Landmark-Based Information Storage and Retrieval in Sensor Networks

Qing Fang* Jie Gao† Leonidas J. Guibas‡

* Department of Electrical Engineering, Stanford University. jqfang@stanford.edu
† Department of Computer Science, Stony Brook University. jgao@cs.sunysb.edu
‡ Department of Computer Science, Stanford University. guibas@cs.stanford.edu

***Abstract*— For a wide variety of sensor network environments, location information is unavailable or expensive to obtain. We propose a location-free, lightweight, distributed, and data-centric storage/retrieval scheme for information producers and information consumers in sensor networks. Our scheme is built upon the Gradient Landmark-Based Distributed Routing protocol (GLIDER) [8], a two-level routing scheme where sensor nodes are partitioned into tiles by their graph distances to a small set of local landmarks so that localized and efficient routing can be achieved inside and across tiles. Our information storage and retrieval scheme uses two ideas on top of the GLIDER hierarchy — a distributed hash table on the combinatorial tile adjacency graph and a double-ruling scheme within each tile. Queries follow a path that will provably reach the data replicated by the producer(s). We show that this scheme compares favorably with previously proposed schemes, such as Geographic Hash Tables (GHT), providing comparable data storage performance and better locality-aware data retrieval performance. More importantly, this scheme uses no geographic information, makes few assumptions on the network model, and achieves better load balancing and structured data processing and aggregation *even* for sensor fields with complex geometric shapes and non-trivial topology.**

I. INTRODUCTION

Making effective use of the vast amount of data gathered by large-scale sensor networks requires scalable and energy-efficient data storage and data dissemination algorithms. Queries on sensor networks may be content-based, in that users are primarily interested in data satisfying certain attributes, not in the details of which node currently contains the data. An information producer generates data that may be of interest to multiple information consumers located in other parts of the network at a much later time. This separation of communication in space and time calls for an information brokerage scheme for sensornets. An *information brokerage* scheme is a mechanism that carries out data publication, data replication for the information producers (a.k.a. *producers*) and data retrieval for the information consumers (a.k.a. *consumers*). In the sensor network setting, we formulate it as a mechanism to enable a network of nodes to self-organize and store the sensed data, cooperate to route and answer the query.

The utility of a sensor network derives primarily from the data it gathers. Previous work has addressed data-centric routing [1], [10], [11], [15] and data-centric storage [16] as efficient data management schemes for sensor networks.

In data-centric routing, low-level communications are based on names that are external to the network topology and relevant to the applications. A typical data-centric routing protocol, directed diffusion [11], uses a flooding algorithm to distribute interests to match with data obtained at source nodes. Matched data are delivered back to the sink (consumer) on reversed paths, the best of which will be reinforced for continuing future use. Little collaborative preprocessing is performed on the data gathered by the sensors in such schemes. Thus the discovery of the desired information has to rely on flooding the network.

In contrast, in data-centric storage data generated by sensors is preprocessed and stored in a distributed fashion. A simple yet powerful idea, the Distributed Hash Table (DHT), was proposed to match data from the producers and interests from the consumers. In DHTs, data is named and stored via its *content*. Specifically, a piece of data is hashed to a node based on its content, using a common hash function known to the producers and consumers. That node stores the data of the producers and acts as a reservoir from which the consumers can retrieve its desired data assuming proper routing service is provided. DHTs avoid expensive network-wide flooding and can achieve good storage load balancing by using hash functions that evenly distribute load across the network.

DHTs were originally proposed for distributed data storage on the Internet. The idea has been adopted in the sensor network context. One example is the Geographic Hash Table (GHT) [16] which combines the DHT idea with geographic naming and routing. It uses geographic locations as reservoirs where data is hashed to and retrieved from, and uses a geographic routing approach, GPSR [12], as the underlying routing scheme. However, since GHT uses location-based naming and routing schemes, it inherits all the disadvantages of them as well: location-based routing requires that the *accurate* location information about the sensors be available; it requires that the communication network follow the unrealistic unit disk graph model; it also assumes that a planar subgraph of the communication graph with the same connectivity is correctly constructed; furthermore, it performs poorly when the sensor networks have complex geometric shapes and non-trivial topologies, e.g., have many holes (caused by obstacles, events, or clusters of failed nodes) [7], etc. Nodes on the boundaries of holes are used more often due to the extensive use of perimeter forwarding when packets bypass holes and subsequently depleted nodes enlarge the holes. This counteracts the load balancing argument for DHT.

DHTs have inefficiencies that are not of major concern in the Internet environment but become significant drawbacks for its application in sensor networks. First, DHT uses a hash function to map a piece of data to a randomly determined node. The property that contents are scattered in the network is good for load balancing, but bad for structured data organization and subsequently bad for queries that require cross-type data aggregations. Second, DHTs are not locality aware. An information consumer may have to travel far to a hashed location to fetch a piece of data even if the data is actually generated from an information producer nearby. Hierarchical hashing [13] has been proposed to improve the locality of the flat hash table. But the construction of this hierarchy complicates the overall design and the asymptotic bounds on storage and retrieval costs have hidden large constants that do matter in practice. It has been experimentally observed that such a scheme only outperforms flat hashing on networks with extremely large sizes [5].

Another approach to information storage and retrieval that gets around the problems suffered by DHTs is what we call *double-ruling*. Double-ruling works as follows: data is stored not at a single node or its nearby neighbors, but at nodes that follow a one-dimensional curve while a data request travels along a set of nodes that follow another one-dimensional curve. The curves are functions of only the locations of the producer and the consumer respectively, and not of the types of data that is stored or sought. Therefore, successful retrieval is guaranteed if every retrieval curve intersects every data storage curve. For an easy case, assume the network is a two-dimensional grid embedded in the plane with nodes located at all the lattice points (see Figure 1). The information storage curves follow the horizontal lines. The information retrieval curves follow the vertical lines. Each vertical line intersects each horizontal line, thus an information consumer can always find the data produced by the producer. In fact, by travelling along a vertical line, an information consumer finds *all* the data stored in the network. This double-ruling scheme is locality aware — if the producer and consumer are actually near each other, they must also be near each other along the path connecting them using the horizontal and vertical lines. Moreover, it has better fault tolerance by replicating data on nodes that are uncorrelated with node proximity.

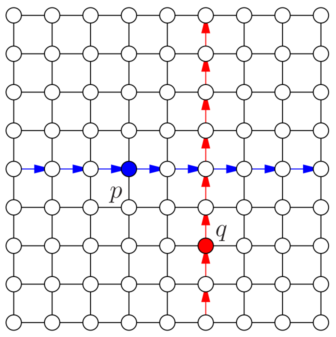


Fig. 1. A nice and simple double ruling scheme on a grid network. The information producer p replicates its data along a horizontal line. The information producer q queries along a vertical line. Data retrieval is guaranteed since every vertical line intersects every horizontal line. This scheme also reflects locality since nearby sensors are also nearby using the L_1 distance.

Despite all these good properties, the double-ruling idea is hard to realize on arbitrary communciation graphs. This is mainly due to the rich geometric flavor of the double-ruling scheme. So far research along this line has either focused on networks with nice graph structures, e.g., those that resemble grids [14], [18], [17] whereby two sets of curves are constructed using geometric information, or depends on heavy data replication to guarantee successful retrieval. An example of the latter is rumor routing [3]. In rumor routing, data are replicated along a set of straight line gradients from the information producer, while a query is sent along a random straight line from the consumer. However, this requires a substantial number of data replication lines to guarantee with high probability that the query line will meet with one of the event lines within the sensor field. In general, in a sensor field with uneven sensor distribution and irregular geometric shapes, it is unclear how to construct a good double-ruling scheme where the data replication paths of different producers and the data retrieval paths of different consumers are not too long each (not too many replications) and are evenly spread out across the network. The problem becomes even harder when geometric coordinates of each node in the network are unknown.

II. Overview

For a class of sensornet applications, multiple producers and consumers of the same data may appear at any locations within the network. One such example is real-time environmental monitoring in a national park. For example, a canonical query a tourist may ask: "where are the giraffes located at the present time?" From the consumers' (the tourists' in this example) point of view, it is desirable to have the giraffe information readily stored at some nearby nodes, which means the same piece of information have to be replicated sufficiently frequently within the network. This increases the cost for information producers. From the producers' point of view, it is most cost effective for them to store data locally, in which case, the consumers have to flood the network to get its query answered.

We are therefore interested in investigating a scheme that: first, balances the cost for the producers and consumers; second, allows the possibility of data aggregation whenever possible to reduce transmission cost (which can benefit real-time monitoring applications); third, is locality aware, meaning when a consumer is close to the producer, the consumer should not travel far before it retrieves the information; fourth, achieves the above goals without requiring nodes' geographic information as in GHT or other double-ruling solutions.

In this paper, we present a two-level data-centric storage and retrieval scheme that integrates the distributed hash table idea and the double-ruling idea, using a naming and routing infrastructure, the Gradient Landmark-based Distributed Routing protocol (GLIDER) proposed by Fang *et al.* [8].

GLIDER is a location-free naming and routing scheme that discovers and abstracts the global topology of the sensor field in a preprocessing stage using landmarks (a well-chosen subset of sensors). It partitions the nodes into routable tiles — regions where the node placement is sufficiently dense and regular so that local greedy methods can work well. Such global topology includes not only connectivity but also higher order topological features, such as the presence of holes, for example, the empty circular region in Figure 2. These tiles are glued in a way represented by the tile adjacency graph whose edges are shown as the line segments in the figure.

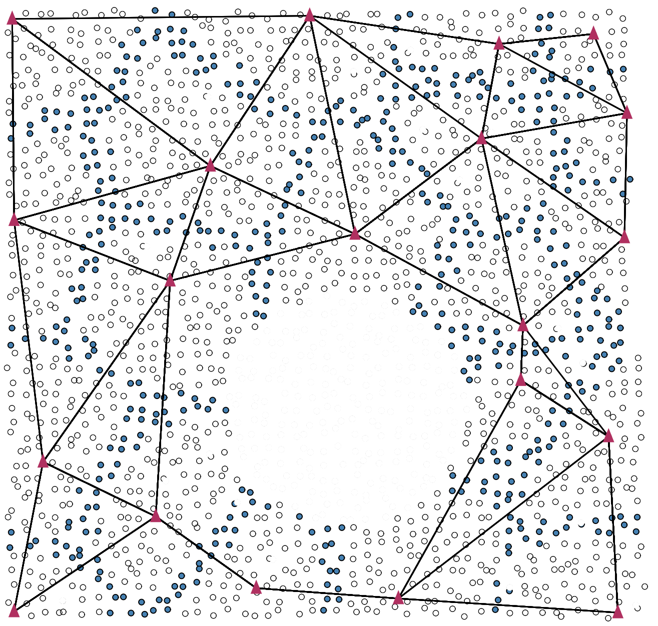


Fig. 2. The tile partitioning in GLIDER by using a number of landmarks. Landmarks are shown by triangles. Sensor nodes are shown as small circles. The nodes are divided into tiles. The dark nodes are the boundaries of the tiles. A combinatorial graph on the set of landmarks is drawn to represent the tile adjacency relationship. Namely, a pair of landmarks are connected by an edge if their corresponding tiles are adjacent.

Conceptually our scheme has two levels: a distributed hash table for information storage at the tile adjacency graph level; and a double-ruling scheme at the lower level (inside each tile), which ensures information retrieval within each tile.

We hash a piece of data to a tile (i.e. the hashed tile) that is determined by the content of data. Based on the tile adjacency graph, the shortest path tree rooted at the hashed tile can be computed at each node. All producers and consumers of the same content proceed to the hashed tile following this common shortest path tree. For a producer, data is replicated inside all the tiles (not all the nodes) along the way from where the producer resides to the hashed tile (which we call the *replication path*). The information consumer proceeds towards the hashed tile and checks each tile on its way for the desired data (which we call the *retrieval path*), returns when the retrieval path meets the replication path.

A. Hashing on the tiles

In the traditional DHT, an information consumer q may have to travel to a hashed location far away, even when the information producer p is actually nearby. However, notice that the CDG usually has low degree (a small constant most of the time). Since the sensors usually are embedded in the Euclidean plane, for a specific landmark x, the number of landmarks with exactly h hops from x on CDG is usually of order $O(h)$. If the information producer and consumer are geographically close, it is likely that their shortest paths to the same hashed tile will merge before reaching the common root. For the information consumer in the same tile as the producer, the consumer will find the data inside its resident tile, without the need to travel to the hashed tile which can be far away. Therefore, our information brokerage scheme is locality aware — an information consumer can retrieve data more quickly without going to the hashed tile if the consumer is actually close to the 'data reservoir' of interest. This is also confirmed through simulations.

Furthermore, routing in sensor network does not generally follow the Internet's end-to-end principle. It is desirable that a node receiving a packet can inspect the data and make a local decision on whether to continue passing along the packet or transform the data or drop the packet. The fact that the producers send their data following the shortest-path-tree on the tile adjacency graph creates opportunities for the data replication paths of different producers but of the same data category to cross before reaching the hashed tile. This creates additional opportunities for en-route data aggregation for the producers and hence contributes to overall system efficiency.

Our scheme also provides better support for structured data storage and query. This built-in correlated data storage allows the possibility for the consumers to fetch all the data of the same category using only a single retrieval path inside the hashed tile and ease of aggregation on data of the same category. More detailed discussion is deferred to subsection IV-D.

B. Double-rulings within each tile

Data replication and retrieval inside a tile are implemented by a double-ruling scheme. Specifically, for a producer, its data is replicated at nodes along three curves organized in a tree. These three curves are formed by the producer following the shortest paths towards three neighboring landmarks, called *guides*, starting from a random node. To save storage, we store data on only one or a small number of sensors on the curves, and store at the other nodes only pointers to where data is replicated. Data retrieval follows either a zigzag curve that visits the tile's boundary with each guide, or simply a shortest path towards the next landmark on the globally planned combinatorial path. It is guaranteed by the properties of the tile partitioning that the data retrieval path actually meets with each data replication tree. Figure 3 gives an example of an information producer and consumer pair.

In general, designing a good double-ruling scheme requires detailed understanding of the geometry of the sensor layout. However, the tile adjacency graph provided by GLIDER captures spatial adjacency information of a landmark with respect to the other landmarks, by exploiting which we can design our double-ruling scheme without the sensors' geographic location information, and in a locally controlled way. This extends the current double ruling ideas on regular grid topologies to arbitrary topologies.

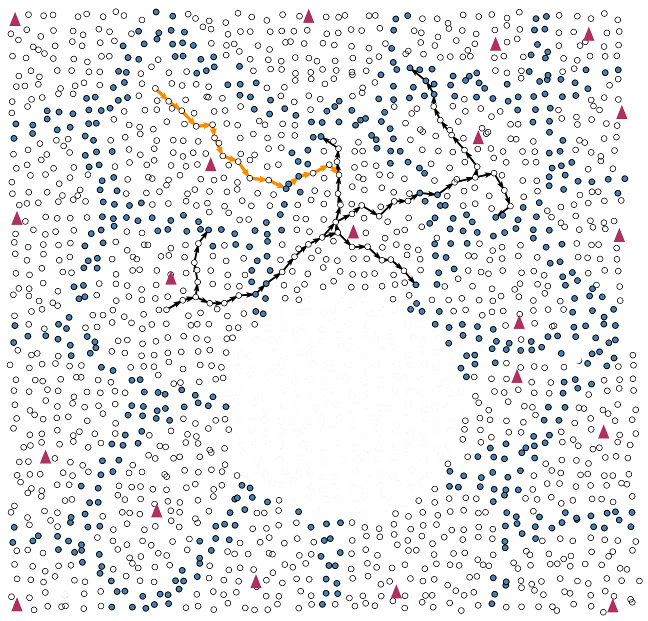


Fig. 3. This figure shows the routes of a pair of information producer and consumer. The darker paths are the replication routes followed by the information producer. The lighter path is the path that the query travels to retrieve the desired data. In this example, the information consumer finds (where the two path cross) the desired data before it reaches the hashed tile (located in the upper right end of the path).

III. Glider naming and routing infrastructure

Glider is a landmark-based location-free naming and routing scheme for sensor networks. It computes in a preprocessing stage the landmark Voronoi complex (LVC), and its dual, the combinatorial Delaunay Graph (CDG) that captures the global topology of the sensor field. Voronoi diagram and Delaunay triangulation were originally defined on points in Euclidean space [6]. Recently they have been extended to a graph setting [4], [8]. Formally, for a connected unweighted graph $G = (V, E)$ and a subset of landmarks $L \subset V$, define the Voronoi cell $T(v)$ of a landmark $v \in L$ to be the set of nodes whose nearest landmark measured in the minimum number of hops on G is v (ties are allowed). See Figure 2 as an example. The combinatorial Delaunay graph is the dual graph of the complex Voronoi diagram that represents the adjacency relationship of the Voronoi cells. There is an edge between two landmarks v_1, v_2 if there is either a node w in both Voronoi cells $T(v_1), T(v_2)$, or two nodes w_1, w_2 in each Voronoi cell ($w_1 \in T(v_1), w_2 \in T(v_2)$) that are neighbors on G. Such nodes w_i are referred to as *witnesses* to the edge v_1v_2. The collection of witnesses to the edge v_1v_2 is called the *Voronoi boundary* of v_1, v_2. The combinatorial Delaunay graph $D(L)$ can be thought of an abstract of the original graph G. It was proved in [8] that $D(L)$ is connected if G is connected. Further, every path in G can be lifted to a path in $D(L)$. Conversely, every path in $D(L)$ can be realized as a path in G. Under reasonable conditions (a dense distribution of nodes, reasonably simple topology, a good selection of well-separated landmarks), the Voronoi cells of a set of landmarks provide a natural partitioning of the sensor field into connected tiles with trivial topology in all dimensions.

The CDG, as an abstraction of the global topology of the sensor field, usually has a size proportional to the number of large topological features of the sensor field (such as holes), which is much smaller than the total number of sensors. Increasing the total number of sensors without increasing the number of topological features will increase the tile size given the same number of landmarks. While tile size may affect individual routes, it does not affect the overall route distribution and the success of Glider. If the sensor distribution is dense and uniform, three landmarks are sufficient for routing purposes. How exactly the number and distribution of the landmarks should be determined is an active ongoing research project. In our simulations, 23 landmarks (randomly selected plus a few on sensor field boundaries) are used out of 2000 nodes, which is shown through experimental studies to have good performance in terms of route length and load balancing. In addition, the CDG is also robust to network dynamics. Unless a large portion of geographically correlated sensors die, the global topology, thus the CDG, does not change. Therefore the CDG is a *compact*, *stable* and *descriptive* abstraction of the *global topological information* of the sensor field, and can be made available to every sensor in the network.

In Glider, each sensor p is given a name with respect to a subset of landmarks. For a sensor p, its closest landmark is denoted as p's *home landmark*, denoted as $h(p)$. The Voronoi tile of p's home landmark is denoted as p's *resident tile*. For the nodes in the Voronoi cell $T(v)$ of a landmark v, the *reference landmarks* are v itself and the neighbors of v in the CDG $D(L)$. Each sensor node knows the minimum hop counts to its local reference landmarks. The name of a node is then expressed by a vector based on hop count distances to its reference landmarks. Effectively, node names are thereby determined by global topology of the field layout and local connectivity. Such an approach to naming in Glider allows three kinds of efficiencies.

- Global network names for nodes can be constructed *locally* after the initial topology discovery phase. Global naming is important to support context-based storage that requires an unambiguous node or region of the network given a network address and any-to-any routing with low stretch factors.
- Mapped references (the landmarks) can be bound to node names, enabling node names with a meaning relevant to the applications. For example, in a building monitoring scenario, a valid node name might be: 2 hops from the first-floor printer room, 5 hops to the building exit, 4 hops from the stairs, etc. This naming scheme not only gives nodes addresses, but also relations to well-known reference points that can be useful in designing locality-aware services for a range of sensor network applications.
- The overhead of communication required for resolving name bindings is eliminated, such as the service of DNS lookups for the Internet or location services for geographic routing.

Once the CDG is constructed, every sensor knows its own name. Notice that the preprocessing stage of using landmarks to discover the global topology is done only once for the entire lifetime of the sensor network. After the preprocessing stage, landmarks are just virtual references and take the same role as other sensors in the routing protocol. With the help of this proactive infrastructure, routing can be performed in a reactive manner.

Intra-tile routing is done by gradient descent using the local virtual coordinates. Namely, the next hop is selected as the one whose virtual distance to the destination is the smallest.

Inter-tile routing between two sequential tiles T_i and T_{i+1} is implemented by following the shortest path toward landmark u_{i+1} by neighborhood distance, which is defined as follows. For each landmark $u \in L$ and its adjacent landmark v on $D(L)$, we define the *neighborhood distance* from a point $p \in T(u)$ to v as the number of hops from p to v measured in the subgraph spanned by $T(u) \bigcup T(v)^1$. By the construction of the landmark Voronoi complex, any $p \in T_i$ has the neighborhood distance to all its reference landmarks, including u_{i+1}. Thus inter-tile routing between T_i and T_{i+1} sets u_{i+1} as the temporary destination. The packet is forwarded to a node which decreases the distance to u_{i+1}. Eventually the packet reaches the boundary of tile T_{i+1}, at which point we switch the temporary destination to u_{i+2} (the home landmark of T_{i+2}, the tile following T_{i+1}); and so on until the packet reaches the destination tile. See Figure 4 for an example.

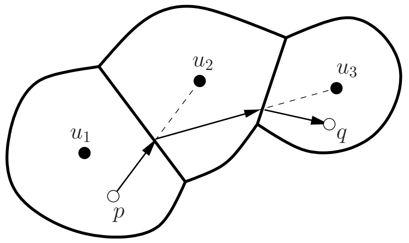


Fig. 4. Routing across tiles in GLIDER.

The shortest paths of a node to its reference landmarks are used for efficient inter-tile routing. In the presence of node or link failures these shortest paths may be broken. However, unless a large number of nodes in the same region die together, efficient routes to reference landmarks can be preserved. Each node stores the IDs of all of its neighbors that are i hops away from a reference landmark if this node itself is $i+1$ hops away from the same reference landmark. In case that all the parents of a node die, a local flooding may be needed to get out of the stuck node.

The novelty of the GLIDER naming and routing scheme is that it explicitly captures the global topology of the sensor field, based on which global route planning and local greedy routing can be smoothly integrated. As the global topology of the sensor field captures where holes are and how to route around them, GLIDER gets around many of the pitfalls of location-based routing schemes, such as the requirement of accurate location information, the unit-disk graph model and the correct construction of a planar subgraph of the link connectivity graph to get around local minima in greedy routing. It was also observed in [8], that the topology-based routing achieves better local balancing compared with geographic routing, such as GPSR and GFG [12], [2], in that sensors near hole boundaries are not as severely over-loaded.

^{1}The definition for neighborhood distance is slightly different from the one defined in [8]. We made this modification to guarantee that the shortest path from $p \in T(u)$ to one neighboring landmark v will cross the Voronoi boundary between u and v.

IV. LANDMARK-BASED DATA-CENTRIC STORAGE AND RETRIEVAL

Utilizing the two level structure adopted in GLIDER, our information storage and retrieval scheme works as follows: at the tile adjacency graph (i.e. CDG) level, we use a modified content-based distributed hash table on the CDG; within each tile, we use the double-ruling idea. We explain the algorithm in detail in the following subsections.

A. Content-based distributed hashing on CDG

For a piece of data produced by the information producer, a hash function f takes the content of the data as input and outputs a landmark node ID, i.e., $f : \mathcal{C} \mapsto L$, with $\mathcal{C}$ being the set of possible values of a specific attribute of the information content and L the set of landmark IDs. For example, $\mathcal{C}$ may be the name space for different species of animals, in which case, different event records concerning the same species will all be hashed to the same tile in the network, no matter where the data was originated. Every sensor stores the hash function f, and therefore can compute the hash result locally. On the other hand, the CDG, usually of a small size, is also known to every sensor in the network. Given the CDG, each node can compute locally the shortest path tree on the CDG rooted at any landmark (there might be multiple shortest path trees rooted at the same landmark — ties are broken arbitrarily). An information producer, located at sensor p, proceeds from its resident tile towards the hashed landmark $f(c)$. In each Voronoi tile along the (abstract) path from its resident tile to the hashed tile, data is replicated at a subset of sensors following a set of rules to be described in the next section. These tiles serve as 'data reservoirs' to store the data produced by the (possibly many) producers.

B. Data replication (for producers) and retrieval (for consumers) within a Voronoi tile

This section describes in details the rules for data replication and retrieval within a Voronoi tile, with the help of inter-tile routing provided by the GLIDER.

1) Data replication for information producers: For an information producer at sensor x in a tile with home landmark u, we first find on the CDG the shortest path from u to the landmark $f(c)$ with the hashed data c. Suppose this shortest path passes through a sequence of tiles $\mathcal{P}(u, f(c)) = S_1, S_2, \cdots, S_k$, where $S_i = T(u_i)$, with u_i as the home landmark of S_i, $u_1 = u$, $u_k = f(c)$. Routing from the first tile S_1 to the last tile S_k is implemented by inter-tile routing in GLIDER. Furthermore, data is replicated at each tile on the way of $\mathcal{P}(u, f(c))$.

The basic idea of data replication inside a Voronoi cell with landmark u is to start from a (randomly selected) sensor p in tile $T(u)$ and replicate data along a path from p to each of the three *guides* r_1, r_2, r_3, until the boundary of $T(u)$ is hit. The collection of paths to three guides is organized by a *finger tree*, denoted as $\mathcal{F}(p)$. The selection of the guides is described in the following.

- **Replication rule A**: If u has at least three neighboring landmarks on the CDG, the three guides are chosen from u's neighboring landmark set. Such selection of guides is fixed with respect to the same destination landmark $f(c)$. If the next Voronoi cell on the path $\mathcal{P}(x, f(c))$ is landmark v. Then we pick v as the first guide r_1. The other two guides can be chosen, e.g., as the neighboring landmarks of both u and v, as shown in Figure 5.
- **Replication rule B**: For the rare case when u has two or less neighboring landmarks on the CDG, u has at least one neighbor v which is chosen as the first guide v_1. The other two guides are picked on the boundary between $T(u)$ and $T(v)$ in such a way that they are distinct and they are not on the shortest path from p to v in the subgraph spanned by the Voronoi tiles of u, v, i.e., $T(u) \bigcup T(v)$.

Notice that the selection of guides for a Voronoi cell $T(u)$ only depends on u and the destination cell $T\big(f(c)\big)$. Thus it is fixed and known to all the sensors in $T(u)$. Data replication is only done inside $T(u)$, on the paths from p to the three guides.

- If the guide r_i is a neighboring landmark of u, by the construction of GLIDER, each node in $T(u)$ has the hop count to r_i in the Voronoi neighborhood graph spanned on $T(u) \bigcup T(r_i)$. We replicate data on the shortest path from p to r_i in the neighborhood distance metric, namely, the shortest path in the subgraph spanned on nodes $T(u) \bigcup T(r_i)$, until a boundary node of $T(u)$ is reached. See Figure 5.
- If the guide r_i is a node on the boundary of $T(u)$, we replicate data on the path routed by the GLIDER intra-tile routing scheme from p to r_i.

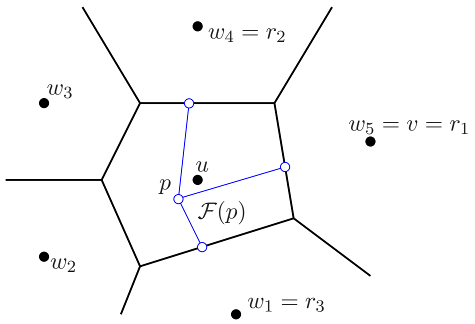


Fig. 5. Data is replicated on a finger tree $\mathcal{F}(p)$ rooted at p towards three guides r_i, $i = 1, 2, 3$.

In summary, inside a tile with home landmark u, we replicate data on a finger tree rooted at a random node p. The finger tree is only determined by u and the content of data. Each path on the finger tree is produced by either inter-tile or intra-tile routing in GLIDER.

Data replication on the finger tree inside tile $T(u)$ can be performed in various ways. In general there is a tradeoff between the storage cost and the query cost. If the data has a small size, then we can replicate the whole piece of data on every sensor on $\mathcal{F}(p)$ such that a query gets the data when it hits any node on the finger tree. To save node storage, on the trail of $\mathcal{F}(p)$ we can save the data on only one or a small number of sensors, say the sensor p, and save a pointer on every other sensor indicating where the real data is stored. If a query gets to a sensor on $\mathcal{F}(p)$, it can either follow the backward pointer stored at each sensor or simply use GLIDER intra-tile routing to reach the real data. The query may have to walk a little more once it finds the finger tree.

2) Data retrieval for information consumers: Suppose an information consumer y stays in a tile v. y routes to the tile with $f(c)$ as home landmark and check each tile on its way to see whether it has encountered a tile with data c. If the sequence of tiles from the y's resident tile to that of the hashed landmark $f(c)$ are $\mathcal{P}(y, f(c)) = T_1, T_2, \cdots, T_k$, where $T_i = T(v_i)$, with v_i as the home landmark of T_i, $v_1 = v$, $v_k = f(c)$. If the consumer finds the data in an intermediate tile, it returns to y. Otherwise the consumer can always retrieve the data from the last tile $T(f(c))$.

Now we describe how an information consumer finds the replicated data in a tile with home landmark u. Notice that the consumer does not know where the producer is, nor the finger tree $\mathcal{F}(p)$ which indicates where data is stored. The goal is to detect as soon as possible whether this tile has the replicated data. The motivation for our data retrieval algorithm comes from an observation as follows.

Theorem 4.1. *For a landmark Voronoi complex on a set L of landmarks, denote by $T(u)$ as the Voronoi tile of $u \in L$. Suppose r_i, $i = 1, 2, 3$, are neighboring landmarks of u. $\mathcal{F}(p)$ is a finger tree rooted at a point $p \in T(u)$. R is a curve in $T(u)$ that visits points on each boundary between u and r_i, $i = 1, 2, 3$. Then the polygonal curves $\mathcal{F}(p)$ and R must intersect*[2] *inside $T(u)$.*

Proof: See Figure 6 for an example. If r_i is a neighboring landmark of u, the path from p to r_i on the finger tree $\mathcal{F}(p)$ follows the shortest path in the subgraph spanned on $T(u) \bigcup T(r_i)$. Clearly this path intersects the boundary between the Voronoi cells $T(u)$ and $T(r_i)$. We denote by the intersection as s_i and denote the path between p and s_i on the finger tree by $P(p, s_i)$. The three paths $P(p, s_i)$ partition $T(u)$ into three pieces. Inside each piece we can not have all the three boundaries between $T(u)$ and $T(r_i)$, for $i = 1, 2, 3$, appear. Therefore, any curve inside $T(u)$ that passes through three nodes, one on each boundary, must intersect $\mathcal{F}(p)$. Thus, $\mathcal{F}(p)$ and R intersect. □

Motivated by the above theorem, we describe an information retrieval algorithm for the consumer to fetch data if the desired data is indeed replicated in $T(u)$. Suppose that the consumer

[2]By intersection we mean the polygonal curves intersect, i.e., they either intersect at a common sensor, or two links intersect.

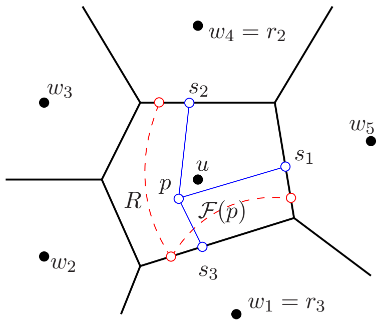


Fig. 6. For a Voronoi diagram on point set L in the plane, denote by $T(u)$ as the Voronoi cell of $u \in L$. r_i, $i = 1, 2, 3$ are three neighbors of u. Suppose $\mathcal{F}(p)$ is the finger tree rooted at a point $p \in T(u)$, R is a curve in $T(u)$ that has points on each boundary between u and r_i, $i = 1, 2, 3$. Then $\mathcal{F}(p)$ and R must intersect inside $T(u)$.

enters a Voronoi cell $T(u)$ at a node q (or y, at the consumer's resident tile). If the next Voronoi cell on the path $\mathcal{P}(y, f(c))$ is v, then v is taken as guide r_1. The path that the consumer follows is denoted as the *retrieval path*, which is defined as follows.

- **Retrieval rule A**: If u has at least three neighboring landmarks, then consumer routes by GLIDER inter-tile routing, i.e., shortest path towards r_2 in the neighborhood graph on $T(u) \bigcup T(r_2)$ until it hits a node t_2 on the boundary of $T(u)$. Then the consumer routes from t_2 towards the landmark r_3 in the same way until it hits a node t_3 on the boundary of $T(u)$. Finally the consumer routes from t_3 towards the landmark r_1 until it hits a node t_1 on the boundary of $T(u)$.
- **Retrieval rule B**: If u has a guide r_i which is not a landmark node, we use the GLIDER intra-tile routing scheme in replace of the inter-tile routing scheme as above.

Theorem 4.1 says that if an information consumer travels along a path that visits a point on each boundary between $T(u)$ and $T(r_i)$, for $i = 1, 2, 3$, then the path that the consumer follows must intersect the finger tree of the producer. Such an intersection might be a common sensor, in which case the information consumer finds the data, or a pointer to where the data is held. In a rare case, it is also possible that there is only an intersection between two links, one on the retrieval path, the other one on the finger tree. When two links intersect, it's very likely that one node has direct communication links to the other three nodes[3]. Since sensors use wireless communication so that a single broadcasting reaches every node inside the communication range, the sensor can respond to the broadcast query if it has the finger tree information. On the other hand, an algorithmic fix to guarantee that the information consumer finds out finger tree at a link crossing can be done in two ways. One scheme is to enhance the storage scheme by storing pointers on all the one-hop neighbors of the sensors on the finger tree such that the consumer will always encounter a sensor within one hop from the finger tree. The other scheme is to enhance the retrieval scheme by searching not only the sensors on the retrieval path but also their one-hop neighbors. By either of the two schemes, it is guaranteed that if the producer did replicate data in the tile $T(u)$, the consumer always finds this out.

[3]As shown in many places before, if the network model resembles the unit disk graph model and two edges intersect, then one node has edges to the other three nodes [9].

We also remark that the finger tree is rooted at a random node inside a tile $T(u)$. In the same tile, multiple producers create finger trees rooted at different nodes. By the randomness, two finger trees have few nodes in common. For different information consumers, since they usually enter the tile at different entry points, the information retrieval paths they follow are different as well. Thus our scheme is a good double ruling scheme that not only guarantees the information retrieval path to hit every information replication finger tree but also balances the load in doing so.

C. Improvement on query cost

In this section we describe a heuristic to improve the performance of our scheme, in particular the query cost. Specifically, the heuristic improves the query cost by using a shorter retrieval path than the zig-zag path. In particular, we construct the finger tree more carefully so that the consumer will hit the finger tree in most cases by just following the GLIDER routing path to the hashed tile.

1) Find guides for finger trees: We first explain a property of GLIDER that we will use later. For a landmark x, we can check whether two neighbors on the combinatorial Delaunay graph are also adjacent by checking whether they have an edge in $D(L)$. An easy observation is as follows.

Lemma 4.2. *Suppose x, y, z are three adjacent landmarks on a shortest path on the CDG, then x, z are not adjacent on CDG.*

Proof: If x and z are adjacent on CDG, then by deleting y we can get a shorter path. □

Notice that the information producer and consumer use the same shortest path tree rooted at the hashed tile $T(f(c))$. Assume the current tile is $T(u)$ with home landmark u. Denote by $\{w_j\}$, $j = 1, \cdots, k$, the set of landmarks adjacent to u on $D(L)$. t is the next hop on the shortest path towards the destination tile. By the above Lemma, the information producer and consumer enter the tile $T(u)$ from the tiles that are not adjacent with the tile $T(t)$. Intuitively we should choose the guides as the next-hop landmark t and u, t's common neighboring landmarks whenever possible. Thus the finger tree 'protects' the boundary between tiles $T(u)$ and $T(t)$ so that the consumer will hit the finger tree by just following the GLIDER routing path towards t. See Figure 8 for an example. In particular, when u has at least three neighboring landmarks on CDG, we choose the three guides r_i, $i = 1, 2, 3$, for the information producer as follows.

- **Replication rule C**: If the current tile $T(u)$ is not the hashed tile, we denote by t the next landmark on the shortest path towards $f(c)$. If we can find two landmarks w_1, w_2 that are adjacent to both u and t, we choose r_1, w_1, w_2 as the three guides.
- **Replication rule D**: If the current tile $T(u)$ is the hashed tile, we choose the first guide r_1 as one neighbor of u on the CDG which is determined by the content of the data.

If we can find two landmarks w_1, w_2 that are adjacent to both u and t, we choose r_1, w_1, w_2 as the three guides.
- For the other cases, the guides are chosen as in replication rule A or B.

It has been observed through simulation that the replication rule C and D are the typical situations encountered. Replication rule B is included mainly for the completeness of the scheme.

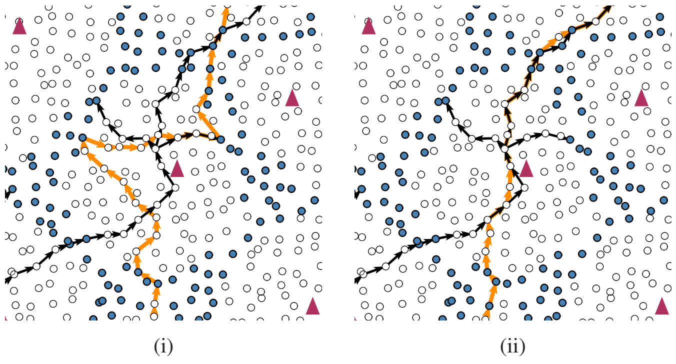


Fig. 7. Implementation of double-ruling within a tile. The black lines shows the path on which the producer replicates its data. The line in orange shows the consumer's path to retrieve the data. (i) The consumer follows a zigzag path; (ii) The consumer follows the GLIDER inter-tile routing to the next landmark.

2) Simplified retrieval scheme: With the help of the improved information storage scheme as described in the previous section, we can simplify the data retrieval scheme in this case. Namely, if data is replicated by replication rule C or D, the information consumer just follows the GLIDER routing path towards the first guide r_1, which is either the landmark on the next hop on the shortest path towards the destination tile $T(f(c))$, or a neighboring landmark determined by the content of data in replication rule D. See Figure 8.

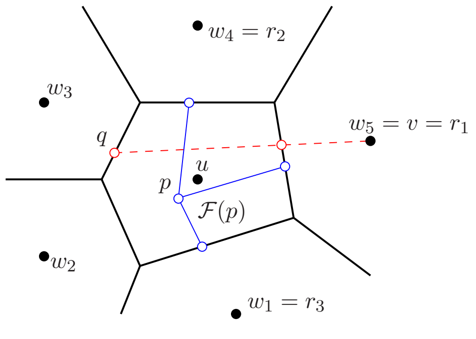


Fig. 8. The consumer's entry node is q. The path from q towards the landmark v via the shortest path on the neighborhood distance metric on $T(u) \bigcup T(v)$ intersects the finger tree $\mathcal{F}(p)$, for any $p \in T(u)$.

Suppose the information consumer enters the Voronoi cell $T(u)$ from the Voronoi cell $T(w_j)$, i.e., u is w_j's next hop on the shortest path from w_j to $f(c)$. By Lemma 4.2, w_j is not adjacent to the landmark t two hops on the shortest path to $f(c)$ on the CDG. Suppose the entry node of the consumer to $T(u)$ is q. The producer's finger tree with three guides as the next hop t and two common neighbors of t and u on the CDG blocks the boundary between $T(u)$ and $T(v)$ from the point q. Thus the path from q towards the landmark t via the shortest path on the neighborhood distance metric on $T(u) \bigcup T(v)$ must intersect the finger tree $\mathcal{F}(p)$, for any $p \in T(u)$. This simplification replaces the zig-zag curve by the natural GLIDER route towards the hashed destination tile. Please see Figure 7 for a comparison.

D. Structured data organization and query

For many applications of sensor networks such as tracking and event detection, data produced by sensors have natural structures and thus can be categorized in a more structured way. For example, a network for tracking produces data related to different moving targets such as cars, trucks, giraffes, elephants, people, etc. These data can be organized by categories at a coarser level, such as vehicle, animals and people. Within the hashed tile with all the data of the same category, the double ruling scheme guarantees that a single query along the data retrieval path will intersect *all* the finger trees on which different data are replicated. Also notice that since each finger tree starts at a random node inside the tile, those collections of finger trees are nicely spread out across the nodes, without overloading any particular one. Thus data generated in the sensor network are distributed not only evenly in the network, but also in a nicely organized way.

V. SIMULATIONS

We simulated the proposed information storage and retrieval scheme at the network layer for the purpose of validation. The simulator is written in C++ and runs the distributed algorithm presented. We focused on designing an algorithm that is both effective and lightweight enough for real system deployment. In this section, we study some practical work-around to further lower the query cost and evaluate the network level performance of our scheme vs. that of Geographic Hash Tables [16].

The simulated network has 2000 nodes distributed on a perturbed grid in a 316m by 316m field. The perturbation follows a zero mean Gaussian random distribution with a standard deviation of 4 meters. The radio range is 11 meters. Each node can communicate with a set of neighbors nearby. The average number of neighbors for each sensor is 6.2, which is sensible in a practical sensor networks setting. Among 2000 nodes, 23 are chosen as landmarks. Of the landmarks 18 are chosen randomly, with another 5 nodes added near the network boundary, after the random selection. In all the figures, sensors are shown as small circles and landmarks are shown as larger triangles. Blue circles represent nodes on the boundaries of Voronoi tiles. A typical scenario is shown in Figure 9 (i). Unless specified, all averages are taken on 50 simulations runs.

A. Data structure and storage requirement

Except for the CDG on the landmarks and the global hash function, each node stores only local information. Since the number of landmarks is small (23 landmarks out of 2000 nodes in our simulations), the total memory required for each node is manageable. Each node stores the following local information.

- the combinatorial Delaunay graph (CDG) on the landmarks;
- neighborhood distances to its reference landmarks;
- a global hash function;
- a bit to record if the node is on the boundary of a tile;
- the IDs of its neighboring sensors.

B. Enroute data aggregation

As discussed in section IV-A, for a specific category of data, for example the sighting of giraffes, the content-based hash function determines a tile, g, where all giraffe related information will be stored. The shortest path tree rooted at g (Figure 9 (i)), denoted by T, provides the *guidelines* at the CDG level for the information producers to reach g. On the way to reaching tile g, information producers replicate their data. In T, if two leaf nodes are near each other and far away from the root, it is very likely that they share a common ancestor before reaching the root. As the actual paths are local realizations of the CDG level paths, the replication paths of producers that disseminate giraffe information may merge before they reach the final tile g. When this happens data aggregation can be performed enroute. Figure 9 (i) shows a shortest path tree on landmark CDG. Figure 9 (ii) shows a possible enroute data aggregation scenario.

C. Reducing consumer query cost

The zigzag path (as shown in Figure 7 (i)) as introduced in section IV-B.2 guarantees that the consumer will find the data in a tile if the data is replicated in that tile. However, when data is not hashed in that tile, zigzag paths roughly double the communication cost compared with that of simply travelling towards the destination tile. The zigzag path is necessary only when replication rule C and D (Section IV-C) are not satisfied, which does not occur frequently in a reasonably designed network. If we opt to let the consumer follow the shortest path to the landmark of the next tile(figure 7(ii)), it will cross the replication path of the producers if the data is replicated in that tile by replication rule C and D. On the other hand, as both the producers and consumers follow the same shortest path tree at the CDG level, once they reach the same tile, they will head for the same next tile. Therefore, if a query misses the data replication tree in one tile, there is a big chance that it can meet the data in the next tile. Our simulation results have shown that, on the average, this practical work-around is about twice as energy-efficient when compared with using the zigzag path if tile adjacency conditions are nice. In the case that tiles are separated by holes, using the zigzag path helps the consumers to retrieve data early on.

D. Performance comparisons with GHT

As discussed in details in [8], routes generated using GLIDER are of comparable length to those by geographic routing, such as GPSR [12], in networks with densely, evenly distributed nodes. In case of nodes not densely and evenly distributed, i.e. with the existence of holes, geographic routing yields paths that 'hug' the boundaries of holes. As a consequence, not only the boundary nodes are excessively burdened with network traffic, but also longer paths (in terms of number of hops) are generated in comparison with the paths given by GLIDER. Because our information storage and retrieval scheme is built on top of GLIDER, this advantage helps the performance of our scheme vs. GHT in terms of load balancing and the path lengths of the producers as well as the consumers.

number of producers per tile	1	2	3	4
cost ratio to GHT	1.63	1.24	1.04	0.98

TABLE I. The cost ratio of the proposed scheme to that of the GHT as the number of producers per tile varies.

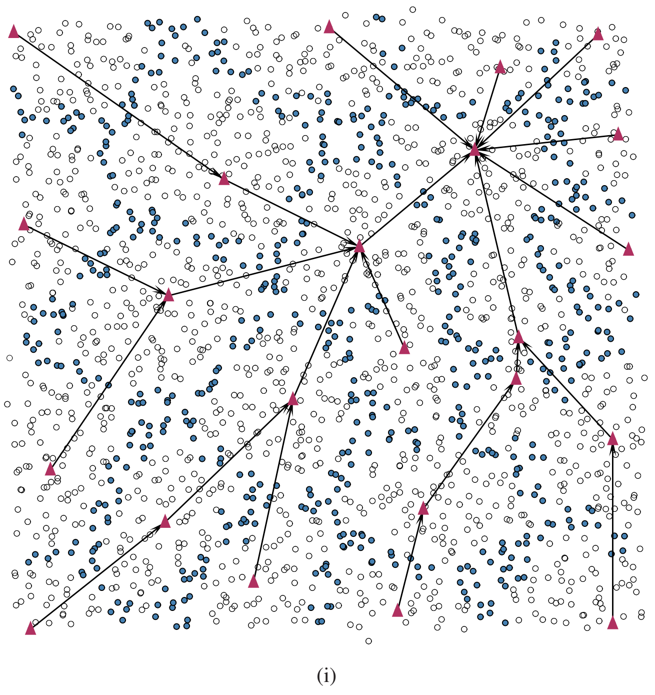
We discuss performance issues such as average path length, locality awareness and load balancing in comparison with GHT in the following subsections.

1) The cost for producers: Compared to GHT, our scheme has its advantages and drawbacks:

1) Advantages: The enroute producer data aggregation made possible by the high-level topological information and the CDG reduces the total cost for producers that send the same type of data to the hashed tile. The biggest saving comes from the case when multiple producers reside in the same tile. They will meet with one another within the tile. They can then share one data replication path to the hashed tile and at that time data aggregation can be performed.
2) Disadvantages: Although the producers travel about the same distance as compared to that in GHT to reach the hashed tile, the additional two finger paths along each title they travel incur extra communication costs.

To evaluate these tradeoffs, we randomly pick one producer in each tile. We then count the number of transmission needed to build the data replication paths and compare it to that required by GHT. Our simulation shows that our scheme gives a cost on average about 1.63 as expensive as that by GHT. However, if we double the number of producers in each tile, the cost ratio reduces to 1.24. In fact, the more the producers per tile, the better cost ratio of our scheme to that of GHT is achieved. This this due to the fact that in GHT, each information producer follows its own path to the hashed node using geographic forwarding. In our scheme, because of the use of hierarchy and the double ruling, we guarantee that producers residing in the same tile will have their data replication paths meet with one another before exiting their residence tile. From that point on, they can aggregate their data and share the same replication path. Of course, such saving is only possible when multiple producers send their data at about the same time. Table I shows more simulation results. These results are affected by the network scenario used in the simulation. More specifically, they are affected by factors such as the total number of tiles, the location of the hashed tile, etc. However, the trend that the cost ratio decreases with increasing number of producers within each tile remains apparent.

2) Locality awareness and the cost for consumers: Locality awareness is realized by the introduction of hierarchy and the shared shortest path tree at the tile level for both the producers and consumers of the same data. For consumers, our proposed



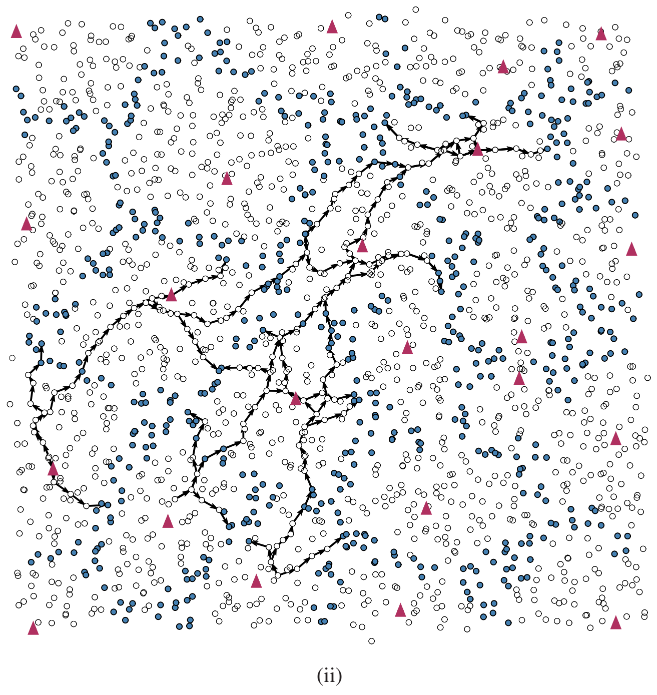


Fig. 9. Landmarks are shown by triangles. Sensor nodes are shown as small circles. Circles in blue are sensors on the boundaries of adjacent tiles. The dark nodes are the boundaries of the tiles. (i) A shortest path tree on landmark CDG; Each straight line denotes a branch in the tree with the arrow pointing to the parent of the node at the other end; (ii) A possible enroute data aggregation scenario.

scheme always yields shorter data retrieval paths. First, for consumers that reside in the same tile with the producer, our scheme guarantees that the consumers can retrieve the data within that tile. Second, as discussed in the previous sections, because the producers and consumers of the same type of data follow the same shortest path tree at the CDG level the consumers may get the data before they reach the hashed tile. The worst case for the consumers is that they have to travel to the hashed tile to fetch the data, which is what the consumers have to do in GHT.

To verify this, we fix a producer and let each node be a consumer of that data produced by the producer. We count the number of hops each consumer has to travel to get the data and compare it with that required by GHT. Table II shows how the cost ratio varies as the distance between the consumer and the producer increases.

Our simulation results show that our scheme to some extent alleviates the locality insensitivity of GHT. When the producer and consumer are near each other, our scheme takes advantage of the locality and gives a shorter data retrieval path. The numbers in this table will change with different network setups. For example, 8 hops seems to be a magic number here. The reason for this is that the tile size is roughly about 8 hops in radius. Changing the size of the tiles will alter the threshold numbers for the cost ratio to change significantly in that table. However, the trend that the cost ratio decreases with decreased distance between the producer and the consumer remains.

We also randomly picked 100 pairs of producer and consumer in the sensor field and simulated the data retrieval processes for the consumers. On average, the consumers travel about 70.2% of the path length compared to that using GHT before they retrieve the data.

disposition (hops)	≤ 4	8	12	16	20	24
cost ratio to GHT	0.17	0.16	0.44	0.47	0.91	0.95

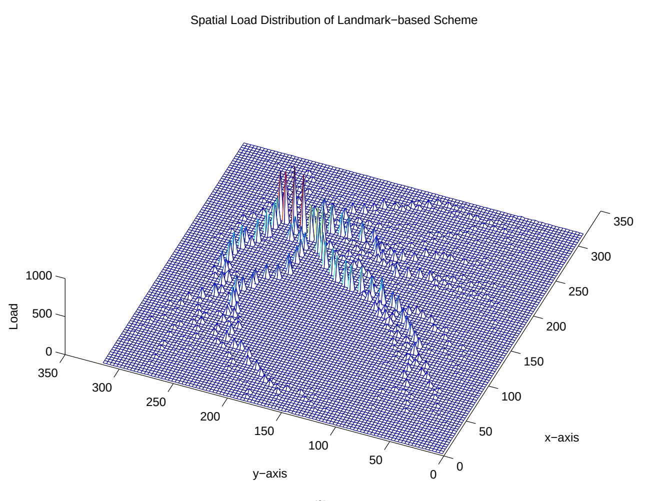
TABLE II. The cost ratio of the proposed scheme to that of the GHT as the hop count distance of the consumer from the producer increases.

3) Load balancing: The fact that information is hashed to a curve brings about other nice properties to our scheme in terms of load balancing and fault tolerance. To show this, we picked one information producer. We then let each node to be a consumer of the data from that producer. Each consumer retrieves the data using the proposed scheme. We record the network traffic load at each node and compare it with that by using GHT. Figure 10 shows the comparison.

The figures shows two things: first, the network traffic overtaxes the nodes on the boundary of a big hole in the network in GHT due to its use of geographic routing; second, there is a more severe hot spot phenomenon created around the hashed site of the data in GHT. For GHT, the max load is 1615. For the proposed scheme, the max load is 793. In addition, because data is hashed along a elongated curve in the proposed scheme, it has better fault tolerance.

VI. Conclusion

In this paper we proposed a data-centric, location-free, landmark-based information brokerage scheme for sensor networks. The efficiency of our scheme is fully manifested when there are multiple consumers and/or producers appearing throughout the network. Such advantages are attributed to the



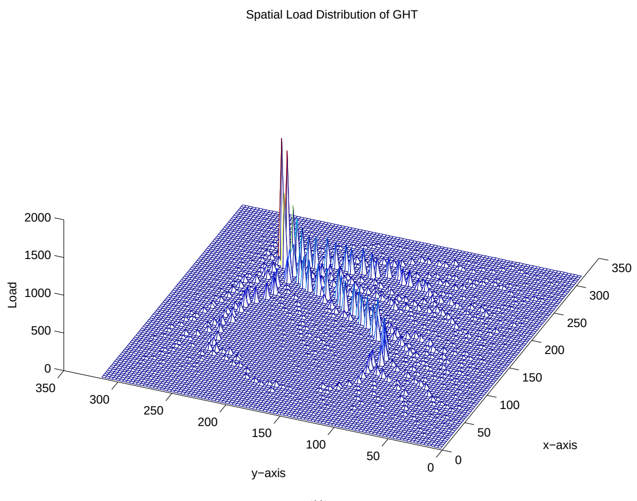


Fig. 10. Comparison of network traffic load at each node. (i) load distribution using the proposed scheme; (ii) load distribution using GHT. Notice that the maximum load value for graph (i) is 793, while the maximum load value for graph (ii) is 1615.

fact that we aggregate producer data before they reach the hashed tile when multiple producers of the same data are in close proximity of one another, as well as to the fact that a consumers can potentially retrieve data on their way to the hashed tile. This is possible because the shortest path tree shared by both the producers and the consumers (of the same content type), which is determined by the content-based hashing and the CDG on the landmarks. In addition, the double-ruling scheme implemented at the tile level not only provides means for the consumers to retrieve the desired data but also provides a fine-grained load-balanced distributed data storage for producers within each tile. We established guaranteed data retrieval for the consumers through an analytical study and demonstarted performance improvements over GHT through simulations.

Acknowledgements: The authors gratefully acknowledge the support of the DoD Multidisciplinary University Research Initiative (MURI) program administered by the Office of Naval Research under Grant N00014-00-1-0637, NSF grants CCR-0204486, FRG-0354543 and CNS-0435111, and DARPA grant 32905. The authors would like to thank Yusu Wang for useful discussions and the anonymous reviewers for their comments on improving the presentation of this paper.

References

[1] W. Adjie-Winoto, E. Schwartz, and H. Balakrishnan. The design and implementation of an intentional naming system. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 186–201, December 1999.

[2] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[3] D. Braginsky and D. Estrin. Rumor routing algorthim for sensor networks. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 22–31. ACM Press, 2002.

[4] G. Carlsson and V. de Silva. Topological approximation by small simplicial complexes, 2003. preprint.

[5] S. M. Das, H. Pucha, and Y. C. Hu. Performance comparison of scalable location services for geographic ad hoc routing. In *Proc. of the 24th Conference of the IEEE Communication Society (INFOCOM)*, March 2005.

[6] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[7] Q. Fang, J. Gao, and L. Guibas. Locating and bypassing routing holes in sensor networks. In *The 23rd Conference of the IEEE Communications Society (INFOCOM)*, volume 23, pages 2458–2468, March 2004.

[8] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang. GLIDER: Gradient landmark-based distributed routing for sensor networks. In *Proc. of the 24th Conference of the IEEE Communication Society (INFOCOM)*, March 2005.

[9] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric spanner for routing in mobile networks. In *Proceedings of the 2nd ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'01)*, pages 45–55, 2001.

[10] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the Symposium on Operating Systems Principles*, pages 146–159, October 2001.

[11] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *ACM Conf. on Mobile Computing and Networking (MobiCom)*, pages 56–67, 2000.

[12] B. Karp and H. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 243–254, 2000.

[13] J. Li, J. Jannotti, D. Decouto, D. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In *Proceedings of 6th ACM/IEEE International Conference on Mobile Computing and Networking*, pages 120–130, 2000.

[14] X. Liu, Q. Huang, and Y. Zhang. Combs, needles, haystacks: balancing push and pull for discovery in large-scale sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 122–133. ACM Press, 2004.

[15] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.

[16] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A geographic hash table for data-centric storage in sensornets. In *Proc. 1st ACM Workshop on Wireless Sensor Networks ands Applications*, pages 78–87, 2002.

[17] I. Stojmenovic. A routing strategy and quorum based location update scheme for ad hoc wireless networks. Technical Report TR-99-09, SITE, University of Ottawa, September, 1999.

[18] 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 '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 148–159, New York, NY, USA, 2002. ACM Press.