Covering Space for In-Network Sensor Data Storage

Rik Sarkar* Wei Zeng† Jie Gao* Xianfeng David Gu*

*Department of Computer Science
Stony Brook University
Stony Brook, NY 11794
{rik,jgao,gu}@cs.sunysb.edu

†Department of Computer Science
Wayne State University
Detroit, MI 48202
zeng@wayne.edu

ABSTRACT

For in-network storage schemes, one maps data, indexed in a logical space, to the distributed sensor locations. When the physical sensor network has an irregular shape and possibly holes, the mapping of data to sensors often creates unbalanced storage load with high data concentration on nodes near network boundaries. In this paper we propose to map data to a *covering space*, which is a tiling of the plane with copies of the sensor network, such that the sensors receive uniform storage load and traffic. We propose distributed algorithms to construct the covering space with Ricci flow and Möbius transforms. The use of the covering space improves the performance of many in-network storage and retrieval schemes such as geographical hash tables (GHTs) or the double rulings (quorum based schemes), and provides better load balanced routing.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network protocols—*Routing protocols*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

General Terms

Algorithms, Design, Theory

Keywords

In-network Storage, Covering Space, Möbius transforms, Ricci Flow, Conformal Mapping, Sensor Networks

1. INTRODUCTION

Prior research on sensor networks have proposed the 'data-centric' notion [14, 23] for sensor network design. The generation, collection, processing, storage and retrieval of sensor data are the most critical functions around which the network protocols should be designed. As the state of the art, networks in the size of thousands of sensor nodes are deployed [1, 2, 20] with the target size of hundreds of thousands in the next few years. As networks grow large in size, centralized data collection has a fundamental bottleneck at nodes near the sink. Distributed in-network data storage in which there is no single sink is more desirable for its robustness.

For distributed in-network storage, data is mapped to rendezvous sensors for storage and processing. Such a mapping is often obtained by considering data in a logical space with indices mapped to geographical locations. For example, in geographical hash table (GHT) [23], data keys are hashed to a random geographical location in the sensor field and the sensor node closest to the hashed location is denoted as the home node and stores the data. In DIM [18], data in a multi-dimensional attribute space is mapped to the sensor field by using a quad-tree, such that data with nearby indices are mapped to physically nearby zones, in order to support range queries. Variations of quadtrees have also been used in other schemes to organize data and the corresponding storage [8, 9, 13].

The mapping from the logical data space to the physical sensor field, done in a straightforward manner, often leads to high data concentration on nodes near network boundaries, simply for the reason that they are adjacent to empty or low density regions. In GHT, for example, all the data hashed inside a hole will eventually be mapped to the nodes on the hole boundary, that get a higher storage load. When GHT exploits the nodes on the boundary of a planar face to also store the data (for robustness to node failures), the load imbalance is even higher. Similarly for DIM, the node near a zone empty of nodes will substitute to store the data. Nodes near hole boundaries store more data and carry more traffic.

When the sensor field is irregular, many geometric based data storage and retrieval schemes run into problems. As another example, double rulings, or quorum based schemes, store data on a curve and retrieve data along another curve. As long as the data retrieval curves intersect with the data storage curve, one can successfully discover the desired data. When a sensor field has a regular shape (a square region or a disk, for example), one can design the storage/retrieval curves as the horizontal/vertical lines [19, 26, 28], or proper circles (great circles through a stereographic mapping) [25]. Both of them may get stuck at network boundaries. Of course one can use various hole bypassing techniques to get around the holes [6, 16]. This may also lead to higher storage and traffic load on the hole boundaries – the same problem encountered earlier.

The imbalance of storage or traffic load adversely affects the system performance. On one hand, the nodes with high load are bottleneck nodes. They carry out more tasks than average. If the nodes are battery powered, this means the highly loaded nodes would run out of battery sooner. When these heavily used nodes are on hole boundaries the problem is worse, as holes are enlarged and the network may be disconnected prematurely. In addition, the nodes with high traffic load denote the bottleneck of communication. Spatial diversity is not best optimized to avoid wireless interference, leading to lower network throughput.

Our contribution. We propose to solve the imbalance of storage and traffic load in an irregular sensor network by 'uniformizing' the sensor field shape. As the logical data space is often regular, we make the sensor field regular as well — irregular shape is turned into circular, and holes are filled up. We propose to create a *covering space* of the sensor network, which is a tiling of the space with transformed copies of the sensor networks. Data hashed to a geographical location inside a hole is actually mapped to another copy of the sensor field. Similarly, with a regular shape, previously proposed double rulings scheme can be applied to irregular network with almost zero modification. Thus our *network regulation* technique provides a generic solution for data storage problems in an irregular network, and greatly extends the application scope of existing schemes. See Figure 3 for an example of the original sensor field and the covering space.

We achieve this by using Ricci flow and conformal Möbius transform. In a previous work of ours [24], we have shown how to embed a sensor network such that all the boundaries (both outer boundary and inner hole boundaries) are circular. In particular, we first extract a triangulation of the sensor network. We modify the metric (edge lengths) in the triangulation such that the interior vertices have zero curvature (thus being flat), and vertices on the same boundary have proper curvature making the boundary a circle. Such deformation is achieved by the Ricci flow algorithm, which is a distributed, gossip-style iterative algorithm. Each node locally calculates its curvature and modifies the adjacent edge lengths with a rate proportional to the difference to the target curvature. Such an operation is proved to converge uniquely to the metric with the target curvature, the convergence is exponentially fast. In the process, curvature is diffused in the same way as heat diffusion.

The new idea in this paper is to use the embedding obtained from the Ricci flow algorithm and *fill up the holes*. Suppose the network has k (circular) holes. For each interior hole C_i, we take a Möbius transform that essentially 'reflects' the network inward with respect to C_i. This Möbius transform is conformal and maps circles to circles. Thus, C_i becomes the outer boundary of the reflected network with all the nodes mapped inside it. This partially fills up the hole C_i, except that there are k smaller circular holes. Now we can continue such transforms so that all the holes are eventually filled up, with infinitely many transformed copies of the original sensor field. The collection of Möbius transforms used to generate these mappings is captured in the *Schottky group*. Thus, one does not need to precalculate any of these mapping and is able to generate the reflections on the fly when necessary.

The generation of the covering space as described above asks for infinitely many transformed copies to completely cover the space. We show that for any practical applications only $O(\log 1/\varepsilon)$ copies are necessary, where ε is the threshold of the size of a hole. Indeed, we prove that the total area of the holes shrinks by a fraction after each Möbius transform, and is reduced exponentially fast. When the holes are tiny, the chance that data is hashed to be inside a hole is very small and can be omitted. Similarly, the chance that a double ruling curve hits the boundary of a tiny hole is negligible. When it does happen, we can get around the hole by following the greedy routes along the circular hole boundaries. In our simulations only 5 reflections are necessary and for some applications 2 levels of reflections give the best result.

With the regulation of the network shape by conformal Möbius transforms, we can improve the performance of various data storage schemes.

- *GHT.* When a piece of data is hashed to a geographical location p inside a hole, in the original GHT scheme, it is allocated to the sensor node whose Voronoi cell contains p. Nodes on the boundary have larger Voronoi cells and share higher load. With the covering space, the area inside the hole is shared by the *entire* network, eliminating fundamentally the storage and traffic overhead on the holes boundaries.
- *Double rulings.* Double rulings design can be directly applied on the covering space. When a curve hits a hole boundary C_i, it then enters another copy of the network mapped to the interior of C_i. Equivalently, in the original embedding, the curve 'reflects' on the hole boundary. The intersection properties are still maintained with the conformal Möbius transforms.
- *Load balanced greedy routing.* The embedding generated by the Ricci flow algorithm allows greedy routing to work with delivery guarantee, as greedy routing can not get stuck at circular holes. However, such greedy routes still tend to hug the hole tightly causing high traffic load on the boundary nodes. Instead, we can execute the greedy routing in the covering space. Instead of getting around the hole by following the circular hole boundary, one can 'enter' the hole to route in another copy of the network, effectively reflect on the hole boundary. Thus the boundary nodes are not used as often, improving the load balancing. The greedy routing can be used in combination with the GHT scheme to deliver and retrieve data from the hashed location.

In summary, the covering space universally improve the load imbalance in data storage and routing in an irregular network. Essentially, in the covering space the holes are filled up so there are no 'boundaries'. The nodes on the boundary are now treated in the same way as the other nodes with respect to data storage or relay routing.

In the following of the paper we first present the mathematics of the conformal Möbius transform, the Scottky group, and the covering space. We then present the use of the covering space in applications such as GHT, double rulings and greedy routing with simulation results.

2. THEORETIC BACKGROUND

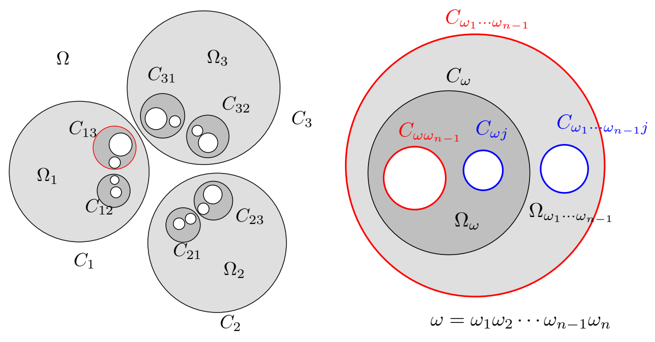
This section focuses on the theoretic background of Möbius transformation and reflections to generate the covering space. We refer readers to [5] and [22] for further details.

2.1 Conformal Mapping for Multiply Connected Domain

Let (S_1, g_1) and (S_2, g_2) be two surfaces with Riemannian metrics g_1, g_2. A mapping $\phi : S_1 \to S_2$ is called a *conformal map* (*angle preserving map*), if the intersection angle of any two curves are preserved.

A planar domain D of connectivity n is called a circular domain, if all its n boundaries are circles. It is known from conformal geometry that any genus zero multiply connected planar domain can be mapped to a circular domain by conformal maps. The different circular mappings of a given planar domain differ by Möbius transforms [4, 22].

One way to compute the conformal mapping from a surface to a circular domain is to use Ricci flow, as introduced in [15, 24]. Given a multiply connected domain Ω with m interior holes, denoted as $\partial\Omega = \{\gamma_1, \gamma_2 \cdots, \gamma_m\}$, by Ricci flow, we can construct



(a) Circular reflection (b) Naming convention

Figure 1. Circular Reflection and naming convention.

a conformal map $\phi : \Omega \to \mathbb{C} \cup \{\infty\}$ to a circle domain, such that each $\phi(\gamma_j)$ is a circle,

$$\partial[\phi(\Omega)] = \{C_1, C_2, \ldots, C_m\}, C_j = \{z : |z - \mathbf{c}_j| = r_j\}.$$

where $j = 1, 2\cdots, m$. Examples can be found in Figure 3, 7 and 8.

2.2 Möbius Transform and Circular Reflection

A *Möbius transformation* is a map that maps a complex plane to itself, represented by $f(z) = \frac{az+b}{cz+d}$, where a, b, c, d are four complex numbers satisfying $ad-bc = 1$. A Möbius transformation is a conformal map and maps circles to circles. A special case of Möbius transformation

$$\rho_C(z) := \mathbf{c} + \frac{r^2}{\bar{z} - \bar{\mathbf{c}}}. \quad (1)$$

is a *circular reflection* that maps the points inside a circle C with center c and radius r to the points outside C, and vice versa. For a circular domain Ω with interior circular holes $C_1, C_2, \cdots, C_m$, we denote ρ_{C_j} by ρ_j for $C = C_j$. ρ_j essentially fills up the hole C_j by reflecting the points out of C_j. We use such circular reflections to fill up the holes in a sensor network.

An example. Figure 1 (a) shows an example of a triply connected circular domain Ω with boundary $\partial\Omega = \{C_1, C_2, C_3\}$. Reflect Ω through C_k to get

$$\Omega_k = \rho_k(\Omega).$$

The circle C_j is reflected by ρ_i to be circle C_{ij},

$$C_{ij} = \rho_i(C_j), i \neq j.$$

Ω_k is a bounded domain with outer boundary C_k and 2 circular inner boundaries. The boundary of Ω_k's in Figure 1 (a) are $\partial\Omega_1 = \{C_1, C_{12}, C_{13}\}$, $\partial\Omega_2 = \{C_2, C_{21}, C_{23}\}$, $\partial\Omega_3 = \{C_3, C_{31}, C_{32}\}$.

Now Ω_1 has two small holes C_{12} and C_{13}. We reflect Ω_1 with respect to each of the interior hole. Thus we have the reflected domains in the next level. In general,

$$\Omega_{ij} = \rho_{ij}(\Omega_i), 1 \leq i, j \leq 3, i \neq j.$$

The new boundary circles are

$$C_{121} = \rho_{12}(C_1), C_{123} = \rho_{12}(C_{13}), \partial\Omega_{12} = \{C_{12}, C_{121}, C_{123}\}$$

$$C_{131} = \rho_{13}(C_1), C_{132} = \rho_{13}(C_{12}), \partial\Omega_{13} = \{C_{13}, C_{131}, C_{132}\}$$

The boundaries of $\Omega_{21}, \Omega_{23}, \Omega_{31}$ and Ω_{32} are similar.

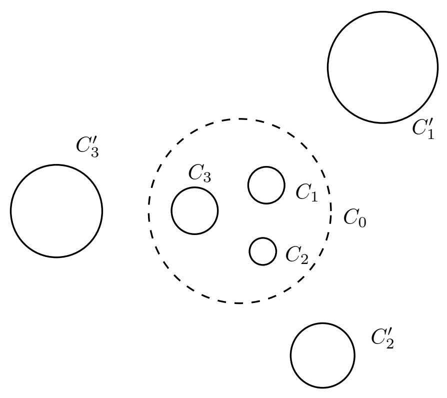


Figure 2. Schottky Group generators.

In general, for a circular domain with m interior holes, the reflected regions and circles are labeled with multi-indices

$$\omega = \omega_1\omega_2\cdots\omega_q, 1 \leq \omega_j \leq m, \omega_k \neq \omega_{k+1}, 1 \leq k \leq q-1.$$

A reflected domain is defined by the reflections following the indices.

Definition 2.1. *The set of multi-indices of length q ($q > 0$) is denoted*

$$\sigma_q = \{\omega_1\omega_2\cdots\omega_q : 1 \leq \omega_j \leq m, \omega_k \neq \omega_{k+1}, 1 \leq k \leq q-1\},$$

and $\omega_0 = \emptyset$.

As shown in Figure 1 (b), if $\omega \in \sigma_q, q > 1$, the circular domain

$$\Omega_\omega = \rho_\omega(\Omega_{\omega_1\omega_2\cdots\omega_{q-1}})$$

has exterior boundary C_ω and $m-1$ interior boundary circles

$$C_{\omega\omega_{q-1}} = \rho_\omega(C_{\omega_1\omega_2\cdots\omega_{q-1}}),$$

and

$$C_{\omega j} = \rho_\omega(C_{\omega_1\omega_2\cdots\omega_{q-1}j}), j \neq \omega_{q-1}, \omega_q.$$

There are $m(m-1)^{q-1}$ elements in σ_q, at the level q, and $m(m-1)^{q-1}$ circles.

2.3 Schottky Groups

Taking reflections of a circular domain with respect to the circular holes to the limit will eventually have all the holes 'filled up'. This is shown by using the Schottky groups, as described below.

Suppose C_j is the circle $|z - \mathbf{c}_j| = r_j^2$, C_0 is the unit circle $|z| = 1$, denote $\rho_0(C_j)$ as C'_j (see Figure 2). The Möbius map

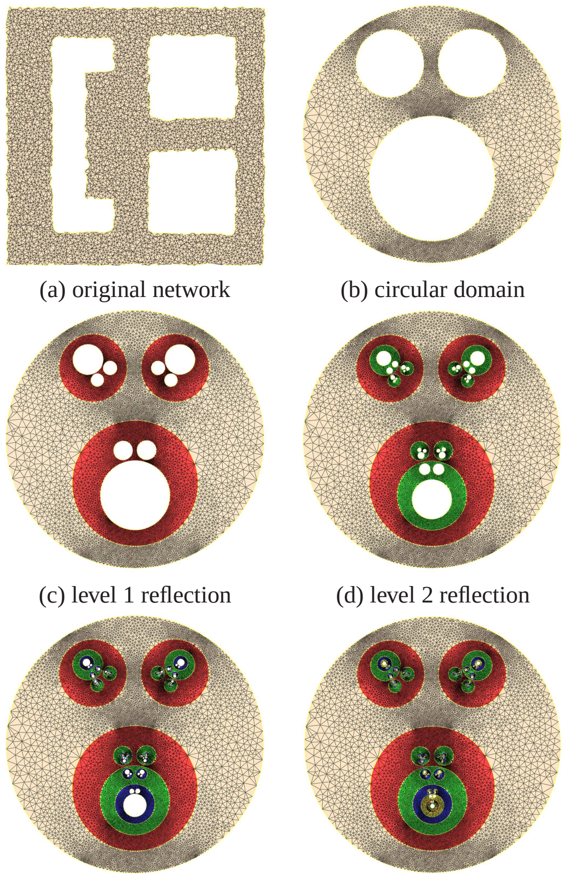
$$\theta_j(z) = \mathbf{c}_j + \frac{r_j^2 z}{1 - \bar{\mathbf{c}}_j z}$$

maps the exterior of C'_j to the interior of C_j (and C'_j to C_j).

The Schottky group Θ is defined to be the infinite free group of Möbius mappings generated by compositions of the $2m$ basic Möbius maps $\{\theta_j | j = 1, \cdots, m\}$ and their inverses $\{\theta_j^{-1} | j = 1, \cdots, m\}$. Consider the unbounded region Ω of the plane exterior to the $2m$ circles $\{C_j | j = 1, \cdots, m\}$ and $\{C'_j | j = 1, \cdots, m\}$. The union of copies of Ω generated by the $\theta \in \Theta$ is denoted as

$$\Theta(\Omega) := \bigcup_{\theta \in \Theta} \theta(\Omega).$$

This work is based on the following fundamental theorem of Schottky groups.


(a) original network (b) circular domain

(c) level 1 reflection (d) level 2 reflection

(e) level 3 reflection (f) level $n \geq 4$ reflection

Figure 3. 4-level circular reflections for a 3-hole sensor network with 5492 nodes. The initial network (a) is conformally mapped to the circular domain (b). The level 1 reflection is in red color in (c), level 2 reflection is in green in (d), level 3 reflection is in blue in (e), level 4 reflection is in yellow in (e).

Theorem 2.2. *The complement set of* $\Theta(\Omega)$

$$\Theta(\Omega)^c := \mathbb{C} - \Theta(\Omega)$$

is a Cantor set of zero measure.

The detailed discussion can be found in [5], [22] and [4].

θ_j can be generated by two circular reflections: first the exterior of C_j' is reflected through the unit circle C_0, then the interior of C_0 is reflected through C_j. The composition of these two reflections is θ_j. In this work, we use circular reflections instead of explicitly using Schottky group.

2.4 Shrinkage Estimation

Asymptotically the whole plane can be covered by the copies of the network using Schottky transformation. But in practice, only a finite number of reflections can be used. Therefore, we need a precise estimation of the size of holes after n levels of reflections. In the following, we give the estimation of the area shrinkage of the holes. We follow the method in [22] and [4].

As shown in figure 4, Ω is a bounded double connected domain on the complex plane $\mathbb{C}$, with exterior boundary Γ_0 and interior boundary Γ_1, $\partial\Omega = \Gamma_0 - \Gamma_1$. There exists a conformal map $\phi : \Omega \to \mathbb{D}$, where $\mathbb{D}$ is a circular domain, with inner radius μ_{01} and outer radius 1,

$$\mathbb{D} = \{z \in \mathbb{C} : \mu_{01} < |z| < 1, \}.$$

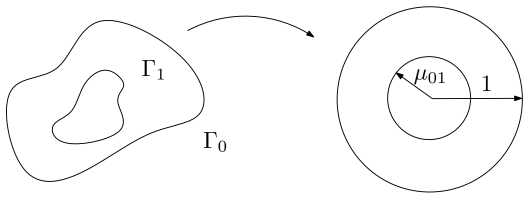
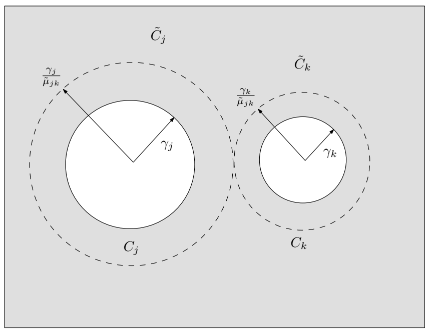


Figure 4. Conformal modulus μ_{01} of a doubly connected domain.



$\tilde{C}_j$
$\tilde{C}_k$
$\frac{\gamma_j}{\tilde{\mu}_{jk}}$
$\frac{\gamma_k}{\tilde{\mu}_{jk}}$
γ_j
γ_k
C_j
C_k

Figure 5. Separation modulus of a doubly connected circular domain.

We call μ_{01} the conformal modulus of the original domain Ω.

Definition 2.3. *The separation modulus for two circles C_j, C_k is defined as*

$$\tilde{\mu}_{jk} := \frac{\gamma_j + \gamma_k}{d_{jk}} < 1, j \neq k, 1 \leq j, k \leq m, \qquad (2)$$

where γ_j and γ_k are the radii of C_j, C_k respectively, and d_{jk} is the distance between the centers of C_j, C_k.

The separation modulus of the region is given by

$$\Delta := \max_{i,j,i\neq j} \tilde{\mu}_{ij}.$$

As shown in Figure 5, suppose $\tilde{C}_j$ is the circle with the center $\mathbf{c}_j$ and radius $\frac{\gamma_j}{\Delta}$, then $\frac{1}{\Delta}$ is the smallest magnification of the m circles, such that at least two $\tilde{C}_j$'s just touch.

The following lemma shows that the separation modulus is bounded by conformal modulus, the proof can be found in the Appendix.

Lemma 2.4. *The conformal modulus is the lower bound of the separation modulus:*

$$\mu_{jk} < (\tilde{\mu}_{jk})^2 \leq \Delta^2.$$

Theorem 2.5. *At level $q+1$, the total area of holes is*

$$\sum_{\omega \in \sigma_{q+1}} S(\tilde{C}_\omega) \leq \Delta^{4q} \sum_{i=1}^{m} S(\tilde{C}_i), \qquad (3)$$

where $S(C_i)$ is the area inside the circle C_i.

This theorem shows that the total area of the holes is reduced exponentially fast. Thus, after $-\log\varepsilon$ number of levels, each hole has a maximum area of ε. This shows that only a small number of levels is needed in practice. The proof of the theorem is put in the Appendix.

3. ALGORITHMS

For a sensor network, we compute the covering space up to level q (for a constant q typically) in the following steps.

1. Extract a triangulation of the network.
2. Apply a distributed Ricci flow algorithm as in [24] to embed the triangulation T such that T is a circular domain — it is embedded in the plane with each hole (a non-triangular face) embedded on a circle.
3. With the circular domain T we apply circular reflections to compute the covering space.

Figure 3, 7 and 8 demonstrate the pipeline of the algorithm.

3.1 Network Triangulation and Distributed Ricci Flow

Given a communication graph, we extract a planar graph from it. All non-triangular faces are treated as network holes. Algorithms for such purpose have been developed and are briefly reviewed below.

In our previous paper [24], a local, distributed algorithm has been developed to obtain a triangulation from the connectivity graph. The idea is to compute the *restricted Delaunay graph (RDG)* [10], i.e., a planar graph containing all Delaunay edges of length no greater than 1. The RDG can be computed by the nodes locally when the communication graph follows a quasi-unit disk graph (q-UDG)[1] of parameter $\alpha \leq \sqrt{2}$. The requirement of a quasi-UDG is to ensure that crossing edges can be detected locally and handled properly.

Funke *et al.* [7] developed a location-free triangulation algorithm by using landmarks and combinatorial Delaunay graph. The idea is to select a set of nodes as landmarks. The landmarks flood with a restricted range such that every node identifies the closest landmark a and is grouped to the Voronoi cell of a. A planar graph is computed on the landmarks by connecting the landmarks a, b that have 2-hop wide 'channel' of only nodes within cells of a, b. The authors showed that this graph is planar when the communication model follows a quasi-UDG of parameter $\alpha \leq \sqrt{2}$.

The two algorithms above both require a quasi-UDG model and thus does not work when a sensor network does not follow the quasi-UDG assumption. The following algorithms compute planar graphs without such assumptions.

Kim *et al.* [12,17] addressed the problem that planarization techniques using relative neighborhood graph or Gabriel graph fail when the communication model does not comply to the unit disk graph assumption. They developed a cross link detection protocol to probe each link, detect and remove possible crossings with other links. The resulting graph is combinatorially planar.

Zhang *et al.* [29] developed a location-free algorithm to extract a planar subgraph from the connectivity graph. The main idea is to planarize adjacent layers of a shortest path tree. Again this method does not require a unit disk graph model or quasi-UDG model.

All these algorithms above can be used in our method. In our implementation, we have used both the restricted Delaunay graph approach [24] and the landmark based triangulation approach [7]. But all the other schemes can also work well with our framework. In the worst case when a triangulation is not available, for example, when crossing edges are introduced, the result of the Ricci flow algorithm is theoretically unpredictable.

[1]In a *quasi unit disk graph* with parameter $\alpha \geq 1$, if two nodes are within distance $1/\alpha$, an 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.

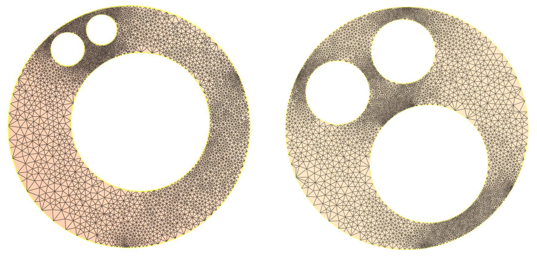


Figure 6. Improve separation modulus by Möbius transformation.

Last we remark that the obtained planar graph should represent a manifold with holes. In case of degeneracies (e.g., two holes sharing a single path, or dangling paths), virtual nodes and edges are introduced to resolve the issue. The details are shown in our previous work [24].

3.2 Circle Estimation

For a circular domain T, for each boundary γ_k, we need to estimate the circle $C_k(c_k, r_k)$. We take three consecutive nodes $\{z_1, z_2, z_3\}$ on the hole boundary to form a triangle, the circle $C_k(c_k, r_k)$ is the circumcircle of the triangle. Its center is

$$c = \frac{|z_1|^2(z_2 - z_3) + |z_2|^2(z_3 - z_1) + |z_3|^2(z_1 - z_2)}{z_1(\overline{z_3} - \overline{z_2}) + z_2(\overline{z_1} - \overline{z_3}) + z_3(\overline{z_2} - \overline{z_1})} \quad (4)$$

and its radius is $r = |z_1 - c|$. The derivation of the equation above can be found in [5].

Since the circle can be computed by any three adjacent boundary nodes, the computation of the circular hole equation can be done locally at each boundary node. The computation only involves a constant number of algebraic operations.

3.3 Separation Modulus Optimization

From the theoretical result, we can see that the total area of circular holes shrink to zero exponentially fast. The convergence rate is governed by the separation modulus Δ. In order to make the holes as small as possible, we can find an optimal Möbius transformation, that minimizes the separation modulus. In some sense, this transformation will map the holes to be as 'well-separated' as possible. We remark that this optimization step is optional.

A Möbius transformation preserving the unit disk is given by

$$\phi_{\theta,z_0}(z) = e^{i\theta}\frac{z - z_0}{1 - \bar{z}_0 z}, |z_0| < 1,$$

which maps circles to circles. $\phi_{\theta,z_0}(C_k)$ is still a circle, whose center and radius can be computed from

$$\{\phi_{\theta,z_0}(A), \phi_{\theta,z_0}(B), \phi_{\theta,z_0}(C)\}$$

using formula 4, where A, B and C are three points on the original circle. The rotation part $e^{i\theta}$ doesn't affect the separation modulus, in practice, we always set the rotation angle θ to 0. Let $\mu(C_j, C_k) = \tilde{\mu}_{jk}$ be the separation modulus between circles C_j, C_k as in formula 2. Define

$$\Delta(z_0) := \max_{j \neq k} \mu(\phi_{0,z_0}(C_j), \phi_{0,z_0}(C_k)).$$

The optimization problem is formulated as:

$$min_{|z_0|<1}\Delta(z_0).$$

In practice, we use gradient descent method to solve this non-linear optimization. See Figure 6 for an example.

To run this optimization step in a sensor network, we only need the knowledge of the center and radii of the circular holes. One node can pull the information of all circular holes together and run the optimization to get the Möbius transformation, which is then disseminated to all nodes in the network. The nodes apply the Möbius transformation to obtain the new mapping.

3.4 Circular Reflection

In a circular domain we perform circular reflection of the network. In our implementation, we use a Hermitian matrix H, $H^T = \overline{H}$ and $\det H < 0$ to represent a circle with center $\mathbf{c}$ and radius r. Suppose

$$H(\mathbf{c}, r) = \begin{bmatrix} a & b \\ \bar{b} & c \end{bmatrix}$$

Then the circle equation represented by $H(\mathbf{c}, r)$ is given by

$$0 = \begin{bmatrix} \bar{z} & 1 \end{bmatrix} \begin{bmatrix} a & b \\ \bar{b} & c \end{bmatrix} \begin{bmatrix} z \\ 1 \end{bmatrix}, \qquad (5)$$

where the center is $\mathbf{c} = -\frac{b}{a}$ and the radius $r = \frac{\sqrt{|b|^2 - ac}}{|a|}$. For a circle $|z - c| = r$, the corresponding Hermitian matrix format is

$$0 = \begin{bmatrix} \bar{z} & 1 \end{bmatrix} \begin{bmatrix} 1 & -c \\ -\bar{c} & |c|^2 - r^2 \end{bmatrix} \begin{bmatrix} z \\ 1 \end{bmatrix}. \qquad (6)$$

Orientation preserving Möbius transformation on $\mathbb{C} \cup \{\infty\}$

$$m(z) = \frac{\alpha z + \beta}{\gamma z + \delta}$$

is represented as a matrix

$$M = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}, \alpha, \beta, \gamma, \delta \in \mathbb{C}, \qquad (7)$$

its inverse is given by

$$M^{-1} = \begin{bmatrix} \delta & -\beta \\ -\gamma & \alpha \end{bmatrix}.$$

The reflection through a circle C is represented as

$$(\rho_C) = \begin{bmatrix} c & r^2 - |c|^2 \\ 1 & -\bar{c} \end{bmatrix}. \qquad (8)$$

Thus, the composition of Möbius transformations are represented as matrix multiplications. A Möbius transformation m maps a circle H to a circle. The Hermitian matrix representation of the image circle is given directly by

$$\overline{M^{-1}}^T H M^{-1}.$$

For orientation reverse Möbius transformation

$$m(z) = \frac{\alpha \bar{z} + \beta}{\gamma \bar{z} + \delta},$$

the matrix representation of the transformed circle is

$$M^{-T} H \overline{M^{-1}}.$$

Therefore, all the computations are carried out using complex matrix multiplication. Notice that the matrices H and M are only two by two matrices. So the matric multiplication and the Möbius transformation can both be done with a constant number of algebraic operations.

The reflection ρ_C through a circle C is given by the analytic formula 1.

The reflection process is recursive. The naming conventions for all the circles C_ω and all the reflections C_ω have been explained in details in subsection 2.2, where ω is a a word in the multi-index set σ_n in definition 2.1. Suppose we are given a multi-index word $\omega = \omega_1 \omega_2 \cdots \omega_n$, the recursive algorithms for computing the reflection ρ_ω and the circle C_ω (represented as matrices) are as follows.

```
Algorithm 3.1: MATRIXOFCIRCLE(word $\omega$)

if $|\omega| = 1$
  then return $(H(C_{\omega_1}))$ //Eqn.6.
Matrix $M$ = reflection($\omega_1 \omega_2 \cdots \omega_{n-1}$);
if $\omega_n = \omega_{n-2}$
  then Matrix $H$ = circle($\omega_1 \omega_2 \cdots \omega_{n-2}$);
  else Matrix $H$ = circle($\omega_1 \cdots \omega_{n-2} \omega_n$);
return $(M^{-T} H \overline{M^{-1}})$
```

```
Algorithm 3.2: MATRIXREFLECTION($\omega$)

Matrix $H$ = circle($\omega$);
compute center and radius $(\mathbf{c}, r)$ from $H$; //Eqn.5
return $(M(\rho(\mathbf{c}, r)))$; //Eqn.8
```

3.5 Computation and Communication Costs

The communication cost of the scheme is analyzed for each component. In the first step of triangulation extraction, the algorithms used to extract a triangulation as in [7,24] are localized algorithms. Each node only requires the knowledge of nodes in a constant size neighborhood. For most practical networks with constant average node degree, the total number of messages required is $O(n)$. The Ricci flow algorithm is an iterative algorithm with all nodes adjusting local metrics and local curvatures. The curvature error decreases exponentially fast. Therefore, the number of steps to reach the desired curvature error bound ε is given by $O(-\frac{\log \varepsilon}{\delta})$, where δ is the step size in the Ricci flow algorithm. The total communication cost is thus $O(-\frac{n \log \varepsilon}{\delta})$. The computation of network reflections are done locally in an on-demand manner. When a message hits a node on the boundary, depending on the number of levels of reflections required, the node can locally compute the reflection transform. The indices of the reflections are attached to the message. No additional communication is needed. Thus the communication cost in theory is linear in the number of nodes. In practice, the Ricci flow algorithm is the dominating factor of the communication cost.

4. APPLICATIONS

4.1 Geographical Hash Tables

We apply the covering space with geographical hash table (GHT) [23] for storing data in the network. Data is indexed with a key. Each data item x is hashed to a geographical location by using a random hash function $h(x) = g'$. The producer of the data item delivers the data towards the location g' using geographical routing (GPSR [16] in particular). If GPSR can not find a node right at the hashed location, it will eventually enter the face routing mode, by following the edges of a planar face f enclosing g'. Such face routing will necessarily fail to find the destination g' and return to the first node when the message enters this face. At this point the algorithm stops. The node on the face f closest to g' is denoted as the home node. The face f is denoted as the home perimeter. A perimeter refresh protocol is used to maintain the home perimeter when there are node or link failures. Except the home node, the

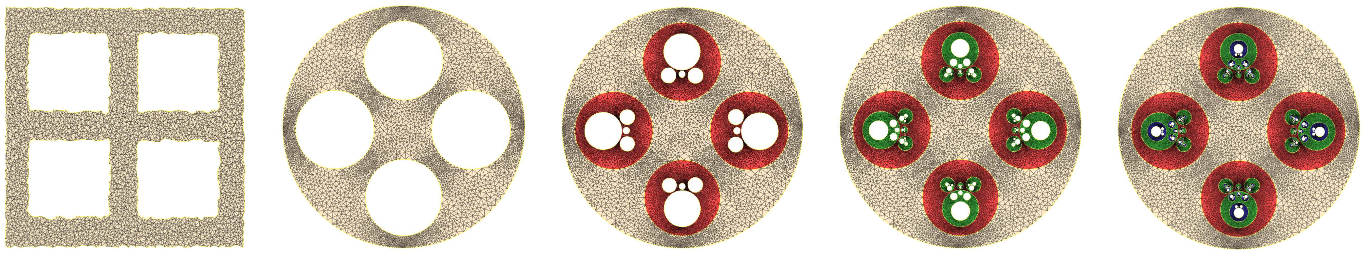


Figure 7. 3-level circular reflections for a 4-hole network with 4764 nodes.

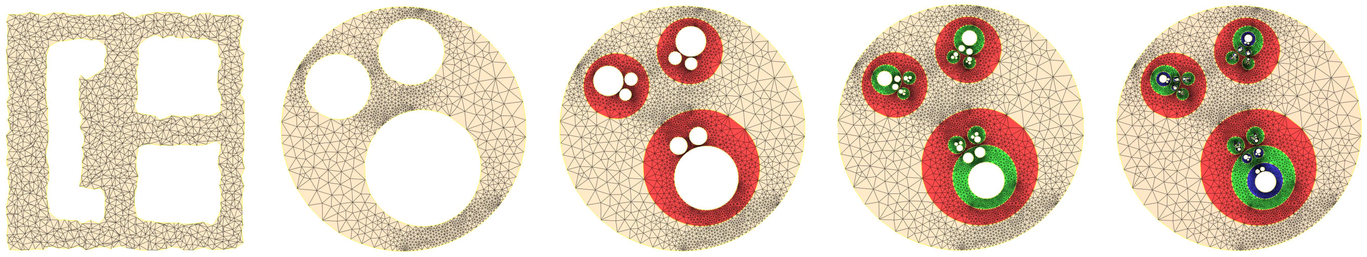


Figure 8. 3-level circular reflections for a 3-hole network with 1376 nodes.

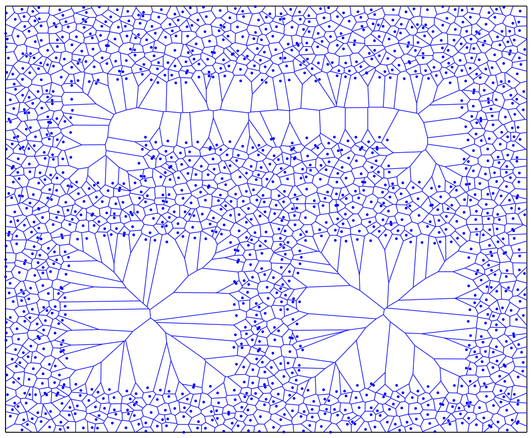


Figure 9. Voronoi diagram for the network in Figure 8 .

other nodes on the home perimeter are called replica nodes. The home node stores the data. The replica nodes may also hold data, to improve the system robustness to failures. GHT also has a hierarchically structured replication scheme where multiple hash images are used. In our case as we examine the influence of the network irregularity to the storage balancing, we use the basic scheme only.

A node p is the home node for all the hashed locations inside its Voronoi cell (which contains all the points closest to p than all other nodes). Thus the storage load of a node is directly proportional to the area of its Voronoi cell. It can be seen that the nodes near a hole has a large Voronoi cell and thus are allocated more data compared with the average load. An example is shown in Figure 9. When the replica nodes also store data, the storage load on boundary nodes is even higher, as the hole creates a large perimeter face such that all the data on the nodes of this face are shared.

One way to deal with the high concentration of data on the nodes in sparse region or near network holes is to use rejection sampling [27]. In particular, we select a random geographical location g' and round to the closest sensor node g. But we only accept the node g with probability $A/(cnA(g))$, where $A(g)$ is the Voronoi cell area of node g, A is the total area of the domain from which g' is selected, n is the number of sensors, and c is a sufficiently large universal constant to make sure $A/(cnA(g)) \leq 1$. In the case of a sample g being rejected, another random location is selected and so on, until a sample node is accepted. This procedure will produce a sensor node uniformly randomly chosen from the network after about c trials. With the presence of holes, the smallest area of a Voronoi cell might be far smaller than A/n. Thus c needs to be large, leading to the waste of communication messages. To ensure a data-centric query eventually finds the data, we assume a source of randomness common to all nodes. Thus, if a node g rejects a data, the data-centric query for this piece of data will be re-directed to the same node and eventually either finds the data or claims failure (when the data is not rejected and the current node is not holding the data). With rejection sampling the storage load can be made uniform, but the rejections can increase the network traffic and energy consumption.

With the reflections and the covering space, the holes are filled by transformed copies of the network. If we use k levels of reflections for a m-hole circular domain, each node has $m^{k-1} + 1$ images (including itself) in the covering space. We calculate the Voronoi diagram of the sensor nodes and their images in the covering space. The chance that a random location is rounded to a sensor node is proportional to the total sum of the area of the Voronoi cell of p and all its images. Since the holes are now covered up, the area of the holes are then distributed to the entire network. All nodes are then selected with similar probability. Thus the maximum storage load of any node is substantially reduced. When rejection sampling is used, the maximum traffic load caused by delivering data to the hashed locations is reduced dramatically, as not only the number of rejections is smaller but also the traffic of the greedy routes are also spread more evenly in the network. For the improvement of traffic load balancing for greedy routing with the covering space, please see Section 4.3.

4.2 Double Rulings

We also apply double rulings, or quorum-based data storage and retrieval scheme, with the covering space. In a double rulings scheme, the producer stores data or data pointers along a storage curve and the consumer (the user who would retrieve the data) routes the request along a retrieval curve. As long as any storage curve intersects with any retrieval curve, it is guaranteed for successful discovery of data. In fact, any retrieval curve can discover *all* the data stored in the network making it more efficient to retrieval data. These curves are often designed as some nice geometric curves. When we route data or request packets, we use greedy routing to select the next hope as the one near the geometric curve and making progress along the curve (e.g., the projection onto the geometric curve is further away) [21]. This leads to routing paths that approximate the geometric double ruling curves. For example, when a network has a rectangular shape, one can use the horizontal lines as storage curves and vertical lines as retrieval curves [19, 26, 28], denoted as rectilinear double rulings. The data packets are routed to the neighbor furthest in the vertical directions, and the retrieval packets are routed to the neighbor furthest in the horizontal directions.

Existing double rulings schemes all assume some nice regular shape (rectangular or circular). When there are holes, many geometric curves may hit a hole. Thus the greedy routing paths approximating the double ruling curves may either get stuck at a local minimum, or, when advanced hole bypassing techniques [6, 16] are used, cause high traffic load on the hole boundaries. Another problem with an irregular shape is that it is not easy to design the geometric double ruling curves that guarantee intersection without using perimeter mode face routing. One can easily construct examples such that rectilinear double rulings, or many other elegant geometric curves fail to intersect inside the sensor domain.

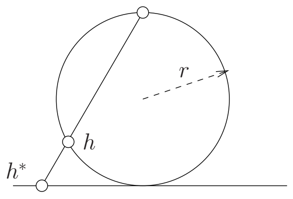


Figure 10. Stereographic projection.

With the covering space, the network is turned into a circular disk D with radius R. Designing geometric double ruling curves is trivial. For example, one can use co-centric circles and rays emitting from an origin as storage and retrieval curves respectively. In our simulation, we use spherical double rulings [25]. We map the covering space to the bottom hemisphere tangent to the center of D with the stereographic projection [3]. We put a sphere with radius $r < R/2$ 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^* in the plane is mapped to the intersection of the line through h^* and the north pole with the sphere. See Figure 10 for a cross intersection. We choose $r < R/2$. Thus the bottom hemisphere is covered by image of the disk into which the network is embedded. For a data item x, we use a hash function f to select a random hash location g^* in the plane and store the data along the storage curves, defined as the great circle through the producer and g. The use of hashing is to allow multiple data items of the same type to be possibly stored and aggregated at the hashed node. The retrieval curve is any great circle through the consumer. It can be shown that any storage curve and retrieval curve have a common intersection in the bottom hemisphere. In the implementation, we find the routing path traversing through the triangles that intersect with a storage (or retrieval) curve. With a sufficiently high k (the

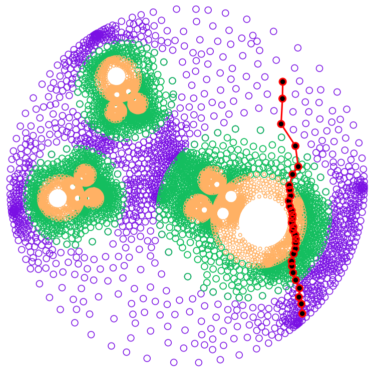


Figure 11. Path in covering space.

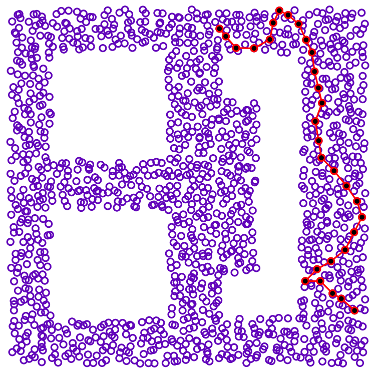


Figure 12. Path in real network.

number of levels of reflections), the holes are mostly filled up with only tiny holes left in the domain. Thus the chance that a double ruling curve hits a tiny hole is very small. For a retrieval curve to discover the data, we only need the intersection of the retrieval curve and the storage curve to be not in a tiny hole. The chance for this to happen is small as well.

4.3 Load Balanced Greedy Routing

Greedy routing selects the next hop as the neighbor whose distance to the destination is minimum, with the distance defined in some coordinate system. It is an extremely simple and completely local algorithm but may not always work if all the neighbors have greater distances to the destination. In our previous work [24], we embed a sensor network in the plane such that all the holes are circular. Thus greedy routing can not get stuck at any hole boundaries. This leads to guaranteed delivery of greedy routing in the generated virtual coordinates. Nevertheless, greedy routes that hit a node on the boundary of a hole C will lead to a path that follows the hole boundary until the tangent point of the line through the destination and C (i.e, when the packet is able to ‘see’ the destination). This effect causes higher traffic load on the hole boundary.

By using the covering space, we can mitigate the imbalance of traffic load caused by the greedy routing method. When a packet reaches a node on the hole boundary, greedy routing directs it to enter another copy (Figure 11), effectively reflect on the hole boundary and take a detour to get around the hole in the real network (Figure 12). Therefore, not all the packets that arrive at a hole boundary will necessarily route along the hole. The detours they take are spread out in the network, reducing the traffic pressure on the hole boundaries.

The routes, reflected on the holes, are possibly longer than before. But this is in some sense necessary in order to improve load balancing. Minimizing the path length and minimizing the maximum traffic load are two contradicting objectives that cannot be achieved at the same time [11]. There is also an interesting trade-off as longer paths increase the total message cost and the average traffic load. In our simulations we demonstrate that a small number of reflections suffice to strike a good balance of path stretch and load balancing.

5. SIMULATIONS

We carried out extensive simulation tests on several different networks to verify the utility of this method. The data presented in this section are based on the network represented in figure 8. This network has about 1400 nodes in a perturbed grid distribution, spread over a 200×200 region, and a maximum communication radius of 12 units. The graph is a quasi-unit disk graph of inner radius $12/\sqrt{2}$.

The following are the important observations we obtained from this set of simulations:

1. The reflection based covering space reduces the maximum storage load at any sensor to almost half, compared with the original embedding, if we uniformly sample geographical locations and then round to the closest sensors.

2. Greedy routing on the covering space has better load distribution than GPSR on the original network.

3. Using the reflections, double ruling schemes can be extended to networks of non-trivial topology. The storage cost is higher but the retrieval cost is much lower.

5.1 In-network Storage and Sampling

As described in the previous section, when using a GHT type storage scheme, the storage load at any node is proportional to the area of its Voronoi cell. The nodes at the boundary of a hole tend to get higher load because their voronoi cells together cover the area of the entire hole. We compared the maximum Voronoi areas of the original embedding with those of covering spaces obtained by one or more reflections. Reflections create multiple images of points for each node, the load on the node is taken to be the sum of the Voronoi cell area of all images.

The results are shown in Figure 13 and Table 1. The total load is normalized to be 1. It is shown that the covering space reduces the max load to almost half that of the original embedding. This is achieved with only 2 or 3 reflections, after which the load does not change too much. This can be understood as follows. The uneven loads are caused by big Voronoi cells, which can be created by the presence of holes, and/or by distortions introduced by the conformal map. After a few reflections the area of the holes are shared by nodes of the entire network and do not contribute large areas to any node in particular. Thus the larger cells created by the distortion of the conformal map are the major reason for the uneven load. This does not get smaller with more reflections.

We carried out rejection sampling on these embeddings, and found that the covering spaces created by 2 or more reflections require about 15% fewer trials on average. Further, when the communication costs are considered, the communication loads are much better balanced in the covering space scheme. This is essentially because greedy routing in generally is better load balanced on the covering space than GPSR on the original network. We describe the routing results in the next subsection.

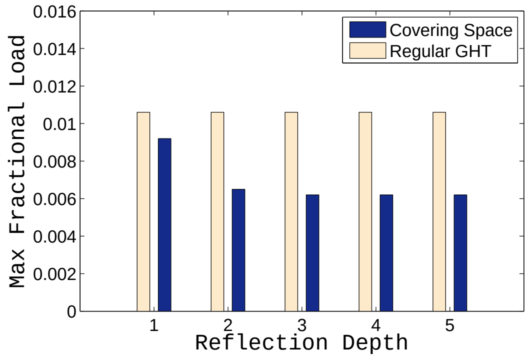


Figure 13. The largest fraction of the storage load at any sensor.

Table 1. Maximum storage load for GHT.

Scheme	1-ref	2-ref	3-ref	4-ref	5-ref
Covering Space	0.0092	0.0065	0.0062	0.0062	0.0062
Regular GHT	0.0106	0.0106	0.0106	0.0106	0.0106

^{+}n-ref - n depth reflection

5.2 Load Balanced Routing

We selected 2000 random source-destination pairs and computed routes. On the original network we used GPSR. On the covering spaces we used simple greedy routing, since greedy routing guarantees delivery in these networks. The number of reflections only shows the maximum number of reflections we permit. The covering space is not pre-computed to that depth, reflections are computed locally as a route progresses and the relevant data is attached to the message.

The results are shown in Table 2. Clearly, 2-3 reflections give the best results. Beyond this point, the route lengths tend to increase as some paths go through several reflections. However, the load balancing is always good, since the covering space method does not hug the boundary when going past a hole. The path bounces off the boundary and spreads the load more evenly (see the plot in Figure 14).

Table 2. Traffic load and path length for greedy routing.

Scheme	Avg. load	Max load	Avg. length	Max length
GPSR	33.6840	620.0	24.1915	92
1-ref	24.0682	319.0	17.571	42
2-ref	35.4960	190.0	25.439	117
3-ref	39.1742	241.0	27.9715	159
4-ref	43.9143	199.0	31.235	196
5-ref	46.3129	216.0	32.8865	228

^{+}n-ref - n depth reflection

Node lifetime experiments. We carried out experiments on how the load balancing properties affect the longevity of nodes. Each node is assumed to have the energy to transmit 200 messages, after which it is considered dead. We count how many nodes die in the process of delivering 4000 messages. As nodes die, the network loses the property of circular holes and guaranteed delivery. We count the number of messages that are delivered successfully under these conditions. GPSR guarantees delivery, but with dying nodes, the network eventually gets partitioned and then some messages fail. The results are shown in Table 3.

Double rulings. Double rulings is a general method that extends GHT. The intuition being that storing data on a path makes it easier for consumers to find that data. Existing double ruling schemes

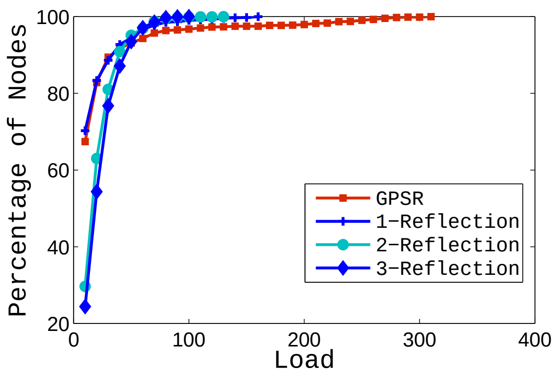


Figure 14. Cumulative distribution of load. Showing that the covering space method has fewer nodes with high load, and also fewer nodes with very low load. GPSR on the other hand has 5% to 10% nodes with substantially high load.

Table 3. Death and message delivery rates for 4000 routing attempts.

Scheme	GPSR	0-ref	1-ref	2-ref	3-ref	4-ref
Deaths	286	33	29	50	122	154
Deliveries	3164	3361	3790	3849	3886	3842

^+n-ref - n depth reflection

such as [25] design the storage and retrieval paths with simple networks in mind. The covering space, by *almost* eliminating the holes in the network can be expected to make double ruling applicable to more general networks. In particular, as the holes get smaller in size, it can be expected that fewer of the storage and retrieval paths hit them. We carried out experiments to test this and other properties on covering spaces of different depths. The radius of the sphere was taken to be one-third the radius of the embedded network in virtual coordinates. The results below are for 2000 random producer, consumer and hash location triples.

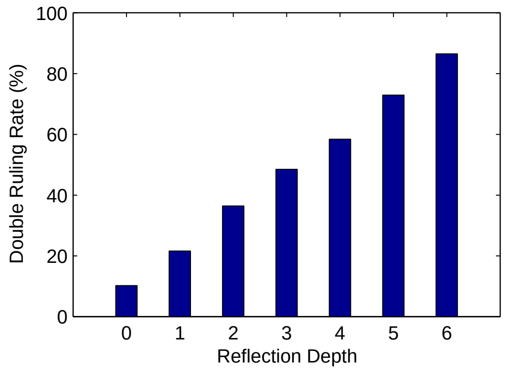


Figure 15. Percentage of successful double ruling retrievals for different depths of covering space.

Figure 15 shows the success rate of double ruling with increasing depth of covering space. For no reflections, the percentage of successes are very small, about 10%, but as holes get smaller, success rate climbs to 86% for 5 reflections. In the following, We ugmented the process at the final level by perimeter mode traversal of boundaries to obtain full success rate.

We carried out path length and load measurements. The results are shown in figures 16 and 17 respectively. The storage costs can be seen to be relatively high, as producers may sometime select a curve that goes through many reflections. In comparison, consumer retrieval costs are lower. This is because the consumer path stops as soon as it intersects a producer path. While this is also true for double ruling in the original embedding, the covering space introduces additional properties. there are many copies of a node x in the covering space. Imagine that the storage path passes through a copy x'. Now it is possible that the retrieval path hits a different copy x'' before it intersects the storage curve in the coring space. In such a case, the consumer has hit a storage node in the real network, and can stop searching. This possibility reduces retrieval costs even further. In a sense, the higher storage costs are compensated by a smaller retrieval cost.

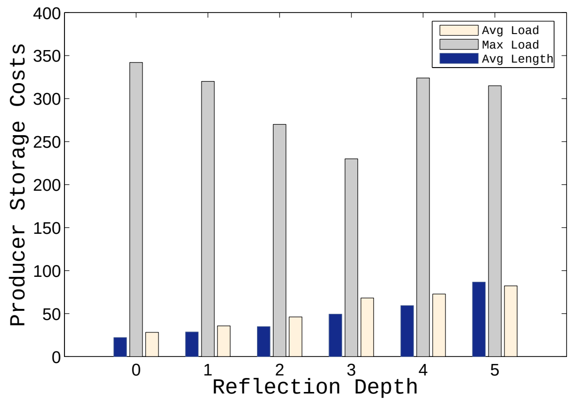


Figure 16. Producer storage costs.

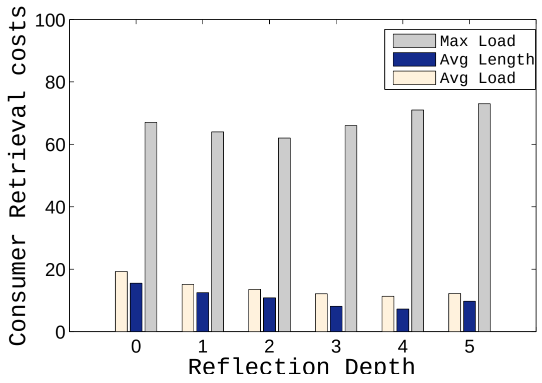


Figure 17. Consumer retrieval costs.

5.3 Network Dynamics

Finally, we tested some effects of network dynamics. In a general sensor network nodes may fail, and occasionally some nodes added. These cause changes in the triangulation. For example, failure of a node creates a ‘hole’ in the triangulation. Many of these changes can be handled by simple adjustments. Failure of a node in a dense network can be handled by adding edges between its neighbors to fill up the hole. In certain cases, say after several failures, such simple local adjustments may no longer suffice and then the embedding needs to be recomputed. However, instead of computing the embedding from scratch, we can use the existing configuration as a starting point to speed up the process. We found that for small changes to the triangulation, the re-convergence is quite fast.

Table 4. Re-convergence Time.

Error bound	$1e-1$	$1e-2$	$1e-3$	$1e-4$	$1e-5$
# Iterations	129	274	697	1392	2221

We made some small changes to the triangulation, such as edge swaps, and measured the time taken for the Ricci flow to converge to some desired error. The results are shown in Table 4.

6. CONCLUSIONS

The network metric and embedding are crucial to sensor network operations. In this paper we presented ideas of using Möbius transforms and Ricci Flow algorithms to regulate a sensor network. All networks can be made to be circular with the interior holes filled up. The created covering space embedding is universally useful to even out sensor storage and traffic load for in-network data storage schemes. We plan to investigate further applications of the embedding in sensor network and sensor data management.

Acknowledgement: Jie Gao and Rik Sarkar acknowledge the support from NSF CAREER Award CNS-0643687. Wei Zeng and David Gu were partially supported by ONR N000140910228, NSF DMS-0626223, and NSF CCF-0448399.

7. REFERENCES

[1] Greenorbs project. http://greenorbs.org/.

[2] Vigilnet project. http://www.cs.virginia.edu/wsn/vigilnet/index.html.

[3] H. S. M. Coxeter. *Introduction to Geometry*. John Wiley & Sons, New York, 2nd edition, 1969.

[4] T. K. Delillo, A. R. Elcrat, and J. A. Pfaltzgraff. Schwarz-christoffel mapping of multiply connected domains. *Journal d'Analyse Mathématique*, 94(1):17–47, 2004.

[5] D.Mumford, C.Series, and D.Wright. *Indra's Pearls*. Cambridge University Press, 2002.

[6] 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.

[7] S. Funke and N. Milosavljević. Network sketching or: "how much geometry hides in connectivity? - part II". In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 958–967, 2007.

[8] D. Ganesan, D. Estrin, and J. Heidemann. DIMENSIONS: Why do we need a new data handling architecture for sensor networks. In *Proc. ACM SIGCOMM Workshop on Hot Topics in Networks*, pages 143–148, 2002.

[9] J. Gao, L. Guibas, J. Hershberger, and L. Zhang. Fractionally cascaded information in a sensor network. In *Proc. of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN'04)*, pages 311–319, April 2004.

[10] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric spanners for routing in mobile networks. *IEEE Journal on Selected Areas in Communications Special issue on Wireless Ad Hoc Networks*, 23(1):174–185, 2005.

[11] J. Gao and L. Zhang. Tradeoffs between stretch factor and load balancing ratio in routing on growth restricted graphs. In *Proc. of the 23rd ACM Symposium on Principles of Distributed Computing (PODC'04)*, pages 189–196, July 2004.

[12] Y.-J. K. R. Govindan, B. Karp, and S. Shenker. Lazy cross-link removal for geographic routing. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 112–124, 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, Anchorage, Alaska, May 2003.

[14] 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.

[15] M. Jin, J. Kim, F. Luo, and X. Gu. Discrete surface ricci flow. *IEEE Transaction on Visualization and computer Graphics*, 14(5):1030–1043, 2008.

[16] 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.

[17] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic routing made practical. In *Proceedings of the Second USENIX/ACM Symposium on Networked System Design and Implementation (NSDI 2005)*, May 2005.

[18] 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, 2003.

[19] 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, 2004.

[20] L. Mo, Y. He, Y. Liu, J. Zhao, S. Tang, X. Li, and G. Dai. Canopy closure estimates with greenorbs: Sustainable sensing in the forest. In *ACM SenSys 2009*, November 2009.

[21] B. Nath and D. Niculescu. Routing on a curve. *SIGCOMM Comput. Commun. Rev.*, 33(1):155–160, 2003.

[22] P.Henrici. *Applied and Computational Complex Analysis*, volume 3. Wiley, New York, 1986.

[23] 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.

[24] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. D. Gu. Greedy routing with guaranteed delivery using ricci flows. In *Proc. of the 8th International Symposium on Information Processing in Sensor Networks (IPSN'09)*, April 2009.

[25] R. Sarkar, X. Zhu, and J. Gao. Double rulings for information brokerage in sensor networks. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 286–297, September 2006.

[26] 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.

[27] J. von Neumann. Various techniques used in connection with random digits. *U.S. National Bureau of Standards Applied Mathematics Series*, 12:36–38, 1951.

[28] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data

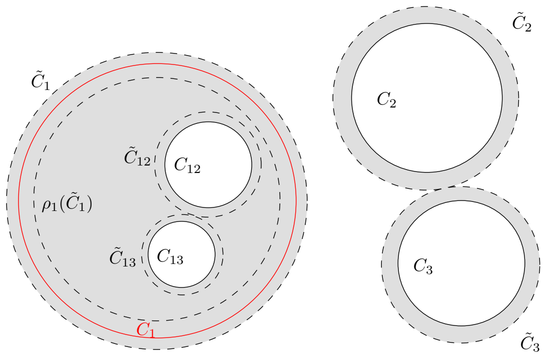


Figure 18. Area Shrinkage.

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, 2002.

[29] F. Zhang, A. Jiang, and J. Chen. Robust planarization of unlocalized wireless sensor networks. In *Proc. of INFOCOM 2008*, pages 798–806, 2008.

APPENDIX

A. PROOF OF LEMMA 2.4

PROOF. Let μ denote μ_{jk} and $\tilde{\mu}$ denote $\tilde{\mu}_{jk}$. The conformal module can be calculated directly for circular domain as

$$\mu = \frac{d_{jk}^2 - \gamma_j^2 - \gamma_k^2 - \sqrt{[(d_{jk}-\gamma_j)^2 - \gamma_k^2][(d_{jk}+\gamma_j)^2 - \gamma_k^2]}}{2\gamma_j\gamma_k}.$$

Then

$$\alpha = \frac{1}{2}(\mu + \frac{1}{\mu}) = \frac{d^2 - (\gamma_j^2 + \gamma_k^2)}{2\gamma_j\gamma_k}$$

$$(\tilde{\mu}d)^2 = (\gamma_j + \gamma_k)^2,$$

It follows

$$\alpha\tilde{\mu}^2 - 1 = (1 - \tilde{\mu}^2)\frac{\gamma_j^2 + \gamma_k^2}{2\gamma_j\gamma_k} \geq 1 - \tilde{\mu}^2.$$

Then

$$\frac{1}{\tilde{\mu}^2} \leq \frac{\alpha + 1}{2} = [\frac{1}{2}(\sqrt{\mu} + \frac{1}{\sqrt{\mu}})]^2.$$

Because $\mu < 1$,

$$\sqrt{\mu} < \frac{2\sqrt{\mu}}{\mu + 1} \leq \tilde{\mu} \leq \Delta.$$

□

B. PROOF OF THEOREM 2.5

The proof depends on the following lemma. As shown in Figure 4, Ω is a bounded double connected domain on the complex plane $\mathbb{C}$, with exterior boundary Γ_0 and interior boundary Γ_1, μ_{01} is the conformal modulus.

Lemma B.1. *Suppose $S(\Gamma_k)$ is the area bounded by Γ_k, $k = 0, 1$, then*

$$S(\Gamma_1) \leq \mu_{01}^2 S(\Gamma_0).$$

Detailed proof can be found in [22], Lemma 17.7c(a), P.503. The proof for theorem 2.5 is as follows:

PROOF. As shown in Figure 18. Ω has three boundary circles C_1, C_2, C_3. Magnify each circle by factor $\frac{1}{\Delta}$, we get (dashed) circles $\tilde{C}_1, \tilde{C}_2, \tilde{C}_3$. By definition of separation modulus, $\tilde{C}_1, \tilde{C}_2, \tilde{C}_3$ may touch, but have no overlaps. Then $\tilde{C}_2, \tilde{C}_3$ are in the exterior of $\tilde{C}_1$.

Reflect $\tilde{C}_j$ through circle C_1 (the red circle). Denote

$$\tilde{C}_{ij} := \rho_i(\tilde{C}_j), i \neq j, 1 \leq i, j \leq 3.$$

Then $\tilde{C}_{12}$ and $\tilde{C}_{13}$ are contained in $\rho_1(\tilde{C}_1)$,

$$S(\tilde{C}_{12}) + S(\tilde{C}_{13}) < S(\rho_1(\tilde{C}_1)).$$

The annulus bounded by C_1 and $\tilde{C}_1$ has conformal modulus Δ. After reflection, the image is the annulus bounded by C_1 and $\rho_1(\tilde{C}_1)$. By Lemma B.1,

$$S(\rho_1(\tilde{C}_1)) \leq \Delta^2 S(C_1) \leq \Delta^4 S(\tilde{C}_1).$$

Similarly

$$S(\tilde{C}_{23}) + S(\tilde{C}_{21}) < \Delta^4 S(\tilde{C}_2),$$

$$S(\tilde{C}_{32}) + S(\tilde{C}_{31}) < \Delta^4 S(\tilde{C}_3).$$

By induction, we can prove Equation 3. □