

# Double Rulings for Information Brokerage in Sensor Networks

Rik Sarkar\*      Xianjin Zhu\*      Jie Gao\*

\* Department of Computer Science, Stony Brook University. {rik, xianjin, jgao}@cs.sunysb.edu

**Abstract**—We study the problem of *information brokerage* in sensor networks, where information consumers (sinks, users) search for data acquired by information producers (sources). In-network storage such as geographical hash table (GHT) has been proposed to store data at rendezvous nodes for consumers to retrieve. In this paper, we propose a *double rulings* scheme which stores data replicas on a curve instead of one or multiple isolated sensors. The consumer travels along another curve which is guaranteed to intersect the producer curve. The double rulings is a natural extension of the flat hashing scheme such as GHTs. It has improved query locality, i.e., consumers close to producers find the data quickly, and structured aggregate queries, i.e., a consumer following a curve is able to retrieve *all* the data. Further, by the flexibility of retrieval mechanisms we have better routing robustness (as multiple retrieval paths are available) and data robustness against regional node failures. We show by simulation that the double rulings scheme provides reduced communication costs and more balanced traffic load on the sensors.

## I. INTRODUCTION

The sensor network community has envisioned a large variety of applications, from scientific data collection, to environment monitoring, to smart sensing and distributed control. Early applications of sensor networks for scientific data collection and habitat monitoring have mainly adopted a many-to-one traffic pattern to deliver raw sensor readings towards data loggers [1], [31]. Emerging applications that combine distributed sensing and control observe more interesting traffic patterns and impose more stringent delay requirements. In this paper we focus on the type of applications where sensor networks provide large-scale intense monitoring over the environment, and/or tracking of interesting targets, as well as delivering the relevant data to the interested parties. Multiple nodes, with their low-level readings, may collaboratively arrive at a high-level semantic event report, e.g., ‘elephant sightings’, based on which corresponding authorities need to be notified for response. Users of the sensor networks may well be embedded in the same physical space and inject queries to the network at any time searching for certain types of data. This category of applications covers a large number of scenarios, from target detection and response, to resource management and coordination, to health-care applications.

These large-scale sense-and-respond applications impose several requirements on data discovery and delivery protocols. Data queried by users are often highly selective. Sensor nodes may detect numerous events of different types, among which

each user is only interested in a much smaller subset. Real-time applications and distributed control systems often impose a high requirement on the access delay at users to ensure event reports or control commands being delivered on time, since information ages fast and stale data is useless. Furthermore, the arrival of queries may be spatially and temporally distributed. In order to serve users across time and space, in-network information storage is needed to support information aggregation and reasoning. History data helps to maintain temporal coherence and consistency, e.g., in target tracking applications. Data collected at spatially distributed locations may need to be integrated in order to come up with a global conclusion, e.g., the limited views from multiple spatially distributed cameras are combined to derive the semantics of an event.

In this paper we formulate the problem as the problem of *information brokerage* which, specifies how data is collected, processed and stored as well as how queries are routed to discover relevant data. We model the problem as the matching of *information producers* (also known as sources), that perform data acquisition and event detection, with *information consumers* (also known as sinks) who search for this information. Naturally in a sensor network there can be multiple producers that generate a variety of data types as well as multiple consumers, possibly mobile, that search for relevant information. We aim to develop a scheme for large-scale networks that support low-delay queries for multiple users that search selectively for data types discovered and stored in the network.

### A. Existing work

Early work on information discovery and routing follow two basic approaches: data-centric routing [14] and data-centric storage [29].

Data-centric routing takes a reactive approach, as in directed diffusion [14] and TinyDB [23]. Little collaborative preprocessing is performed. Thus the discovery of the desired information usually relies on flooding the network. This approach targets at infrequent queries for streaming data type so that the cost of flooding can be justified and amortized by the following long-term data delivery. For queries from multiple consumers for the same data source, the performance deteriorates as data sources might be re-discovered separately by multiple consumers. The delay incurred by information discovery may also be too high for real-time queries or delivery of control demands.

Data-centric storage is proposed for large-scale networks with many simultaneously detected events that are not necessarily desirable for all users [29], [26], as in the applications considered in this paper. A producer leaves data on rendezvous

A preliminary version appeared in the Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MobiCom’06), 286-297, 2006.

nodes for consumers to retrieve. Thus data across space and time can be aggregated at rendezvous nodes. The geographical hash tables (GHTs) [26] is a celebrated pioneer work in this category. In GHTs, data is hashed by its data type to geographical locations. The node closest to the hashed location is identified as the rendezvous (home) node where data is replicated at a set of ‘perimeter nodes’ around the home node. The consumer applies the same hash function and retrieves data from these nodes. Data and query delivery to the rendezvous node is implemented by geographical routing such as GPSR [17]. Later, the PathDCS [8] method generalizes GHTs to avoid using a routing infrastructure and bases routing entirely on trees rooted at landmarks.

GHTs have greatly reduced the communication cost and energy consumption by avoiding network-wide flooding for information discovery. Its simplicity is also attractive. While GHTs being a fundamental data storage scheme, one can improve on GHTs in the following directions. First, the data retrieval scheme in GHTs is not distance-sensitive. Even when the consumer is close to the producer, it may have to go to a far away rendezvous node. Second, the rendezvous node for popular data queried by many consumers imposes a communication bottleneck. This artifact in traffic patterns may eventually hurt the network lifetime. Third, as the rendezvous node and the replication nodes near the rendezvous node are vulnerable to regional node failures, structured replications on mirror nodes need to be adopted to improve the system robustness, at a higher cost of communication. Fourth, to achieve storage balancing, data is randomly scattered in the network. But with uniformly randomly distributed data it is difficult to support, in a communication efficient manner, efficient structured data organization or queries that require cross-type data aggregations. Improvement of the flat hashing by hierarchical hashing has been investigated with hash locations aware of data correlation, i.e., similar data is stored close by, or query locality, i.e., nearby consumers should discover producers more quickly [19], [20], [12].

### B. Double rulings

Our approach is to develop what is called *double rulings* scheme (also called *quorum-based* schemes), an extension of the basic GHTs hashing. The idea is to choose the rendezvous nodes along a continuous curve, instead of one or multiple isolated sensor nodes, as in the case of GHTs. The motivation is two-fold. Data delivery from data source to a rendezvous node is implemented by multi-hop routing. Thus it is natural to leave information hints along the trail that the data travels on, at no extra communication cost. Furthermore, data hint replication on multiple nodes provides more flexibility for a consumer to discover relevant data — it is easier to encounter a 1D curve than a 0D node.

A basic *double-ruling* scheme works as follows: data or pointers are stored at nodes that follow a *replication curve* while a data request travels along a *retrieval curve*. Any retrieval curve intersects the replication curve for the desired data. Thus successful retrieval can be guaranteed. For an easy familiar case, assume the network is a two-dimensional grid

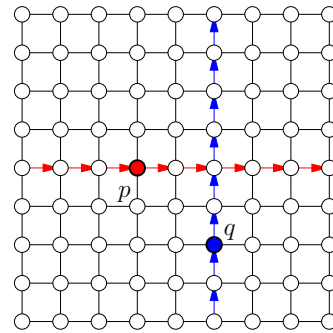


Fig. 1. A simple double ruling scheme on a grid.

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. To be differentiated with the double rulings we will propose in this paper, we call this simple double rulings scheme the *rectilinear double rulings*. Notice that the data retrieval curves are independent of the location of the data sources. In fact, a consumer traveling along the vertical line through itself is guaranteed to hit *all* horizontal storage lines, and thus is able to find *all* the data stored in the network. This double-ruling scheme is also distance sensitive — 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. By replicating data on more nodes that are not in close proximity with data sources or hashed locations, double rulings scheme enables better fault tolerance against geographically concentrated node failures.

Despite all these good properties, the rectilinear double rulings idea is so far restricted on networks with nice graph structures, e.g., those that resemble grids [22], [34], [30], due to its rich geometric flavor. Recently Fang *et al.* [11] has studied double rulings for general sensor field with non-trivial topology. The idea is to combine double rulings with GHTs on a two-level routing hierarchy that partitions the sensor field into tiles with respect to the global topology [10]. GHTs are adopted on the top level of the hierarchy and a data type is hashed to a tile instead of a single node. Inside each tile, a double rulings scheme is invented to hash data on routing paths such that retrieval paths will intersect the replication paths for sure. We note that rumor routing [3] can be considered as a probabilistic double rulings scheme. Information producer takes a walk (either a random walk or a straight trajectory) and leaves data pointers on the trail. A consumer travels along another walk hoping to encounter one of the data pointers. Any two walks have a probability to intersect. The consumer sends out enough retrieval walks to have a sufficiently high probability to meet with one of the event curves. Essentially the challenge of designing good double rulings is to find data replication and retrieval paths that intersect, are not too long each (not too many replications), and are evenly spread out across the network.

### C. Our contribution

Our work is motivated by GHTs and previous double rulings schemes. In this paper we investigate double rulings schemes with a focus on the flexibility of retrieval mechanisms. We propose a simple double rulings scheme that actually has GHTs as a subcase. Same as in GHTs, a data item is hashed by its data type (also called key in GHTs) to a geographical location. However, instead of traveling along the geographical greedy path to the rendezvous node, the producer travels along a circle that goes through itself and the rendezvous node and replicates data or data pointers on the way. We show that this simple modification to GHTs suddenly allows a large variety of retrieval mechanisms. The consumer does not necessarily travel to the hashed location to retrieve the data. It only needs to hit the replication curve. And we show that there are many such retrieval curves. Thus the consumer has great flexibility to design its retrieval strategy subject to the current network load and energy level. Among these retrieval schemes, several have special properties:

- **Distance-sensitive retrieval:** if the consumer is of distance  $d$  from the producer, the consumer can discover the data with a cost of  $O(d)$ , although neither has the knowledge of each other's location or the bound on  $d$ . This is an attractive feature in many applications, as information will be most useful, thus queried more frequently, in the spatiotemporal locale where it was collected.
- **Aggregated data retrieval:** in GHTs, if a consumer is interested in multiple data types, such as detections of both vehicles and animals, the consumer has to visit multiple rendezvous nodes for these data types to collect all the data. In our double rulings scheme, we show there is a simple rule based on which one can design a curve (actually many such curves) that will surely intersect with all replication curves of desired data types. Thus the consumer travels along a simple curve and gathers all the information.
- **Double rulings retrieval:** the most powerful retrieval mechanism is to travel along any double ruling curve (among many such curves) that will intersect all replication curves. Thus a user can discover all the information discovered and stored in the network. This has further applications in data collection by data mules.

Our double rulings scheme can be considered as a special extension of GHTs. By modestly increased replication, it supports distance-sensitive retrieval and structured data retrieval. In addition, the double rulings scheme has also improved load balancing and robustness to node failures. With the flexibility in retrieval curves, the rendezvous node is no longer a bottleneck since retrieval curves may not necessarily visit it. We show that the data storage admits a local recovery scheme. If the sensors in a certain region are destroyed, then all the relevant data are stored on the boundary and thus can be locally recovered. Compared with structured replication in GHTs or hierarchical hashing that aims to improve data robustness, the double rulings scheme imposes much lower communication cost for replication, since the replicas are organized along a

closed curve that are easy to visit.

We name this new double rulings scheme *spherical double rulings*, to be differentiated from the simple double rulings scheme with vertical/horizontal lines (which is denoted as *rectilinear double rulings*). As it may not be apparent, the spherical double rulings philosophically generalizes rectilinear double rulings and contains it as a subcase. The key insight about the difference between spherical and rectilinear double rulings and why the spherical double rulings provide more nice features will be discussed in subsection II-D, after the description of our design. We discuss the implementation issues in section III and compare its performance with GHTs [26] and a previous double rulings scheme [11] in the simulation section.

## II. SPHERICAL DOUBLE RULINGS

In this section we will use a continuous domain for the intuition and easy explanation. In a discrete network, a continuous double ruling curve can be easily implemented by a path in the network in a greedy fashion [25]. The implementation details are presented in the next section. As in GHTs, we assume that the sensor nodes know their geographical locations and a few parameters of the sensor field such as the diameter and the boundary. We consider this a reasonable assumption, as localization is a fundamental component for network functionalities and is important for the integrity of sensor readings.

### A. Projective mapping

For an easy explanation, we use projective geometry to map sensor nodes onto a sphere, in particular, the stereographic projection [5]. We put a sphere with radius  $r$  tangent to the plane at the origin. Denote this tangent point as the south pole and its antipodal point as the north pole. A point  $h^*$  on the plane is mapped to the intersection of the line through  $h^*$  and the north pole with the sphere. See Figure 2. One can view the north pole as the location of an observer and the plane as the canvas. This provides a one-to-one mapping of the plane to the sphere, in addition, with the north pole mapped to the point of infinity. More details on projective geometry can be found in [27]. Stereographic projection preserves circularity. Any circle on the sphere, including great circles, is mapped to a circle in the plane. It is also a conformal mapping, i.e., one for which local (infinitesimal) angles on a sphere are mapped to the same angles in the projection. It does not preserve distances or area, however. The distortion around the north pole can be high.

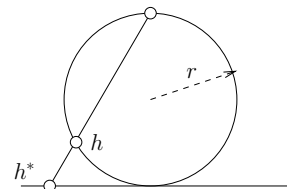


Fig. 2. Stereographic projection.

Let the sphere be defined by the equation  $(\mathbf{x} - \mathbf{p}) \cdot (\mathbf{x} - \mathbf{p}) = r^2$  where  $\mathbf{p}$  is the center of the sphere, and  $r$  its radius. The

straight line from a point  $\mathbf{q}$  to the north pole of the sphere (denoted by  $\mathbf{n}$ ), is given by  $\mathbf{l}(t) = \mathbf{q} + t\mathbf{v}$ , where  $t$  is the parameter and  $\mathbf{v} = \mathbf{n} - \mathbf{q}$ . Then the intersections of the straight line with the sphere are defined by the roots of the quadratic equation

$$t^2(\mathbf{v} \cdot \mathbf{v}) + t(2\mathbf{v} \cdot (\mathbf{q} - \mathbf{n})) + ((\mathbf{q} - \mathbf{n}) \cdot (\mathbf{q} - \mathbf{n}) - r^2) = 0.$$

One root corresponds to the north pole  $\mathbf{n}$ , and the other is the projection. Thus, given the sphere, and a point in the plane, we can compute the mapping of the point on the sphere.

Conversely, given a point  $\mathbf{h}$  on the sphere, its projection on the plane will lie on the straight line  $\mathbf{l}'(t) = \mathbf{h} + t\mathbf{w}$ , where  $\mathbf{w} = \mathbf{h} - \mathbf{n}$ . We define the plane by  $(\mathbf{x} - \mathbf{o}) \cdot \mathbf{z} = 0$  where  $\mathbf{o}$  is the origin, and  $\mathbf{z}$  is the unit vector perpendicular to the plane. Then the projection of  $\mathbf{h}$  on the plane is given by

$$\mathbf{h}^* = \rho(\mathbf{h}) = \mathbf{h} + \frac{(\mathbf{o} - \mathbf{h}) \cdot \mathbf{z}}{\mathbf{w} \cdot \mathbf{z}} \mathbf{w}.$$

The stereographic projection maps an infinite plane onto a sphere. For a sensor network field, the area in which the sensor nodes lie correspond to a finite region of the plane. Let this region be called  $\mathbf{S}$ . Thus, any point in  $\mathbf{S}$  maps to a point  $\mathbf{h} = (x, y, z)$  on the sphere where  $z \leq k$  for some  $0 < k < 2r$ . The radius  $r$  can be adjusted for a suitable value of  $k$  in this range. The distance from the origin to the point  $\mathbf{h}^* = \rho(\mathbf{h})$  is given by  $2r\sqrt{z/(2r-z)}$ . Also, the distance from the origin to the point  $(x, y, 0)$  is given by  $\sqrt{z(2r-z)}$ .

With the knowledge of the sensor field, we can place the sphere at the center of the sensor field. Suppose the furthest sensor node is of distance  $D$  from the origin (the south pole of the sphere). Then the parameter  $k$ , i.e., the  $z$ -value of the highest projection on the sphere, is at most  $2r \cdot \frac{D^2}{4r^2 + D^2}$ . At the end of this subsection we show that for a finite region, we can choose  $r$  such that the mapping gives a constant distortion on the distances. Specifically, we choose  $r$  as  $D/(2\sqrt{\varepsilon})$ ,  $\varepsilon > 0$ .  $k = 2r\varepsilon/(1 + \varepsilon)$ . Recall that circles on the sphere map to circles in the plane. The next theorem shows that the lengths on the circle on the sphere is not too much different from the lengths on the circle in the plane. The proof is in the appendix.

**Theorem 2.1.** *Consider any two points  $p_1$  and  $p_2$  on the sphere with their projections on the plane,  $\rho(p_1)$  and  $\rho(p_2)$ . If the distance from  $p_1$  to  $p_2$  along a circle is  $d$ , and the distance between  $\rho(p_1)$  and  $\rho(p_2)$  along the projection of the circle is  $\ell$ , then we have*

$$\frac{\ell}{d} \leq \frac{2r}{2r-k} = 1 + \varepsilon.$$

When  $\varepsilon = 1$ , all the points map to the bottom half sphere. We usually take  $\varepsilon$  as a constant larger than 1. For any two points in  $\mathbf{S}$ , their distance in  $\mathbf{S}$  along a circle is within a constant factor of the distance between their mappings on the sphere along the corresponding circle.

We use  $d(\cdot, \cdot)$  to represent the geodesic shortest distance between two points on the sphere and  $|\cdot|$  to represent the Euclidean distance in the plane. Thus we have,

**Corollary 2.2.**  $|p^*q^*| \leq (1 + \varepsilon)d(p, q)$ .

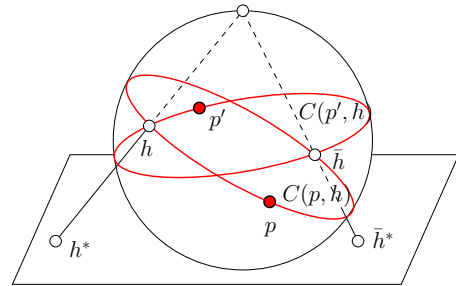
*Proof:* The shortest distance between  $p, q$  on the sphere must be along a circle. The distance between  $p^*$  and  $q^*$  along the projected circle in the plane is bounded by  $(1 + \varepsilon)d(p, q)$ . Further, the Euclidean distance of two points is always smaller than the distance along any circle. Thus  $|p^*q^*| \leq (1 + \varepsilon)d(p, q)$ . ■

With this mapping specified, we will explain our replication and retrieval schemes on a sphere. The above theorems imply that we can focus on the distances on the sphere. The real distances travelled in the sensor field are bounded by at most a constant multiplicative factor<sup>1</sup>.

## B. Data replication

For points on a sphere, there is an intuitive double ruling scheme — any two great circles of the sphere must intersect. Thus we can use great circles as the double rulings to replace the horizontal and vertical lines in rectilinear double rulings. There is one difference however. In rectilinear double rulings, the replication curves and the retrieval curves purely depend on the locations of producers and consumers. Through each node, there is a unique horizontal line and a unique vertical line. A point on a sphere, however, stays on infinitely many great circles. This property implies that the producer and consumer curves can have a lot of flexibilities, as we will see in the following.

We design a double rulings scheme that actually includes GHTs as a special case. Each data type is hashed to a geographical location  $h^*$  as in GHTs. When a producer routes towards the hashed location, instead of following the geographical greedy route as in GHTs, it follows the great circle defined by its own location  $p$  and the hashed location  $h$ , denoted by  $C(p, h)$ . Data from different producers with the same data type will be routed to the same hashed location where information aggregation can be performed. All the great circles with type  $C(*, h)$  pass through the hashed location  $h$ , as well as the antipodal point  $\bar{h}$ . Thus there are actually two rendezvous nodes,  $h$  and  $\bar{h}$ , located far away in the network that have all the information of the same data type. Notice that the hashed location  $h$  depends only on the data type. Thus the location  $\bar{h}$  can be derived by a simple geometric computation. See Figure 3 for an example.



**Fig. 3.** A point in the plane  $h^*$  is projected to a point  $h$  on the sphere. The great circles for two producers  $p, p'$  are drawn in red.

<sup>1</sup>This is subject to the assumption that the projective curve is within the sensor field and the sensors are dense enough such that the hop count of the path is proportional to its Euclidean length.

By the properties of stereographic mapping, a great circle is mapped to a circle in the plane. In particular, the image of any great circle of the sphere encloses the tangent point of the sphere and the projected plane. These circles, i.e., replication curves, may have different sizes and centers. Figure 4 shows the actual routes followed by multiple producers.

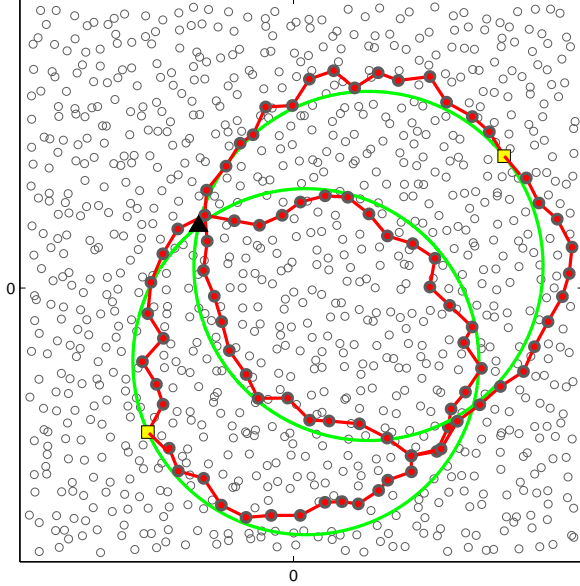


Fig. 4. Replication curves of multiple producers with the same data type. The hashed location is denoted by the dark triangle. Both the virtual replication circles and the actual routing paths are shown.

The hash function picks two geographical locations  $h^*$  and  $\bar{h}^*$ . The rendezvous nodes are selected as those closest to these locations and can be discovered by greedy forwarding in a similar way as in GHTs [26] (more details in next section). We abuse the notation a little bit and use  $h^*$  and  $\bar{h}^*$  to represent the hashed rendezvous nodes as well. The data is always replicated at the hashed rendezvous nodes  $h^*$  and  $\bar{h}^*$ . Data of the same type from multiple producers is aggregated at the rendezvous nodes  $h^*$  and  $\bar{h}^*$ . Dependent on the storage requirement, other nodes on the replication curve either store the real data or simply a pointer to where the real data is stored.

### C. Data retrieval

With this new routing strategy from producers to hashed locations, the retrieval scheme for the consumer  $q$  can be more flexible than that in GHTs. Observe that the mapping described in section II-A leaves an empty region near the north pole of the sphere that projects to points outside the network, and it is possible that a curve chosen by the consumer on the sphere intersects the replication curve in this region. However, it is possible to set up the projection in a way such that the consumer always has a guaranteed strategy to retrieve the data.

**Lemma 2.3.** *Any two great circles  $g_1$  and  $g_2$  on the sphere  $S^2$ , will have an intersection in the closed lower hemisphere  $\mathbb{H}_l$ .*

*Proof:* The sphere  $S^2$  is a double cover<sup>2</sup> of the real projective plane  $\mathbb{RP}^2$  [24], with the great circles corresponding to the geodesics of the projective plane. In particular, consider 2 copies of  $\mathbb{RP}^2$  glued along the equator to form  $S^2$ . By definition, any two geodesics in the projective plane must intersect. Therefore, great circles  $g_1$  and  $g_2$  on  $S^2$  must have an intersection in each of the hemispheres, in particular, they must have an intersection in  $\mathbb{H}_l$ . ■

**Theorem 2.4.** *If the sensor field fully contains a disk of radius  $2R$ , then the projective sphere of radius  $R$  placed at the center of this disk guarantees that any pair of great circles will have an intersection that projects to a point inside the sensor field.*

*Proof:* A point at the equator of the sphere corresponds to a point in the plane that is at a distance of  $2R$  from the south pole, by similarity of triangles. Therefore, projection of the disk of radius  $2R$  centered at the south pole completely covers the lower hemisphere of the projective sphere. By the lemma above, any two great circles will have an intersection in the lower hemisphere, which will thus project to a point inside the sensor field. ■

Depending on the requirements of the application and the specifications of the user, it may be desirable to use different data retrieval strategies. We present a number of such retrieval rules as well as their properties.

1) *GHT retrieval:* Obviously the same retrieval rule as in GHTs can still be used, with two rendezvous nodes though.

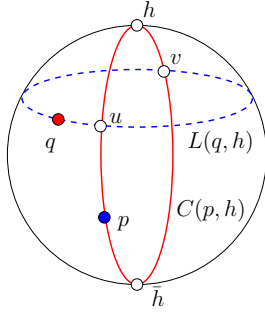
**Definition 2.5. GHTs retrieval rule:** *the same as in GHTs, the consumer can route to the hashed location  $h^*$  or  $\bar{h}^*$ , whichever is closer, to retrieve all the data of the same type.*

This retrieval scheme, as in GHTs, suffers from two disadvantages. It is not distance sensitive. Even when the consumer is actually close to producer, the hashed location might be far. On the other hand, popular data items will create communication bottleneck around the rendezvous nodes that hold them. With the simple modification of the replication curve, we show in this section several retrieval schemes that are distance sensitive and also alleviate traffic hot spot for popular data. Besides, it is attractive to have the flexibility of different data retrieval schemes, simply for load balancing and routing robustness.

2) *Distance-sensitive retrieval:* Assume that the distance between a producer and a consumer on the sphere is  $d$ , we would like to have a retrieval scheme where the distance traveled by the consumer is  $O(d)$ . Such a retrieval scheme is named *distance-sensitive*. Notice that the consumer does not know where the producer is and vice versa. The goal of the retrieval scheme is to travel along a curve that hits the replication curve as quickly as possible.

If we rotate the sphere so that the hashed location  $h$  is at the north pole, then the replication curve is exactly a longitude curve. The distance-sensitive retrieval scheme follows the

<sup>2</sup>That is, the sphere can be thought of as a union of two copies of the projective plane. The two copies are glued together at the boundaries to form the sphere, and diametrically opposite points on the sphere are the same point on the projective plane.



**Fig. 5.** The consumer follows the circle with fixed distance (dashed circle) to the hashed location  $h$  to retrieve all the data with the same data type.

latitude curve searching for a replication curve. We denote by  $L(q, h)$  this latitude curve. It is not necessarily a great circle. There are two intersections,  $u, v$ , between the retrieval curve and the replication curve, as shown in Figure 5. Now we claim that the closer one, in this case,  $u$ , is of distance at most  $d \cdot \pi/2$  from the consumer along the latitude curve  $L(q, h)$ . Obviously, the minimum distance from a point  $q$  to a set of points  $C(p, h)$  is always smaller than the distance from  $q$  to one point in this set, for example  $p$ . The following lemma says that the distance between  $q$  and  $u$  along the latitude curve  $L(q, h)$  is at most a factor of  $\pi/2$  of this shortest distance. The proof of the lemma appears in the appendix.

**Lemma 2.6.** *Take a longitude curve  $C$  through the north pole  $h$  and a latitude curve  $L(q, h)$  through a point  $q$ . Assume that  $u$  is the closer intersection of  $C$  and  $L(q, h)$  to  $q$ . Denote by  $k'$  the distance between  $q$  and  $u$  along  $L(q, h)$  and  $k$  the shortest distance from  $q$  to  $C$  on the sphere. Then  $k'/k \leq \pi/2$ .*

The consumer, however, does not know which direction to go to on  $L(q, h)$  to find the closer intersection  $u$ . This can be easily solved by a doubling trick, where the consumer chooses a direction randomly and travels a distance  $2^i$ , with  $i$  initially set as 0. If the consumer has not encountered an intersection with  $C(p, h)$ , it turns around, increases  $i$  by 1 and travels a distance  $2^i$  along the opposite direction from  $q$ . The process stops when the consumer discovers an intersection. Suppose at this point we have a parameter  $i$ , then  $d\pi/2 \geq k' > 2^{i-2}$ , where  $k'$  is the distance from  $q$  to  $u$  along  $L(q, h)$  and  $d = d(p, q)$ , the shortest distance between  $p, q$  on the sphere. The total distance traveled by the consumer is bounded by

$$2 \cdot \sum_{j=0}^{i-1} 2^j + k' \leq 9k' - 2 \leq 9\pi d/2 - 2.$$

In summary, we have

**Definition 2.7. Distance-sensitive retrieval rule:** *the consumer travels along the circle on the sphere with equal distance to the hashed location  $h$ , and uses a doubling trick to discover the closer intersection with the replication curve. The distance traveled by the consumer is at most  $O(d)$ , if the distance between producer and consumer is  $d$  on the sphere.*

The bound on the consumer cost is for the worst case scenario. We show by simulation later that the performance

is pretty good if we just choose a random direction. We note that here we focus on the continuous replication and retrieval curves. In a discrete network, the curves are realized by routing paths. When two continuous curves intersect, the corresponding routing paths may either have a common node, or have a pair of crossing links. We remark that under a unit disk graph model, if there are two crossing links, then one node must have links to all the other three nodes. With wireless broadcasting, all the nodes in the neighborhood can hear the message and are able to respond if they have the data. In practice, a consumer can also explicitly check the neighbors along the retrieval path or a producer explicitly store pointers on the neighbors along the replication path.

3) *Aggregated data retrieval:* The data replication scheme enables a number of interesting retrieval schemes for aggregated data. If the consumer travels along the latitude curve  $L(q, h)$  with  $h$  as the north pole, it actually can discover all the data with the same data type. In fact, *any* closed curve that separate the hashed location  $h$  from its antipodal point  $\bar{h}$  will intersect all the replication curves with the same data type. Thus a consumer is given great flexibility in choosing the retrieval curve according to the current network traffic load and energy consumption level. We formalize the data retrieval rule for aggregated data of several data types  $\{T_i\}$ ,  $i = 1, \dots, m$ .

**Definition 2.8. Aggregated data retrieval rule:** *the consumer searching for all the data with data type  $\{T_i\}$ ,  $i = 1, \dots, m$ , can follow a data retrieval curve that, for each data type  $T_i$ ,*

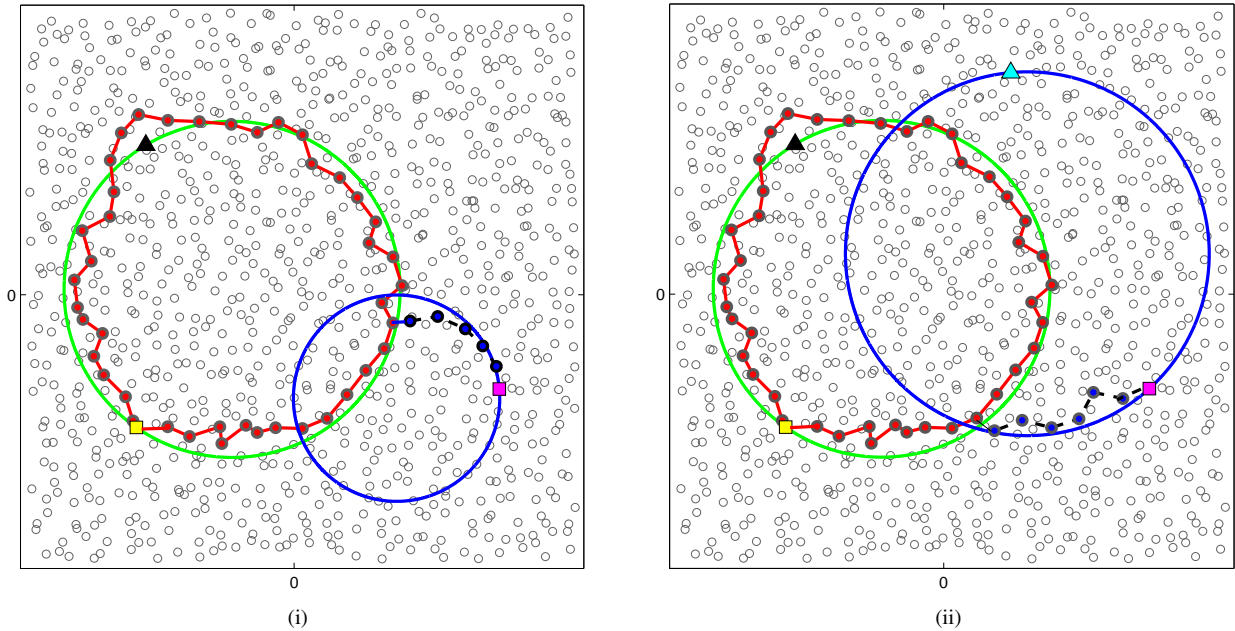
- *either goes through the hashed location  $h = h(T_i)$  or  $\bar{h}$ , where the aggregates are computed and stored;*
- *or is a closed curve that separates  $h$  from  $\bar{h}$ , collects all the relevant data and computes the aggregates.*

We remark that the above retrieval rule does not specify a unique retrieval curve but allow infinitely many possibilities. In fact, this is one of the advantages provided by this double rulings scheme. The design of a retrieval curve satisfying this rule can be performed at each consumer node. All the information needed is the data type and their hashed locations. Thus multiple consumers searching for the same data type may choose, by their own decisions, different routes. This flexibility of data retrieval rule enables load balanced traffic patterns and routing robustness.

4) *Double rulings retrieval:* The double rulings property enables a full power retrieval scheme. A consumer  $q$  following any great circle will definitely cross all the producer curves.

**Definition 2.9. Full power data retrieval rule:** *the consumer travels along any great circle and is able to retrieve all the data stored in the network.*

5) *Locality-aware data recovery:* The idea of replicating on a 1-dimensional curve, rather than a 0-dimensional point, greatly enhances the system robustness to failures. In GHTs, geographical routing with the combination of greedy forwarding and perimeter routing is used to deliver data to the hashed node. A planar subgraph, such as the relative neighborhood graph or the Gabriel graph, is extracted from the connectivity graph. When greedy forwarding can not find a neighbor closer to the destination, perimeter routing is adopted



**Fig. 6.** (i) Consumer latitude curve. (ii) Consumer great circle curve. Dark triangle denotes the hashed location; the red paths denote producer replication curves; dashed blue paths denote retrieval curves; yellow square denotes one producer and magenta square denotes one consumer.

to traverse the face in the planar graph. Specifically, the hashed geographical location, most likely, does not have a sensor node right there. Thus perimeter routing will be adopted to tour around the face that encloses the hashed location. The basic GHTs scheme relieves data loss at the home node, the one closest to the hashed location, by replicating the data around these perimeter nodes. However all the perimeter nodes are still in geographical proximity thus a ‘block error’ that destroys the sensors in a nearby region may destroy all the replicas. Structured replication can be used to improve the system robustness and relieve the traffic bottleneck at the home rendezvous node, in cases when too many events with the same key are detected in the network. Producers only put data at a nearby mirror node, while consumers may need to access multiple mirror nodes until they get what they want. The mirror nodes are chosen in a hierarchical way by using quad-tree structure. For the 1-level replication, the sensor field is partitioned into 4 equal size quadrants. The hashed location falls in one of them. 3 mirror nodes are chosen as those with the same relative locations inside other quadrants. More replication can be made in a recursive way. Such structured replication is costly since the mirror nodes are chosen to be geographically sparse. Replication along a curve improves the robustness without paying extra communication cost. Further we show that our replication rule supports local recovery when a group of nodes die.

In our spherical double rulings scheme, the hashed node is no longer the single point of failure. If the nodes in the neighborhood of the hashed node  $h^*$  are destroyed, the nodes on the boundary of the destroyed region contain all the relevant information and can be used to recover the aggregates. This is possible as long as the destroyed region does not include both the hashed location  $h^*$  and its antipodal point  $\bar{h}^*$ . Since  $h^*$  and  $\bar{h}^*$  are geographically fairly apart in the network, a

local disaster is not likely to cover such a large region. All replication curves for this data type will leave data replicas on a curve connecting  $h^*$  and  $\bar{h}^*$ , thus intersect the boundary of the destroyed region. So all the data replicated inside the destroyed region have their corresponding replica on some boundary nodes. These boundary nodes can be detected by a local greedy sweeping as in [9], or by using a topological method as in [33].

#### D. Spherical and rectilinear double rulings

After the detailed description on spherical double rulings and the various nice features it provides for data retrieval and recovery, we will give some insights on the key difference between spherical and rectilinear double rulings and why replacing vertical/horizontal lines with circles suddenly makes our lives so much easier. The following discussion is on a philosophical and mathematically abstract level and the goal is to help the readers better understand the essence of our design.

Let us begin with some superficial differences between spherical and rectilinear double rulings. Through each point in a plane, there is only a unique horizontal and vertical line. Data replica are left on the horizontal line through the producer, which depends completely on the producer location. Different producers (not on the same horizontal line) with the same data type store data on parallel horizontal lines. Thus data items with the same type do not encounter each other and can not be aggregated in-network. The nice feature of spherical double rulings is to introduce a hash location that brings the producer curves with the same data type together at the hashed node. This allows data aggregation at the hashed location, consistent aggregated data query, etc.

Note that this has particular advantage in aggregation of dynamic data. If new events of a particular type take place

during the network operation, then the rectilinear double ruling will simply store this information along a horizontal line, without aggregating it with existing data of that type. In the spherical double ruling case, this information will get aggregated with the other data of that type at the common hash node, at no additional expense.

From a projective geometry’s point of view, however, two horizontal lines do intersect — at the point of infinity. We can consider the rectilinear double rulings a special case of spherical double ruling — with all data types hashed to the point of infinity! Obviously there is no reason that all data types be hashed to the same node (point of infinity, i.e., the north pole of the sphere). In addition hashing to the point of infinity is not feasible in a finite sensor field. The spherical double ruling scheme essentially makes two natural modifications to rectilinear double rulings to do it right: we distribute hash locations grouped by data types and bring all hashed locations back to be within a finite sensor field.

### III. IMPLEMENTATION

We present a number of implementation issues of the double rulings scheme for a sensor networks, from the aspect of information stored at a sensor and the implementation of the replication and retrieval paths.

#### A. Information stored at a node

The information kept at a node is very small. Each node stores its own location and a few parameters about the stereographic projection used in the scheme. Every node keeps the global content-based hash function that takes a data type, e.g., types of target detections, and outputs a hashed location on the sphere. The hash function can be either a random function or a function that is locality-sensitive, i.e., similar data is hashed to similar locations [6]. The choice of this function is dependent on the applications. As mentioned earlier, data is either replicated on all the nodes along the replication curves; or, for the sake of saving storage, data is only replicated selectively on some nodes (say, uniformly sampled) along the curve and pointers are left on the rest of them pointing to the closest replica on the curve. A retrieval path hits the replication curve, gets the pointer and travels to the closest replica to find the data. More replication reduces data retrieval cost. We evaluate this tradeoff in the simulation section.

#### B. Greedy routing on a curve

The replication curve and retrieval curve are implemented as routing paths in the sensor field. We use the mechanism described in [25] to allow producers and consumers to respectively replicate and retrieve data along a curve. A node is passed a parametric equation of the curve in terms of a parameter  $t$ , and the direction along the curve for forwarding the message. Using the coordinates of its neighbors, a node finds a neighbor that is further along the curve than itself in the required direction, and sends the message to this neighbor. This can be done by taking a uniform sampling along the curve within a certain distance of the node, and computing the nearest such sample for every neighbor. Further, we use

the greedy version of this scheme, that is, the message is forwarded to the neighbor who advances furthest along the curve in the required direction. This ensures that replication or retrieval is completed in the fewest number of steps. We also require that the next hop is not too far from the virtual curve, say within 1 unit. For example, in Fig 7 the node  $S$ , when forwarding messages along curve  $C$  in the direction indicated, can only forward messages to its neighbors in the shaded region.

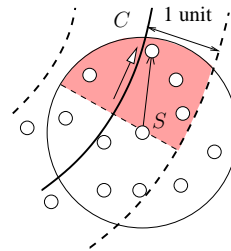


Fig. 7. Routing on a curve.

In networks with relatively low density of nodes, it is possible for a message to arrive at a node that has no neighbor in this region. When that happens, we perform a local flooding and search 2-hop neighbors. In a network with uniformly randomly deployed sensor nodes, simulation results show that the chance of finding a greedy neighbor is high even in networks with average degree as low as 5 or 6. It is easy

Avg degree	5	6	7	8
Avg 2-hop Neighbor	12	15	18	21
Min Success Prob.	0.94	0.97	0.985	0.992

TABLE I. Average number of 2-hop neighbors and the minimum probability of finding a 2-hop neighbor for greedy forwarding for different node degrees.

to see that the feasible region for forwarding always contains a sector of angle  $\theta_1 = \frac{\pi}{2} - \arcsin\left(\frac{1}{2(c+1)}\right)$ , where  $c$  is the radius of the circle along which routing is done. If we consider a 2-hop neighborhood, then the corresponding sector is given by an angle  $\theta_2 = \frac{\pi}{2} - \arcsin\left(\frac{2}{2(c+1)}\right)$ . Thus, the probability of finding a suitable neighbor within a 2-hop neighborhood is at least  $1 - \left(\frac{2\pi - \theta_2}{2\pi}\right)^{n_2}$  where  $n_2$  is the number of 2-hop neighbors. Table I shows the average number of 2-hop neighbors for different node degrees in a poisson distribution, and the corresponding worst case probabilities of a node finding a suitable 2-hop neighbor. For small values of  $c$ , the curve is small, and a search in a small neighborhood will always yield the results. The values in table I were computed with  $c = 3$ . For larger values of  $c$ , the probability can only be larger.

#### C. Dealing with sensor field boundaries

Recall that in a finite sensor field, we choose the location and size of the sphere so that the distance distortion of the projective mapping is bounded by a constant. Since the sphere maps to an infinite plane we actually chop off the region around the north pole, as no sensors are mapped there. The parameters used in the projective mapping (section II-A) do

make sure that the bottom half sphere are within the sensor field.

A great circle that goes near the north pole may not be completely within the sensor field. The intersection of two great circles, if close to the north pole, may also be outside the sensor field. In our implementation, a producer sends out replication data packets along both directions of the replication circle, which, when hit the network boundary, simply stop there. In other words, we only keep the replication curve that is within the finite sensor field. Still the actual replication curve, either a complete circle or not, partitions the sensor field in two parts, inside and outside the circle, such that any curve that visits the two parts will have to cross the replication curve.

Now we show that an incomplete replication circle is not much of a problem for data retrieval. To see this, assume the sensor field is circular and the sphere is placed at the center (how to deal with irregular sensor field and holes is challenging and will be discussed separately). Equivalently, the sensor field maps to the sphere except the cap above a certain latitude curve. First, any two great circles have two intersections. If one of the intersections is outside the sensor field (thus on the top half sphere), the other one must be on the bottom half sphere and be within the sensor field. This shows that full power retrieval can always find an intersection inside the sensor field and thus retrieve the data. Similarly, among the hashed location and its antipodal point, one of them must be inside the sensor field. Thus any closed curve that separates the hashed location and its antipode will certainly intersect the (possibly incomplete) replication curve. This includes the distance sensitive retrieval curves and aggregated data retrieval curves.

#### D. Networks of low density

In a network with low density it is possible that routing on a curve will fail at small local holes, where it is not possible to find a neighbor that makes progress or all such neighbors are far from the curve. This can be handled using a variation of face routing. In this variation, we simply follow the face boundary until we detect a second crossing with the curve, and then take up the curve again. We always follow the boundary in a fixed orientation (clockwise/counterclockwise). This will guarantee that for any two curves that intersect in the interior of the face, the corresponding paths will intersect.

## IV. SIMULATIONS

We show that simulation results confirm our assertions on the properties of double rulings. In this section, we compare the performance of *double rulings* and GHTs for both retrieval quality and load balancing.

#### A. Simulation setup

We simulated double rulings, GLIDER and GHTs on a network with 4225 nodes in a square field of size  $35 \times 35$  units. The communication radius of each node was taken to be 1 unit, hence the connectivity was that of a unit disk graph. The average number of neighbors was 9.5. Nodes were arranged in a grid model with perturbation. Each node deviates from its grid position by a random distance less than 0.5 units along

each axis. Producer and consumer costs were measured in the number of hops each had to take along their respective curves. Greedy routing along a curve was used for replication as well as for retrieval. The parameter  $\theta$  in the parametric form of a circle was used as the parameter for routing. To prevent the greedy forwarding from straying a large distance from the actual curve, forwarding was restricted to within a unit distance of the curve on the plane. For both distance-sensitive retrieval or full power double rulings retrieval, the consumer selects the retrieval curve, chooses a random direction and tours along the circle until it hits the desired replication curve(s). Once data is discovered, it is delivered back to the consumer in one of three ways: (i) finish up the remaining circle and get back to the consumer; (ii) turn around and follow the reversed path back to the consumer; (iii) use any routing scheme available in the system to deliver the data back to the consumer. Producers need only to replicate data, hence the cost for a producer is that of following a great circle producer curve.

#### B. Distance sensitive queries

It is desirable that consumer nodes are able to access data from nearby producers at a low cost. In GHTs, the consumer has to communicate with the same rendezvous node irrespective of the relative location of the producer. We simulated both schemes with pairs of producers and consumers at varying distances.

We also simulated the scheme of [11]. This is a scheme integrated with the GLIDER [10] landmark based routing protocol. Essentially a set of landmarks is selected in a preprocessing stage and the network is partitioned into Voronoi tiles, each formed by a landmark and all the nodes nearer to it than other landmarks (in the connectivity graph sense). This Voronoi tiling and its dual combinatorial Delaunay complex, in which two landmarks are joined by an edge if and only if their respective Voronoi tiles share common boundary, are used to aid routing. In [11] a distributed hash table is used to hash data into a tile, data storage inside the tile is implemented by a double-ruling scheme, which ensures information retrieval within each tile. As the producer travels to the hashed tile using GLIDER, and in every tile traversed, it replicates data on a “finger tree” with 3 arms. This guarantees that any retrieval trajectory passing through the tile will intersect the one of the arms. Thus, the consumer can find the information before reaching the hashed tile, and for a nearby producer, it is likely that the consumer will not need to travel all the way to the hash location.

It was mentioned in section II-B that in a double ruling scheme, actual data may be stored at a few nodes, while other nodes on producer curve store pointers to these. This saves cost of storing data, but increases communication cost for retrieval. We simulated the effects of this form of data replication.

For a particular distance value, we randomly select 100 pairs of producer/consumer. Figure 8(a) shows the simulation results averaging on 100 values for each plot. The different plots for double ruling are for different intervals of replication of the data. In all schemes we assume that data is delivered back to the consumer along the reversed data discovery path.

For GHTs, in which the average cost of retrieval is not related to the distance between producer and consumer, the consumer incurs the same cost in accessing local data as that for accessing remote data. The GLIDER based scheme is more distance sensitive, but the cost increases rapidly with increasing distance, and in the worst case is as bad as GHTs. With double ruling, the consumer searched for data along a latitude circle with respect to the hash location. There are two observations, the consumer cost in double rulings scheme is consistently smaller than that in GHTs, and is smaller than the GLIDER based scheme in almost all cases, except when the replication interval is very large. The average cost and the distance between producer and consumer exhibit a linear correlation for distances up to about half the length of the field, beyond that, the average cost does not increase. The glider based scheme performs well at short distances, because there the producer and consumer are likely to lie in the same tile, thus the consumer finds an intersection in the first tile itself. As the distance increases, the expected distance between the tiles on the producer's curve and those on the consumer's curve increases.

In networks of low density, we need to use face routing to guarantee delivery, as described in section III-D. We simulated this effect on networks of various densities by varying the number of nodes in fixed area. The plot in fig. 8(b) shows the average cost for 100 queries in networks with average number of neighbors per node varying between 4 and 12. The result shows that the expected cost does not really escalate too much. Note that, since at an average degree of 4 or 6, the network contains many holes, these results are representative of networks with non-trivial topologies.

### C. Double rulings retrieval for aggregated data

With queries involving different data types, double rulings scheme searches for data in a great circle, and can retrieve data at a fixed cost of 77.5 irrespective of the number of producers or data type. Whereas in GHTs, the cost is proportional to the number of different data types required, if we make a round trip to the hash location for each type<sup>3</sup>. Table II shows

No. Data Type	2	3	4	5
Consumer Cost	107.26	145.24	201.38	248.46

TABLE II. Average consumer costs for GHTs with aggregate queries for different types of data.

the consumer cost with aggregate queries for varying different types of data. The value is averaged on 100 randomly selected consumers for each column. It shows that for aggregated queries that search for more than 1 data types, it is almost always beneficial to adopt double rulings scheme instead of GHTs.

<sup>3</sup>One can also seek for the minimum tour visiting all hashed locations. This is the traveling salesman problem and is NP-hard to solve.

### D. Replication cost and tradeoff between storage and communication costs

In GHTs, structured replication can be used for data replication at several hash locations. The network is partitioned recursively by a quad-tree. For a given hashed location and a given hierarchy depth  $d$ , we can compute  $4^d - 1$  mirror images, for each square in the quad-tree with depth  $d$ . In our simulation, we set  $d$  to 1 and divide the entire network region into 4 quadrants. Thus, for each point, there are 3 mirrors in the other 3 quadrants. Producers replicate data at both the original hashed location and mirrors, so that consumers can retrieve data by accessing the closest location. Double rulings, on the other hand, selects a curve, and replicates data at nodes along this curve. We compared the performance of GHTs (with and without replication) against that of double rulings where the consumer uses a latitude curve as well as where the consumer uses a great circle to retrieve data. All of the simulations run over 500 producer/consumer pairs.

Table III shows the producer and consumer costs for double rulings and GHTs. In GHT, producer cost is higher than that of consumer, because the producer always replicates data in perimeter mode while consumers stop as soon as they reach any perimeter node. In a bad case, the *perimeter* may turn out to be the outer perimeter of the network, increasing the average producer cost by a large number. Structured replication improves consumer cost in GHTs at the expense of increasing producer cost several times. The producer cost in our double rulings scheme is better than that for the GHTs with replication, and worse when no replication is used. For the consumer cost, the average cost in our scheme is better than that for the GHTs without replication, and worse when 1-depth structured replication is used.

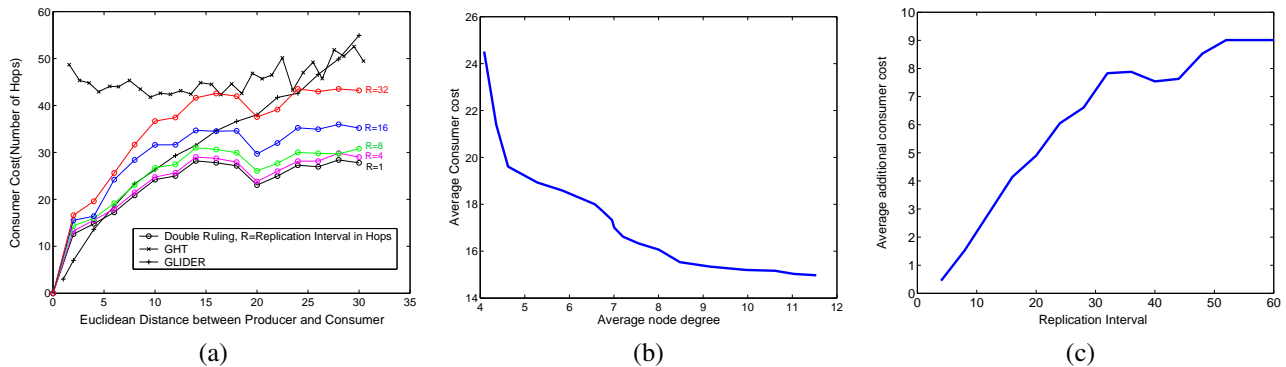
We simulated the effect of storing data at regular intervals (one node in every  $R$  hops stores the data) along the producer curve, and at  $h$  and  $\bar{h}$  with 100 pairs of randomly chosen producers and consumers. The retrieval process, after meeting the producer curve, travels to the nearest replication point to obtain data.

Figure 8(c) shows the relation between storage costs and average additional communication incurred by the consumer. When storage cost is decreased, that is, the interval of replication is increased, the additional cost of communication increases linearly with it. Beyond a certain value that depends on the size of the field, the replication interval does not affect the communication costs. This is because in such cases,  $h$  and  $\bar{h}$  become the actual points of retrieval in almost all cases.

Figure 8(a) shows the effect of different replication intervals on the distance sensitiveness of retrievals. Larger replication intervals increase consumer cost at all distances, but the scheme is still desirable compared to GHTs, and is better than the GLIDER based scheme except at very large intervals of replication. However, we should note that the GLIDER based scheme incurs high storage cost, by replicating on all branches of the finger tree.

### E. Non unit disk graphs

We applied our method to quasi unit disk graphs. In a *quasi unit disk graph* [18] with parameter  $d \leq 1$ , if two nodes are

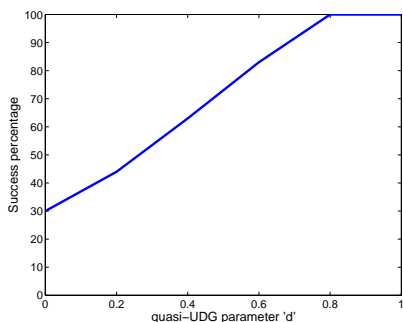


**Fig. 8.** (a) Distance between producer and consumer v.s. the average cost incurred by the consumer to retrieve information in GHTs, the GLIDER based scheme and in double ruling with different replication intervals. (b) Average degree Vs Average retrieval cost. (c) Additional communication required to retrieve actual data from nearest replication point for different replication intervals.

	Producer cost	Consumer cost
GHTs without structured replication	44.9	45.6
GHTs with 1-depth structured replication	172.5	17.4
Double Ruling with latitude consumer curve	77.5	29.0
Double Ruling with great circle consumer curve	77.5	34.6

**TABLE III.** Average producer and consumer costs for GHTs and double rulings.

within distance  $d$ , then the edge between the two exists, if they are at a distance more than 1, the edge does not exist; while for other distances, the existence of the edge is uncertain. The quasi-UDG graph was computed on the same network of about 4225 nodes. We varied the node density to keep the average node degree fixed at 9.5. In such a case, success rate was extremely high, above 96% for all values of  $d$ . Therefore, we do not plot this result. However, if density is constant, with varying  $d$ , the average degree varies between 5 at  $d = 0$  and 9.5 at  $d = 1$ . Figure 9 shows the percentage of successful queries against the parameter  $d$ . The discussion above suggests that the drop in performance is more due to the low degree than any particular property of the quasi-UDG.



**Fig. 9.** quasi-UDG parameter Vs success of delivery.

#### F. Load balancing

In a sensor network, it is important to have schemes that balance the load of operation across the nodes in the network. This avoids bottlenecks and improves the network lifetime. To evaluate load balancing properties of GHT and double ruling schemes, we simulated a scenario where there is a single producer, and 500 different consumers searching for the data.

Figure 10 (i) is a plot of loads across the network when using double rulings. The load of each node is measured by the

number of messages passing through it. We used the distance-sensitive retrieval mechanism. The load is seen to be well distributed across the network with no particular preference for occurrence of peaks signifying high loads. Further, the node suffering the highest traffic has a load of only 18. The highest load created by GHTs is an order of magnitude higher (126). Figure 10 (ii) shows the load distributions with GHTs. Nodes near the hashed location suffer much higher loads than the rest of the network, which is likely to result in a bottleneck slowing down the network, and also draining the batteries of these nodes rapidly.

#### V. CONCLUSION, DISCUSSION AND FUTURE WORK

In this paper we propose a simple replication mechanism that supports flexible retrieval mechanisms. In the future, we would like to combine the data replication mechanism with mobile data collectors such as data mules. Another direction is to investigate natural double rulings mechanisms in sensor field with irregular geometry.

##### A. Double rulings with mobile nodes

Information collection and delivery can explicitly use mobile nodes, such as data mules [15], [21], [32], [16], [28]. This is motivated by the observation that nodes around static sinks suffer from unbalanced traffic and energy consumption. Furthermore, controlled mobility helps to get around fundamental capacity problems imposed by insufficient sensor density. In an extreme case, such as a disconnected network, mobile nodes have to be involved to deliver information between two disconnected components. However, designing the moving trajectory for data mule is challenging. One obvious metric is to have the data mule travel a short distance. Finding the shortest path that visits all the communication ranges of the nodes with data is a traveling sales man problem and is NP-hard [2].

We observe that data mules can be naturally combined with double ruling approaches to shorten the traveling distance of the data mule, with a modest in-network storage and aggregation. A mobile node physically traveling along a consumer curve is able to retrieve all the data in the network. This substantially decreases the distance traveled by the data mule. If the network is uniformly deployed in a squared region of  $n$  nodes. The shortest traveling salesman path is roughly  $O(n)$  (visiting each node), but the double ruling curve has length roughly  $O(\sqrt{n})$ .

### B. Alternative double ruling schemes

In this paper we introduced the spherical double rulings scheme and compared with prior work on rectilinear double rulings. There are many ways to design two families of double rulings curves to support data storage and retrieval in a network, for example, by using concentric circles and radial lines, or by using gradient lines and iso-contours with an artificial single peak potential field. The choice of proper double rulings curves depends on application scenarios and requirements. As suggested in this paper, special properties of the double rulings curves may bring particular benefits for improved efficiency and load balancing.

**Networks of irregular shapes.** In this paper we focus on a double rulings mechanism for a nicely distributed sensor field. In the case that sensors are deployed in an irregular shape with holes, the double ruling curves may accumulate on the hole boundaries. Thus we will need to take the global geometry into consideration and define double ruling curves in a virtual coordinate system that adheres to the underlying network geometry. For example, in the virtual coordinate system defined by the medial axis of the sensor field [4], there are natural double rulings curves, those that are parallel to the medial axis and those perpendicular to the medial axis.

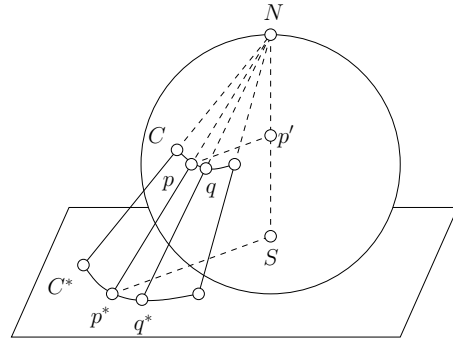
Another approach to apply double rulings mechanism in a sensor field with complex geometry is to partition the sensor field into nicely shaped components and construct double ruling curves for each piece. This is the approach taken in the GLIDER-based scheme [11], in which the double rulings scheme is in the virtual coordinate system defined by landmark distances. In that case a data mule collecting data from the network only needs to visit each piece and pays a traveling cost proportional to the minimum spanning tree connecting all the segmented pieces.

### C. Advanced hashing

An additional variance that is not discussed in this paper is the choice of data-centric hash functions. The choice of hash functions is orthogonal to the double rulings scheme. In this paper we had used a uniform random hash function. Advanced hashing schemes, such as the one used in DIFS [13] or any distance-sensitive hashing schemes [7] that preserve data proximity can be directly incorporated. For example, we may prefer to hash similar data types nearby that may facilitate efficient aggregated data retrieval. The discussion of advanced hashing mechanisms and their interaction with double rulings will be interesting future work but is beyond the scope of this paper.

## APPENDIX A PROOF OF THEOREM 2.1

*Proof:* The proof basically follows from the fact that projective transformation preserves cross ratio. Denote by  $C$  the segment along the circle on the sphere between  $p_1$  and  $p_2$  and  $C^*$  the segment along the projected circle between  $\rho(p_1)$  and  $\rho(p_2)$ . The length of  $C$  is  $d$ , the length of  $C^*$  is  $\ell$ . First we notice that there is a one-to-one mapping under  $\rho$  between points on  $C$  and points on  $C^*$ . Now  $d = \int_C dx$ , where  $dx$  is a miniature segment on  $C$ . Similarly,  $\ell = \int_{C^*} dx'$ , where  $dx'$  is the projection of  $dx$  in the plane. See Figure 11. Now we take



**Fig. 11.** The length of a segment of a circle on the sphere is bounded from the length of its projection in the plane.

a tiny segment  $pq$  on  $C$  with length  $dx \rightarrow 0$ ,  $dx = |pq|$ . The projection of  $pq$ , denoted by  $p^*q^*$  has length  $dx' = |p^*q^*|$ . Denote by  $N$  is the north pole,  $S$  is the south pole and  $p'$  is the projection of  $p$  on line segment  $NS$ . We now have

$$\frac{dx}{dx'} = \frac{|pq|}{|p^*q^*|} = \frac{|Np'|}{|NS|}.$$

This last equality can be argued as follows. The vector  $\vec{pq}$ , the tangent vector at  $p$  along  $C$ , can be decomposed as the sum of a latitude component  $v_1$  (parallel to the plane) and a longitude component  $v_2$  (derivative along the longitude curve through  $p$ ). Correspondingly  $v_2$  and  $v_1$  have projections as  $v_2^*$  along the direction  $p^*S$  and  $v_1^*$  perpendicular to  $v_2^*$ , respectively. By the one-to-one mapping we have  $\vec{p^*q^*} = v_1^* + v_2^*$ .

For the latitude component  $v_1$ ,  $v_1$  is parallel to  $v_1^*$ . We have  $\|v_1\|/\|v_1^*\| = |Np|/|Np^*| = |Np'|/|NS|$ , by the property of projection and similarity of triangle  $Npp'$  and  $NpS$ . For the longitude component one can verify that the equality holds as well. Specifically, take the plane defined by  $N, p, S$  and define  $S$  as the origin and  $Sp^*$  as the positive  $x$ -axis. Denote the coordinate of  $p^*$  is  $(x, 0)$ . Then the coordinate of  $p$  is  $(4r^2x/(x^2 + 4r^2), 2rx^2/(x^2 + 4r^2))$ . We compute the length of the tangent vector at  $p$ , which is  $4r^2/(x^2 + 4r^2) = |Np'|/|NS|$ . Thus  $\|v_2\|/\|v_2^*\| = |Np'|/|NS|$ . Putting them together we have

$$\frac{|pq|}{|p^*q^*|} = \frac{\|v_1 + v_2\|}{\|v_1^* + v_2^*\|} = \frac{|Np'|}{|NS|}.$$

Now, since  $p'$  has height ( $z$ -coordinate) at most  $k$ , we have

$$\frac{|Np'|}{|NS|} \geq \frac{2r - k}{2r} = \frac{1}{1 + \varepsilon}.$$

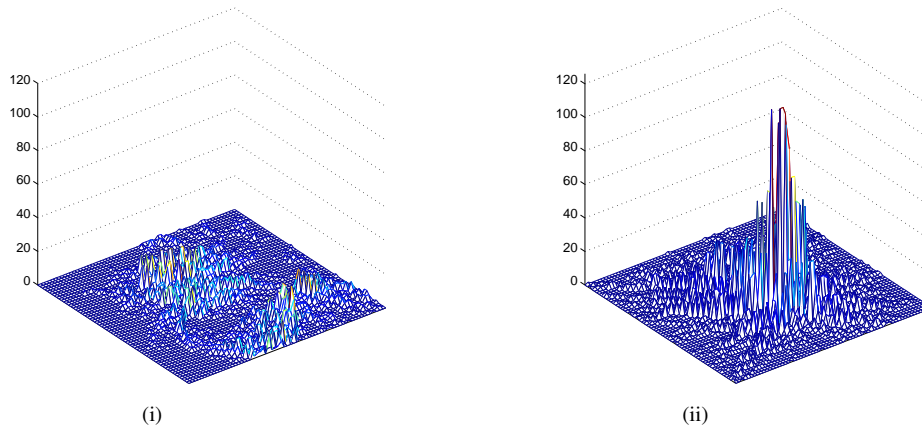


Fig. 10. (i) Load distribution in double rulings with 500 consumers and one producer with one data type. (ii) Load distribution in GHTs with 500 consumers and one producer with one data type.

Thus,  $d = \int_C dx \geq \int_{C^*} dx' / (1 + \varepsilon) = \ell / (1 + \varepsilon)$ . The theorem is true. ■

#### APPENDIX B PROOF OF LEMMA 2.6

*Proof:* Suppose the longitude curve  $C(p, h)$  through  $p$  and the north pole  $h$ . The latitude curve through  $q$  is denoted as  $L(q, h)$ .  $u$  is the closer intersection of  $C(p, h)$  and  $L(q, h)$  to  $q$ . Denote by  $k'$  the distance between  $q$  and  $u$  along  $L(q, h)$  and  $k$  the shortest distance from  $q$  to  $C$ . See Figure 12. We want to argue that  $k'/k \leq \pi/2$ .

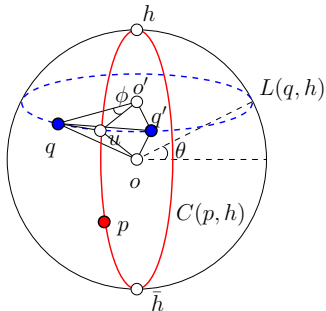


Fig. 12. Denote by  $k'$  distance between  $q$  and  $u$  along  $L(q, h)$  and  $k$  the shortest distance from  $q$  to  $C$ . Then  $k'/k \leq \pi/2$ .

We denote by  $o$  the center of the sphere and  $o'$  the center of the latitude circle  $L(q, h)$ . The angle between the line  $oq$  and the horizontal line is defined as  $\theta$ ,  $0 \leq \theta \leq \pi/2$ . Then the radius of  $L(q, h)$ ,  $r' = r \cos \theta$ , where  $r$  is the radius of the sphere.

Suppose the angle  $\angle qo'u = \phi$ .  $0 \leq \phi \leq \pi/2$ . The distance between  $q$  and  $u$  on  $L(q, h)$ ,  $k' = r'\phi = r\phi \cos \theta$ . Take  $q'$  as the mirror point of  $q$  reflected by the plane defined by  $C(p, h)$ . Since the minimum distance between two points  $q, q'$  must be along the great circle defined by  $o, q, q'$ , which intersects  $C(p, h)$  at point  $w$ . Thus by symmetry the minimum distance between  $q$  and  $C(p, h)$ , is exactly the distance between  $q$  and  $w$ ,  $d(q, w)$ . The Euclidean distance between  $q, q'$ ,  $|qq'| = 2r' \sin \phi = 2r \cos \theta \sin \phi$ . Thus  $\angle qow = \angle qoq'/2 = \arcsin(|qq'|/(2r)) = \arcsin(\cos \theta \sin \phi)$ .

Therefore,  $k = d(q, w) = r \angle qow = r \arcsin(\cos \theta \sin \phi) \geq r \cos \theta \sin \phi$ . The last inequality follows from the fact that  $\sin x \leq x$ .

The claim then follows:

$$\frac{k'}{k} = \frac{\phi \cos \theta}{\arcsin(\cos \theta \sin \phi)} \leq \frac{\phi}{\sin \phi} \leq \pi/2.$$

The function  $\phi / \sin \phi$  achieves its maximum value when  $\phi = \pi/2$ . ■

#### ACKNOWLEDGEMENT

We thank Qing Fang for sharing with us the code of [11] and her numerous help with running simulation comparisons with the scheme in [11]. We also thank the anonymous reviewers for their helpful comments in improving this paper.

#### REFERENCES

- [1] <http://www.greatduckisland.net/>.
- [2] E. M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Appl. Math.*, 55(3):197–218, 1994.
- [3] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 22–31, New York, NY, USA, 2002. ACM Press.
- [4] J. Bruck, J. Gao, and A. Jiang. MAP: Medial axis based geometric routing in sensor networks. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'05)*, pages 88–102, August 2005.
- [5] H. S. M. Coxeter. *Introduction to Geometry*. John Wiley & Sons, New York, 2nd edition, 1969.
- [6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on  $p$ -stable distributions. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, New York, NY, USA, 2004. ACM Press.
- [7] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on  $p$ -stable distributions. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, New York, NY, USA, 2004. ACM Press.
- [8] C. T. Ee, S. Ratnasamy, and S. Shenker. Practical data-centric storage. In *NSDI'06: Proceedings of the 3rd conference on Networked Systems Design & Implementation*, pages 24–24, Berkeley, CA, USA, 2006. USENIX Association.
- [9] Q. Fang, J. Gao, and L. Guibas. Locating and bypassing routing holes in sensor networks. In *Mobile Networks and Applications*, volume 11, pages 187–200, 2006.
- [10] 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'05)*, volume 1, pages 339–350, March 2005.

- [11] Q. Fang, J. Gao, and L. J. Guibas. Landmark-based information storage and retrieval in sensor networks. In *The 25th Conference of the IEEE Communication Society (INFOCOM'06)*, pages 1–12, April 2006.
- [12] S. Funke, L. Guibas, A. Nguyen, and Y. Wang. Distance-sensitive routing and information brokerage in sensor networks. In *Proc. IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'06)*, pages 234–251, July 2006.
- [13] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and S. Shenker. DIFS: A distributed index for features in sensor networks. In *Proceedings of First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 163–173, May 2003.
- [14] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. of the 6th Annual Int'l Conference on Mobile Computing and Networking (MobiCOM '00)*, pages 56–67, August 2000.
- [15] D. Jea, A. A. Somasundara, and M. B. Srivastava. Multiple controlled mobile elements (data mules) for data collection in sensor networks. In *IEEE/ACM Int'l Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 244–257, 2005.
- [16] A. Kansal, M. Rahimi, W. J. Kaiser, M. B. Srivastava, G. J. Pottie, and D. Estrin. Controlled mobility for sustainable wireless networks. In *IEEE Sensor and Ad Hoc Communications and Networks (SECON'04)*, pages 1–6, 2004.
- [17] 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.
- [18] F. Kuhn, R. Wattenhofer, and A. Zollinger. Ad hoc networks beyond unit disk graphs. *Wirel. Netw.*, 14(5):715–729, 2008.
- [19] 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.
- [20] X. Li, Y. J. Kim, R. Govindan, and W. Hong. Multi-dimensional range queries in sensor networks. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 63–75. ACM Press, 2003.
- [21] W. Lindner and S. Madden. Data management issues in periodically disconnected sensor networks. In *Proceedings of Workshop on Sensor Networks at Informatik*, 2004.
- [22] 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.
- [23] 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.
- [24] J. Munkres. *Topology, 2nd Ed.* Prentice Hall, 2000.
- [25] B. Nath and D. Niculescu. Routing on a curve. *SIGCOMM Comput. Commun. Rev.*, 33(1):155–160, 2003.
- [26] 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 and Applications*, pages 78–87, 2002.
- [27] P. Samuel. *Projective Geometry*. Springer-Verlag, New York, 1988.
- [28] R. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling a three-tier architecture for sparse sensor networks. In *IEEE SNPA Workshop*, pages 30–41, May 2003.
- [29] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin. Data-centric storage in sensornets. *SIGCOMM Comput. Commun. Rev.*, 33(1):137–142, 2003.
- [30] 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.
- [31] R. Szweczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 214–226, New York, NY, USA, 2004. ACM Press.
- [32] Z. Vincze and R. Vida. Multi-hop wireless sensor networks with mobile sink. In *CoNEXT'05: Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, pages 302–303, New York, NY, USA, 2005. ACM Press.
- [33] Y. Wang, J. Gao, and J. S. B. Mitchell. Boundary recognition in sensor networks by topological methods. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 122–133, September 2006.
- [34] 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.



**Rik Sarkar** is currently a Ph.D. Student at Stony Brook University. He completed his M.Tech. in Computer Science from Indian Institute of Technology, Bombay, India in 2005. His research interests are in distributed systems, sensor networks and geometric algorithms.



**Xianjin Zhu** received her M.S. degree in computer science from Stony Brook University, and her B.S. degree from Nanjing University in China. She is currently a Ph.D. candidate in computer science at Stony Brook University. Her research interests are sensor networks, ad hoc networks, algorithms and computational geometry.



**Jie Gao** received her Ph.D in computer science from Stanford University in 2004, and her BS degree from University of Science and Technology of China in 1999. She is currently an assistant professor at Stony Brook University. Her research interests are algorithms, ad hoc communication and sensor networks, and computational geometry.