

5.1 A distributed protocol for constructing GIST

Let (x_v, y_v) be the coordinate of $v \in V$. Without loss of generality, root node r has coordinate $(0, 0)$. Let $c = R/\sqrt{5}$ be the smallest square size (side length of

the square). As discussed in Section 4, the output of Algorithm 1 is a hierarchical overlay tree, and a selected leader ℓ of a larger square region (higher level) is also the leader for the smaller region (lower level) in which ℓ resides in. Each node running the distributed protocol maintains a variable Cur_Level indicating the largest square region for which it currently is a leader. The size of a square where v is a leader is equal to $3^{Cur\_Level-1}c$, i.e., a leader for the square with size c has $Cur\_Level = 1$, a leader for squares with size $3c$ has $Cur\_Level = 2$, and so on. By default, each node is a leader of level 0.

The protocol for constructing GIST proceeds in rounds. In each round, a leader is elected by exchanging LEADER_ELECTION packets. The leader election packet contains the following information: $[L, SQ\_X, SQ\_Y, id]$, where id is an integer uniquely identifying a node who is participating in the leader election process. The first field, L, represents the level (also the size of the square region) for which a leader is *to be* elected. The level number L, together with SQ_X, SQ_Y, defines the geographical region where a leader is elected. SQ_X and SQ_Y are integers that are defined as follows,

$$SQ\_X = \lfloor x_v/(3^{L-1} \cdot c) \rfloor, \quad SQ\_Y = \lfloor y_v/(3^{L-1} \cdot c) \rfloor,$$

where (x_v, y_v) is the coordinate of node v. Thus, a node is able to compose a packet by filling in the leader selection level, SQ_X, SQ_Y, and its id number.

Upon receiving a LEADER_ELECTION packet, a node u can decide whether it is contained in the square region defined by the packet's L, SQ_X and SQ_Y: if

$$SQ\_X \cdot 3^{L-1}c \leq x_u < (SQ\_X + 1) \cdot 3^{L-1}c,$$
$$SQ\_Y \cdot 3^{L-1}c \leq y_u < (SQ\_Y + 1) \cdot 3^{L-1}c,$$

then u is in the region; otherwise, u is not.

In each round, each node v broadcasts its own LEADER_ELECTION packet with $[Cur\_Level + 1, SQ\_X, SQ\_Y, id]$, if v has not heard of any $Cur\_Level + 1$ LEADER_ELECTION packet with lower node id, or any $Cur\_Level + 2$ or higher level LEADER_ELECTION packet. When v broadcasts a packet, it also records the level number and its own id in a local database. The root node r can fill in a negative id number in r's broadcast packet to ensure that it is elected a leader in each round. Since by default a node is a level 0 leader, the leader election starts with level 1. A node u upon receiving a LEADER_ELECTION packet pkt will invoke Algorithm 2.

The if-block of Line 4 handles the case where v actively participates in the leader selection process, since $Cur\_Level = L - 1$. The if-block of Line 10 is only for forwarding leader selection packets for higher level election, since $Cur\_Level < L - 1$ and v is aware of the fact that it cannot participate in this leader election. The if-block of Line 17 handles the case where v is already a leader for a certain region while another node is attempting leader election. Combined with Line 10 in Algorithm 2, the purpose of Line 17 is to suppress such leader election requests. However, different policies can be considered here:

Algorithm 2 A distributed protocol for constructing GIST.

1: **if** u is not contained in pkt's region **then**
2: u discards pkt;
3: **end if**
4: **if** u is contained in pkt's region, and u's Cur_Level is equal to $L-1$ **then**
5: **if** u's database has a record indexed by pkt's L and pkt's id field is lower than the record's id field, or u does not have such a record yet **then**
6: Record pkt's L and id, and re-broadcast pkt;
7: **else**
8: u discards pkt;
9: **end if**
10: **else if** u is contained in pkt's region, and u's Cur_Level is smaller than $L-1$ **then**
11: **if** u's database does not have a record indexed by pkt's L, or u has such a record and pkt's id field is lower than the record's id field **then**
12: u re-broadcast pkt and record L and id;
13: **else**
14: u discards pkt;
15: **end if**
16: **else if** u is contained in pkt's region, and u's Cur_Level is larger than $L-1$ **then**
17: u broadcasts a LEADER_ELECTION packet with $[Cur\_Level + 1, SQ\_X, SQ\_Y, id]$;
18: **end if**

e.g., an old leader can allow a new leader to be elected if it decides that its residual energy is not sufficient for reliable communication any more. The local database used for recording level number and node id helps in reducing the broadcast traffic in the network.

Each node u keeps a timer with time out value τ. When u sends out a LEADER_ELECTION packet attempting to be a level $Cur\_Level + 1$ leader, u starts the timer. When the timer fires after τ, if no better leader is detected, u increment its Cur_Level variable by 1, indicating that it is now a leader; otherwise, u picks the node id indexed by $Cur\_Level + 1$ in its database to be its parent. After a node u has picked its parent, u sends a register packet to the parent.

lgorithm 2 ensures that at least one leader is elected within the contended region in each level. Note that there are several cases in which multiple leaders may be elected for the same region and same level:

- The timeout value τ is too small for the broadcast packets to reach all destined nodes; or some broadcast packets are destroyed due to collision while they are being forwarded. The value τ is thus a adjustable protocol parameter.
- If there are multiple connected network components within the region for which a leader is to be elected, multiple leaders will be elected.

Note that the presence of multiple leaders within the same region on the same level, though it may have impact on the performance of the GIST, does not affect the correctness of Algorithm 2. This is because 1) each level i leader is able to promote itself to a level $i+1$ leader (possibly after timeout) and compete in the next round for a level $i+2$ leader, which eventually leads to root r; 2) each node will have one and only one parent, due to Line 5 of Algorithm 2.

The tie-breaking scheme in the above process is based on node ID. However, other tie-breaking schemes can easily be adopted by putting extension fields in the LEADER_ELECTION packet. For example, to elect leaders with higher residual energy, an extension field containing the residual energy reading can be used.

5.2 Data aggregation with GIST

After a GIST T is constructed, the information sink r can broadcast queries and collect data readings using T. This process is similar to most other data aggregation schemes, e.g., [31]. Due to space constraints, we only give a brief description of the query distribution and data aggregation phase, and omit detailed discussions of protocol parameters.

In the query distribution phase, the root can send a query to its direct children. Each direct child sets a timer of τ_w (a function of the *Epoch* time) for awaiting replies from its children, and includes this information in the query passing down the tree. The next level children will set their own timer (smaller than their parent) and pass down the query, and so on. The choice of timer τ_w is related to the hop distance between a child and its parent. In our scheme, the region is divided into 9 smaller regions in each recursion. Consequently, the parent-child distance decreases by a factor of 3 each time. Thus, the timeout values for the leaders along a query distribution path can be approximated by a geometric series by $\tau_w^{child} = c \cdot \tau_w^{parent}$, where $0 < c < 1$. We omit the details for selecting c here, and consider it in our future study. In the data aggregation phase, each intermediate node waits for the timeout, aggregates all the received data readings, and sends the packet to its parent.

5.3 Tree maintenance and Fault tolerance

We now consider the maintenance of a GIST in case of node failures. We assume an independent protocol for failure detection. For example, each child can check the status of its parent using periodic ping messages. If a node u concludes that its parent has failed, it sends out a LEADER_ELECTION packet with $[Cur\_Level + 1, SQ\_X, SQ\_Y, id]$ to be a level $Cur\_Level + 1$ leader. At the same time, other children of the failed parent may also broadcast such LEADER_ELECTION packets. Such operations can easily be handled by the same procedure as in Algorithm 2. This is because Algorithm 2, though presented in a synchronous fashion, is in fact an asynchronous protocol. Multiple leaders can appear in this process. However as we have discussed in preceeding sections, the correctness of the protocol is not affected. Note that if a level i

leader u failed, only the several level $i-1$ leaders in the region where u resides will initiate LEADER_ELECTION packets. Other nodes in that region only forward such packets (Line 10 in Algorithm 2). In a similar manner, the operations for new nodes joining the tree can also be handled by Algorithm 2. This way, our scheme can potentially be combined with activity scheduling protocols [6] to achieve more energy savings.

In the data aggregation phase, fault tolerance can be achieved by constructing two independent trees. Each time a node sends its data reading, it sends two copies. For fault tolerance in the data aggregation phase, we can adopt the techniques of [31].

6 Experimental evaluation

In this section, we compare the performance of our GIST algorithm with MST and SPT under the aggregation cost and aggregation delay metrics as discussed in Section 3. We simulate under two different settings: SNM (sensor network model) and SRNM (sensor relay network model). Recall that SNM models the situation where each sensor node in the network is capable of sensing, aggregating and transmitting; while in the SRNM model, the relay node is only capable of forwarding traffic.

6.1 SNM performance evaluation

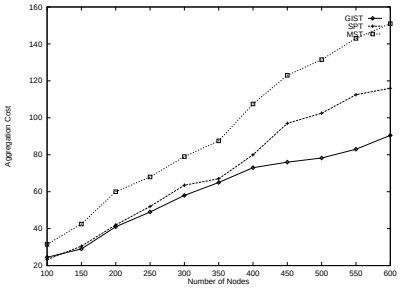

Fig. 4. Aggregation cost with increasing network size under SNM model. Event sources are picked within a strip.

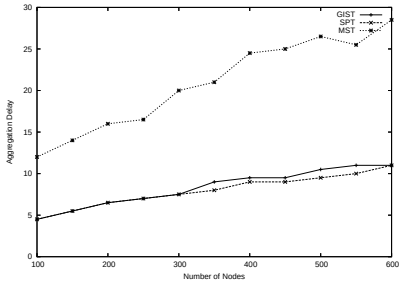

Fig. 5. Aggregation delay with increasing network size under SNM model. Event sources are picked within a strip.

In our simulations, the network topology is generated by randomly distributing nodes in the deployment region. The density (average number of neighbors within a node's transmission range), is around 10 in each experiment. In our first experiments, we study the aggregation cost when the event sources are picked

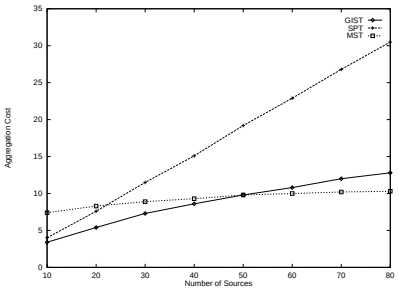

Fig. 6. Aggregation cost with increasing number of random sources under the SRNM model. 200 sensor nodes.
.

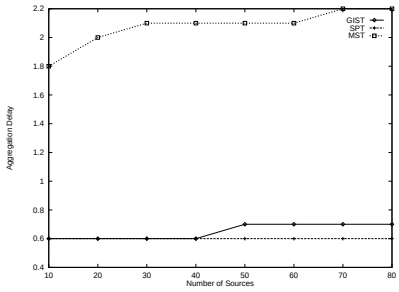

Fig. 7. Aggregation delay with increasing number of random sources under the SRNM model. 200 sensor nodes.

within a strip close to the border of the sensor deployment region. This is useful in applications where the sensor nodes observing common phenomena are along a line, for example when a group of intruders approaching the border in a battle field trigger the sensors. Figure 4 demonstrates the result. In this simulation, GIST performs better than SPT, which is better than MST. The reason that GIST is better than SPT is because when the event sources are picked along a line strip, a large number of event sources may traverse non-overlapping paths to the sink, thus reducing the chance for data aggregation in intermediate nodes. Figure 5 illustrates the aggregation delay in this experiment, in which GIST demonstrates close to optimal (SPT) delay.

6.2 SRNM performance evaluation

In this section, we evaluate the performance of GIST, MST and SPT under the SRNM model, where the network consists of sensor nodes as well as relay nodes. We simulate a network that consists of 200 sensor nodes, and a dense network of relay nodes. The density is such that the probability that a underlying routing path between two sensor nodes containing another sensor node is small. This case can be thought of as the opposite of the SNM model, where the relay density is 0. In this model, the SPT overlay tree from the sensor nodes to the sink exhibits a star-like structure. Therefore, the aggregation in intermediate sensor nodes is minimal.

Figure 6 and Figure 7 illustrate the aggregation cost and delay in the experiments with $10, 20, \ldots$, or 80 random event sources. Note that the values in the figures are calculated by normalizing the network deployment region to a 1×1 region. The aggregation cost (induced tree cost) of SPT is the worst among the 3 algorithms as expected, due to the lack of aggregation. MST and GIST both have much better performance than SPT. When the number of sources is small, an induced aggregation tree of MST costs more than that of GIST, because some event sources may traverse long routes to the root, as can be observed in the

aggregation delay in Figure 7. When the number of random sources increases, such long paths may contain more and more other sources so that aggregation can be performed.

7 Conclusion and future work

In this paper, we proposed a novel approach to data aggregation based on the concept of group-independent spanning tree GIST, and show that such a tree can be found in polynomial time with $O(\log n)$ performance guarantee. Specifically, we have designed an algorithm for constructing an GIST for dense sensor networks such that for any group, the cost of the induced subtree of our GIST is within a logarithmic factor of the optimal solution for the group. We have also shown that traditional spanning tree algorithms MST and SPT are extremely poor in the worst case. An important aspect of our GIST algorithm is its simplicity and amenability to distributed implementation. We have presented a protocol for constructing and maintaining GIST and for performing data aggregation over the tree.

Our algorithm for constructing GIST yields an overlay tree and assumes an underlying routing mechanism. This assumption can limit the application of our protocol. We propose an algorithm ([22]) for constructing GRID_GIST for grid sensor networks that does not require an underlying routing service. In this algorithm, each edge selected into the physical GRID_GIST tree has to be an existing edge in grid networks. Routing from any node in G to the root is specified during tree contruction phase, thus eliminating the dependence on an underlying routing service as required by GIST. Such physical trees on grids can be useful for densely deployed sensor networks. Due to space constraints, we present our algorithm GRID_GIST in the full version [22]. One limitation to the GRID_GIST scheme is that it assumes regularity of the network. An important direction for future research is to determine the best "physical" GIST for arbitrary network topologies.

In our simulations, we have studied two variants of the sensor network model, one consisting purely of sensor nodes, and the other consisting of sensors placed in a dense relay field. Our simulations validate our theoretical work by demonstrating that our GIST outperforms both MST and SPT. As part of our future work, we will compare our scheme with other data aggregation schems in this literature in addition to MST and SPT.

References

1. J. Albowicz, A. Chen, and L. Zhang. Recursive position estimation in sensor networks. In *IEEE ICNP*, November 2001.
2. S. Bhattacharya, H. Kim, S. Prabh, and T. Abdelzaher. Energy-conserving data placement and asynchronous multicast in wireless sensor networks. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, May 2003.

3. B. Bonfils and P. Bonnet. Adaptive and decentralized operator placement for in-network query processing. In *Proceedings of Information Processing in Sensor Networks*, April 2003.
4. E. Brosh and Yuval Shavitt. Approximation and heuristic algorithms for minimum delay application-layer multicast trees. In *INFOCOM*, March 2004.
5. N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low cost outdoor localization for very small devices. In *IEEE Personal Communications, Special Issue on Smart Space and Environments*, October 2000.
6. U. Cetintemel, A. Flinders, and Y. Sun. Power-efficient data dissemination in wireless sensor networks. In *ACM MobiDE*, September 2003.
7. J. Chen, L. Jia, X. Liu, G. Noubir, and R. Sundaram. Minimum energy accumulative routing in wireless networks. In *In Proceedings of IEEE INFOCOM 2005, The 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, 2005.
8. J. Chen, R. Kleinberg, L. Lovász, R. Rajaraman, R. Sundaram, and A. Vetta. (Almost) tight bounds and existence theorems for confluent flows. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 529–538, June 2004.
9. J. Chen, R. Rajaraman, and R. Sundaram. Meet and merge: Approximation algorithms for confluent flows. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 373–382, June 2003. This paper has been accepted for publication in the special issue of the Journal of Computer and System Sciences (JCSS).
10. K. Chen and K. Nahrstedt. Effective location-guided tree construction algorithm for small group multicast in manet. In *IEEE INFOCOM*, June 2002.
11. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.
12. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.
13. C. Gui and P. Mohapatra. Efficient overlay multicast for mobile ad hoc networks. In *IEEE WCNC*, April 2003.
14. C. Gui and P. Mohapatra. Scalable multicasting in mobile ad hoc networks. In *INFOCOM*, March 2004.
15. P. Gupta and P. Kumar. Capacity of wireless networks. *IEEE Transactions on Information Theory*, IT-46:388–404, 2000.
16. J. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Towards sophisticated sensing with queries. In *Intl Workshop on Information Processing in Sensor Networks (IPSN)*, 2003.
17. Y.T. Hou, Y. Shi, H. Sherali, and S. Midkiff. On energy provisioning and relay node placement for wireless sensor networks. *IEEE Transactions on Wireless Communications*, 4, 2005.
18. Q. Huang, C. Lu, and R. Gruia-Catalin. Spatiotemporal multicast in sensor networks. In *SenSys*, November 2003.
19. Y. Huang and H. Garcia-Molina. Publish/subscribe tree construction in wireless ad-hoc networks. In *Proceedings of the 4th International Conference on Mobile Data Management*, January 2003.
20. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *ACM MobiCom*, August 2000.

21. L. Jia, G. Lin, G. Noubir, R. Rajaraman, and R. Sundaram. Universal approximations for TSP, Steiner tree, and set cover. In *Proceedings of the Thirty-Seventh ACM Symposium on Theory of Computing (STOC)*, May 2005.
22. L. Jia, G. Noubir, R. Rajaraman, and R. Sundaram. Gist: Group-independent spanning tree for data aggregation in dense sensor networks. Technical report, Northeastern University, May 2006.
23. B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of ACM Symposium on Mobile Computing and Networking*, pages 243–254, August 2000.
24. H. Kim, T. Abdelzaher, and W. Kwon. Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks. In *ACM SenSys*, November 2003.
25. S. Kim, S.H. Son, J.A. Stankovic, S. Li, and Y. Choi. Safe: A data dissemination protocol for periodic updates in sensor networks. In *Workshop on Data Distribution for Real-Time Systems (DDRTS)*, May 2003.
26. Y.B. Ko and N.H. Vaidya. Geocasting in mobile ad hoc networks: Location-based multicast algorithms. In *WMCSA*, Feburary 1999.
27. B. Krishnamachari, d. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems Worshops*, July 2002.
28. B. Krishnamachari, D. Estrin, and S. Wicker. Modelling data-centric routing in wireless sensor networks. In *IEEE INFOCOM*, June 2002.
29. F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *ACM Mobihoc*, June 2003.
30. J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, pages 120–130, August 2000.
31. S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad hoc sensor networks. In *OSDI*, December 2002.
32. Madhav V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, Daniel J. Rosenkrantz, and Harry B. Hunt III. Bicriteria network design problems. In *Automata, Languages and Programming*, pages 487–498, 1995.
33. M. Penrose. On fk-connectivity for a geometric random graph. *Random Structures and Algorithms*, 15:145–164, 1999.
34. B. Sirkeci-Mergen and A. Scaglione. A continuum approach to dense wireless networks with cooperation. In *INFOCOM*, March 2005.
35. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2003.
36. J. Xie and R. Talpade. AMRoute: Ad hoc multicast routing protocol. *ACM Mobile Networks and Applications*, 7, December 2002.
37. M. Yang and Z. Fei. A proactive approach to reconstructing overlay multicast trees. In *INFOCOM*, March 2004.
38. Y. Yao and J. Gehrke. The Cougar approach to in-network query processing in sensor networks. *Sigmod Record*, 31(3), September 2002.
39. F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *MOBICOM*, September 2002.