Exploration of Path Space using Sensor Network Geometry

Ruirui Jiang, Xiaomeng Ban
Computer Science
Stony Brook University
{jiang,xban}@cs.sunysb.edu

Mayank Goswami
Applied Math & Statistics
Stony Brook University
mgoswami@ams.sunysb.edu

Wei Zeng, Jie Gao, Xianfeng Gu
Computer Science
Stony Brook University
{zengwei,jgao,gu}@cs.sunysb.edu

ABSTRACT

In a sensor network there are many paths between a source and a destination. An efficient method to explore and navigate in the 'path space' can help many important routing primitives, in particular, *multipath routing* and *resilient routing* (when nodes or links can fail unexpectedly) as considered in this paper. Both problems are challenging for a general graph setting, especially if each node cannot afford to have the global knowledge. In this paper we use a geometric approach to allow efficient exploration of the path space with very little overhead. We are motivated by the recent development on regulating a sensor network geometry using conformal mapping [44, 45], in which any sensor network can be embedded to be circular (and any possible hole is made circular as well) and greedy routing guarantees delivery. In this paper we explore the freedom of a Möbius transformation inherent to this conformal mapping. By applying a Möbius transformation we can get an alternative embedding with the same property such that greedy routing generates a different path. We present distributed algorithms using local information and limited global information (the positions and sizes of the holes) to generate disjoint multi-paths for a given source destination pair or switch to a different path 'on the fly' when transmission failure is encountered. The overhead of applying a Möbius transformation simply boils down to four parameters that could be carried by a packet or determined at need at the source. Demonstrated by simulation results, this method compares favorably in terms of performance and cost metrics with centralized solutions of using flow algorithms or random walk based decentralized solutions in generating alternative paths.

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

IPSN'11, April 12–14, 2011, Chicago, Illinois.

Keywords

Multipath Routing, Resilient Routing, Möbius Transformation, Ricci Flow, Greedy Routing, Sensor Networks

1. INTRODUCTION

Scalable routing on a sensor network has been an active research topic for the past ten years. The major challenge comes from the fundamental resource limitation of sensor nodes, in terms of storage size and communication bandwidth. The solution that requires a node to acquire the entire network topology does not scale well. In the past few years there have been a number of innovative proposals on scalable routing schemes where each node only keeps local information and a routing path can be discovered by iteratively applying greedy routing decisions. Such work has mainly focused on issues such as guaranteed delivery and low path stretch, and has been relatively successful in that regard.

In this paper we move on and focus on more advanced communication primitives, in particular, routing schemes that exploit the existence of multiple paths in a sensor network. Between a source and a destination there can be multiple different paths. A single path from source to destination may give limited throughput due to bandwidth constraints, hop length, wireless interference or other transmission failures. If there is a lot of data to be delivered, it is natural to consider using multiple disjoint paths[1] such that different data segments can be simultaneously delivered to the destination. With multipath routing one obtains higher throughput and lower delay. Such multipath routing can also be used to enhance data security. For example, sensor data can be encoded such that different codewords are sent along different paths. Therefore a single compromised node stays on at most one path and with its captured data segments it is unlikely to reconstruct the original data.

Exploring the space of routing paths between two nodes is also helpful for fast recovery from link or node failures. In a large wireless network, there are network changes of different scales. At the node level, wireless links have high link quality variation. Nodes may fail. Interference with other nodes could also be unpredictable, e.g., as in the hidden terminal problem. At a large scale, communication links in a region can be temporarily disabled by jamming attacks, either imposed by malicious parties [53], or as a result of co-located multiple benign wireless networks interfering with each other. For example, experiments have shown that 802.15.4 sensor network interferes with existing WiFi network resulting in 54% packet loss [31]. In case of a transmission failure, it would be good

[1]In this paper we focus on node disjoint paths as wireless communication is by nature broadcast. Two paths that share the same intermediate nodes may experience inter-path interference where the common node is the bottleneck.

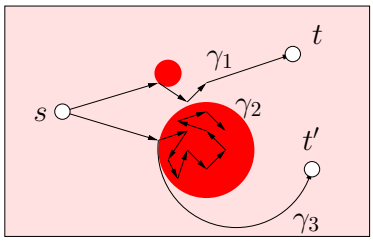

Figure 1. Consider some part of the network experiencing heavy inference (or jamming attacks), shown as the dark colored circles. Links inside these 'failure' regions have much higher loss rate. A route that hits a small failure region might be able to get around by performing some random walks in the neighborhood, as in the case of path γ_1. A route that hits a large failure region has difficulty recovering from it – as simple random walk is likely to wander around for a long time, as shown by the path γ_2. In this case a path that makes big de-tours would perform much better, as shown by the path γ_3.

to quickly discover an alternative path to the destination. A single isolated link failure can possibly be bypassed by local random detouring attempts. A large scale node or link failure, in particular one with strong spatial correlations and a geometric pattern, would need some non-trivial exploration of the path space – by making possibly a big de-tour from the planned path. See Figure 1 for an example. To allow such robustness and quick response to network conditions, routing schemes that find a single path are not enough and it is important to understand the 'space' of paths and efficiently navigate within this space.

Finding multiple routings in a network is a challenging problem for a large scale network, in particular if a node does not have the entire topology. On the theoretical side, one can run a flow algorithm to find a maximum number of node disjoint paths between source and destination. But the flow algorithm requires centralized knowledge and also has a high computational cost of $O(n^3)$ if the network size is n. Even if one can afford to pre-compute multiple routing paths, storing these paths at the sensor nodes will be, storage-wise, too overwhelming, which makes the centralized algorithms scale poorly with the size of the network. In literatures on mobile ad hoc networks, there have been a number of heuristic proposals to find multiple paths but the theoretical understanding of these schemes rarely exists. If one wants to use multiple paths to recover from en-route node or link failure, there is not much understanding (globally) on where the second path is going to be.

Our Approach. In this paper we approach the problem from a geometric angle. Since nodes are typically densely deployed in a geometric domain, the network topology is not like a general graph. We explore different embeddings of the network such that by controlling a *constant* number of parameters we can quickly switch between different network embeddings such that greedy routing in such embeddings generates different routing paths. Thus one can easily come up with multiple node disjoint paths, or even switch to an alternative path in the middle of a routing process, by spontaneously changing to a different embedding.

We are motivated by our recent development of conformal mapping of a sensor network [44, 45]. We first compute an embedding of a sensor network such that all the holes are deformed to be circular. We name this embedding to be a *circular domain*. On a circular domain, greedy routing that always delivers the message to the node closer to the destination using the new coordinates guarantees message delivery. However, the embedding as a circular domain is not unique, and all such embeddings differ by Möbius transformations, which maps a complex plane to itself and can be represented by

$$f(z) = \frac{az+b}{cz+d},$$

where z is a complex variable and a, b, c, d are four complex numbers satisfying $ad-bc = 1$. A Möbius transformation always maps circles to circles. Thus applying a Möbius transformation on a circular domain essentially 're-arranges' the positions and the sizes of the circular holes and the new embedding remains to be a circular domain. Therefore there are actually *infinitely* many circular domains on each of which greedy routing guarantees delivery. The previous work as in [44, 45] only considered one such circular domain by fixing one hole to be at the center of the network. In this paper we investigate all possible circular domain embeddings and the applications to multipath routing in a sensor network.

The main difficulty for efficient and scalable routing in a sensor network is due to lack of the global knowledge. Embedding the network as a circular domain makes that difficulty go away in some sense. Our routing scheme avoids the requirement for the global network topology, while the geometric information we need — the locations and shapes of the holes and the boundary – are typically of a constant size and usually remain stable. With a circular domain one can predict where the path is (subject to the assumption that the sensor nodes are sufficient dense so that the continuous path is a good approximation of the discrete path by greedy routing) and by applying a Möbius transformation we know what a path we will get and how different it is from the previous one. Since a Möbius transformation only uses four parameters, we can attach the current Möbius transformation at the packet such that by applying the Möbius transformation a node can compute its coordinates under the transformation on the spot to generate the greedy path under the new embedding. In case of a link failure on the current greedy path, a node can generate a new Möbius transformation and switch to a different path immediately. The new Möbius transformation is simply attached to the packet.

Using a circular domain representation gives us the following advantages that will be proved in the continuous case and evaluated by simulations for the discrete network setting:

- By using different Möbius transformations one generates multiple paths to the destination that are disjoint except at the source and the destination. We present algorithms for networks with or without holes.
- In case of node failures, we present an algorithm that identifies a different path to the destination. The second path takes a big circular arc type of de-tour that is likely to jump over correlated failure regions.

In the following we first quickly review prior work on multipath routing. We present the theoretical proofs of our method and present simulation results afterwards.

2. RELATED WORK

In this section we quickly review prior work on three relevant topics: multipath routing both in theory and in practice; some of them focus on how to recover from node or link failures; and previous greedy routing schemes.

Multipath routing. Multipath routing has been investigated extensively in computer networking in order to improve routing robustness [4, 9], achieve better load balancing [11, 50, 54], reduce network congestion, reduce end-to-end delay [57] and increase network throughput [18, 52]. Between a pair of source and destination,

multipath routing looks for multiple paths that are sufficiently different from each other such that node or link failures will not destroy all of them. One formulation is to look for k node disjoint or edge disjoint paths, which can be computed by flow algorithm [8]. But this is a centralized algorithm and would require the knowledge of the entire network [17]. Distributed algorithms only exist for special case of $k = 2$. In [37] two colored trees were constructed for routing such that the paths in the two trees are link or node disjoint. Relaxation of the node/edge disjointness of the multiple paths leads to the approach of braided multipath [14] in which the multiple paths are only partially disjoint.

In a mobile ad hoc network, multipath routing has also been developed to enhance the performance of on-demand routing protocols such as AODV [1, 6, 25, 29, 40] or DSR [28, 32, 47] as the network topology undergoes constant changes. Prior work in this direction uses extensive message exchange or flooding to discover alternative paths to bypass a broken link. A major problem of these schemes is that they suffer from high recovery delay from node or link failures, which severely affects the performance of end-to-end QoS measurements in the transport or application layer.

Fast recovery from failures. Recently there has been a number of interesting work that studies the problem of fast recovery from link or node failures, even for a centralized situation. When a link or node fails, the goal is to quickly discover an alternative path with nearly no delay, such that the current traffic is not interrupted. For the intra-domain routing protocols on the Internet, the recent IP fast re-routing (IPFRR) schemes (Loop-free alternate (LFA) [3], O2 [41,42,46], DIV-R [39], MARA [34] and protection routing [24]) aim to ensure fast re-covergence when node failures are detected. In general this family of work would like to find an alternative next hop when the intended next hop is not reachable. Depending on the detailed implementations, the design often suffers from one or more of the following problems: having possible transient loops, the requirement for a lower bound on node degree, computational intractability (e.g., verifying whether a graph has a protection routing or not turns out to be NP-hard [24]).

Our work is motivated by routing with multiple metrics as introduced in the path splicing idea [30], which is proposed for increasing routing reliability on the Internet. Given a weighted graph, one perturbs the weights of the edges and computes a shortest path tree on each node. These multiple shortest path trees are used in combination to generate a routing path in case of in-transit link failures. Traffic in the network can freely switch between different shortest path trees, which results in a large number of routing paths (these paths are the braided multi-paths). The overhead of switching between different trees is done by just changing a few bits in the packet header. This supports fast recovery from link or node failure and ensures low end-to-end delay. However, this is mainly for interdomain routing on the Internet. The computation of the multiple shortest path trees is too costly for a large scale sensor network. For sensor network setting we need to have a low cost method to generate multiple metrics with great flexibility and path diversity.

Greedy routing. Our technique uses greedy routing on different network embeddings, or different metric spaces. Each of the embeddings has the property that all the holes are circular – thus delivering the message towards the neighbor closest to the destination can always get to the destination[2] [44]. In the past few years various greedy routing schemes have been proposed by using a proper embedding of the graphs [2, 12, 16, 23, 26, 35, 38, 44, 45, 55]. Most of these work only focus on finding a single route to the destination [2, 12, 16, 23, 26, 35, 38, 44, 45]. In this paper, we are dealing with a more sophisticated situation – multipath routing in a vulnerable sensor network. Zeng *et al.* [55] considered embedding the sensor network in the hyperbolic covering space such that a network is mapped to multiple copies glued to each other properly; greedy routing to the image of the destination in different copies will lead to homotopy different paths (i.e., these paths get around the network holes in different ways). In some sense this is also a type of multipath routing except that the multiple paths are required to be homotopy different and are not necessarily node disjoint. For the same homotopy type only one path is generated by the greedy algorithm. In our work, even for the same path homotopy type we want to get multiple node disjoint paths. We also handle dynamically appearing 'holes' or link failure regions, while the previous work in [55] assumes all the holes are given and the embedding is computed with respect to that.

For greedy routing using virtual coordinates, typically a location service is available to translate node ID to the virtual coordinates. We assume the same setting for this paper.

[2]This is subject to a small caveat that in certain cases routing on an 'edge' might be needed.

3. ALGORITHMS

Our routing algorithm tries to find various embeddings, or mappings from the original sensor network to a circle domain, in which all holes are of a circular shape and greedy routing can guarantee delivery. Such mappings are conformal (angle preserving) and computed by Ricci flow. Then Möbius transformations could help us to find more embeddings, or controllable multiple metrics.

We first review our previous work [44] of deforming a sensor network shape to make all boundaries circular in section 3.1. Then we describe Möbius transformations applied on such a circular domain. The algorithms for generating multiple node disjoint paths and for loop-free fast recovery from node failure are presented in section 3.3 and section 3.4 respectively.

3.1 Embedding into Circular Domains with Ricci Flow

Conformal Mapping. In the continuous setting for Riemannian surfaces, 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 mapping* (*angle preserving mapping*), if the intersection angle of any two curves is preserved.

A planar domain D of connectivity m is called a *circular domain*, if all its m 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 mappings. Such a mapping is not unique: all such mappings differ by Möbius transformations [10, 36].

To compute the conformal mapping from a surface to a circular domain, one can use Ricci flow as introduced in [19, 44]. In the case of sensor network setting, we will use the discrete version, which represents a domain by a discrete triangulation. We first give some references on how to obtain such a triangulation from a sensor network setting and then move on to the algorithm description of the discrete Ricci flow.

Sensor Network Triangulation. To apply discrete Ricci flow we require a triangulation of a planar domain. There have been quite a number of prior results on extracting a triangulation from the sensor network communication graph. We quickly go through such results. Karp *et al.* [20], Bose *et al.* [5], Gao *et al.* [15], and Li *et al.* [27] proposed distributed, localized methods to extract a planar graph from a unit disk graph. Such a planar graph can

be considered as a triangulation of the planar domain where non-triangular faces are considered as network holes. Sarkar *et al.* [44] extended the algorithm in [15] for the case of a quasi-unit disk graph and also showed that sufficiently big holes are indeed captured as real holes in the triangulation. Funke *et al.* [13] worked on the case when node locations are not available and proposed a triangulation method for a quasi-unit disk graph.

For a general case when no unit disk graph or quasi-unit disk graph assumptions are available, the algorithms in Cross Link Detection Protocol proposed by Kim *et al.* [21, 22] produce a planar graph. Zhang *et al.* [56] proposed to use matching to eliminate crossing edges and produce a planar graph. This algorithm also does not require node locations.

In fact, to run Ricci flow algorithm and find the circular domain embedding, one does not require the triangulation must be a *subgraph* of the communication graph. The triangulation can be a virtual graph of the underlying domain as long as the sensor nodes jointly maintain it. In other words, each wireless node takes in charge of a group of virtual vertices of the triangulation and carries out the computation for these virtual nodes. The position of a wireless node can be made identical to any node it is in charge. This idea of 'virtualization' allows the method to be extended to a general setting when the wireless node is relatively powerful and the distribution is sparse (as a natural consequence since fewer nodes can sufficiently monitor the domain). Therefore the triangulation can be any proper triangulation of the underlying geometric domain, for example by standard meshing techniques such as Delaunay refinement methods [43, 48]. In the following discussion we assume that a proper triangulation is obtained and we focus on the embedding of the network from now on.

Discrete Ricci Flow. In the following we explain Ricci flow in the discrete setting for a triangulation of a domain with m holes. The triangulation is denoted by Σ with vertex set V, edge set E and face set F.

In the discrete setting we define a Riemannian metric by using the edge lengths on Σ:

$$l : E \to \mathbb{R}^+,$$

such that for a triangle face f_{ijk} with vertices v_i, v_j, v_k, the edge lengths satisfy the triangle inequality:

$$l_{ij} + l_{jk} > l_{ki}, \quad \forall i, j, k.$$

The lengths of the edges of the triangulation determine the corner angles of the triangles. For a triangle f_{ijk} with edge lengths $\{l_{ij}, l_{jk}, l_{ki}\}$, and the angles opposite to these edges $\{\theta_k^{ij}, \theta_i^{jk}, \theta_j^{ki}\}$ respectively, we have the following equations by cosine law:

$$l_{ij}^2 = l_{jk}^2 + l_{ki}^2 - 2 l_{jk} l_{ki} \cos \theta_k^{ij}. \tag{1}$$

Now we can define the discrete Gaussian curvature at a vertex v_i as the angle deficit:

$$K_i = \begin{cases} 2\pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, & v_i \text{ is an interior vertex;} \\ \pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, & v_i \text{ is at boundary.} \end{cases} \tag{2}$$

where θ_i^{jk} represents the corner angle at vertex v_i in the triangle f_{ijk}. In other words, the curvature at a vertex v is the difference of 2π or π and the total corner angles at v, for an interior vertex or a vertex on a hole boundary respectively. The curvature is 0 when it is locally flat (for interior vertices) or locally a straight line (for boundary vertices).

Ricci flow uses the circle packing metric in the discrete case, proposed by [49, 51], to approximate the conformal deformation of

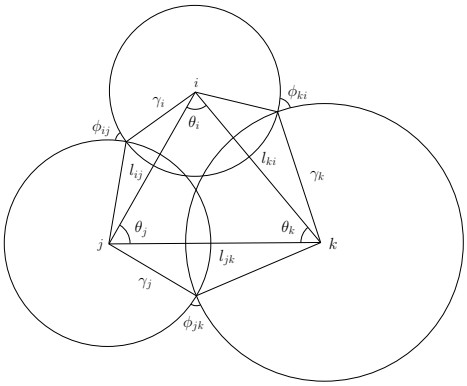

Figure 2. The circle packing metric.

metrics. See Figure 2. Each vertex v_i has a circle with radius γ_i. On each edge e_{ij}, ϕ_{ij} is defined as the intersection angle of the two circles at v_i and v_j. The pair of vertex radii and intersection angles at the edges on a mesh Σ, (Γ, Φ), is called a *circle packing metric* of Σ. Two circle packing metrics (Γ_1, Φ_1) and (Γ_2, Φ_2) on Σ are *conformal equivalent*, if $\Phi_1 \equiv \Phi_2$. Therefore, a conformal deformation of a circle packing metric only modifies the vertex radii γ_i's and maintains the intersection angles ϕ_{ij}'s to be constant.

For a given mesh, the circle packing metric (Γ, Φ) and the edge lengths can be converted to each other by cosine law as below:

$$l_{ij}^2 = \gamma_i^2 + \gamma_j^2 + 2\gamma_i \gamma_j \cos \phi_{ij}. \tag{3}$$

Thus given a circle packing metric, we can calculate the edge lengths of the triangulation Σ and then the embedding in the plane realizing the given curvatures.

Let u_i to be $\log \gamma_i$ for each vertex. The discrete Ricci flow is defined as the following differential equation:

$$\frac{du_i(t)}{dt} = (\bar{K}_i - K_i), \tag{4}$$

where K_i is the current curvature at vertex i and $\bar{K}_i$ is the target curvature at i. Define an energy function

$$f(\mathbf{u}) = \int_{\mathbf{u_0}}^{\mathbf{u}} \sum_{i=1}^{n} (\bar{K}_i - K_i) du_i, \tag{5}$$

as the *Ricci energy*, where $\mathbf{u_0}$ is an arbitrary initial metric. It has been proved by Chow and Luo [7] that the discrete Ricci flow will converge to a unique minimum of the Ricci energy. The convergence rate of the discrete Ricci flow using Equation 4 is shown to be exponentially fast, i.e.,

$$|\bar{K}_i - K_i(t)| < c_1 e^{-c_2 t}, \tag{6}$$

where c_1, c_2 are two positive constants.

The Ricci flow algorithm is naturally an iterative algorithm with all vertices adjusting local metrics and local curvatures. All the radii at the vertices are initialized to be $1/2$. That is, the circles at adjacent vertices of Σ are kept to be tangent to each other. We set the target curvature to be 0 at all interior vertices. That is, the network should be embedded to be flat in the domain. We set the target curvature at a boundary vertex to be $-2\pi/k$, if the boundary of the hole has a total number of k vertices. That is, the boundary circle should be perfectly circular. We apply the Ricci flow algorithm by changing the circle packing metric, u_i, by $\delta(\bar{K}_i - K_i)$, where δ is a constant parameter as the step size. The algorithm stops when the current curvature is within an error bound of ε from the target curvature.

Since the curvature error decreases exponentially fast, the number of steps in the Ricci flow algorithm is in $O(\frac{\log(1/\varepsilon)}{\delta})$, where δ is the step size in the Ricci flow algorithm. The total number of messages is thus in $O(\frac{n\log(1/\varepsilon)}{\delta})$, if the algorithm is running on a network of n vertices.

3.2 Möbius Transformations

Möbius transformations are rational functions defined on the complex plane $\mathbb{C}$. The general form of a Möbius transformation is

$$f(z) = \frac{az+b}{cz+d}.$$

Here $a, b, c, d \in \mathbb{C}$ and satisfy $ad - bc \neq 0$. If $c \neq 0$ we can extend this mapping to the Riemann Sphere (or the extended complex plane, i.e., with a point of infinity) $\widehat{\mathbb{C}} = \mathbb{C} \cup \{\infty\}$ by specifying $f(-d/c) = \infty$ and $f(\infty) = a/c$. In the case when $c = 0$, we specify $f(\infty) = \infty$.

Here are the important properties of Möbius transformations:

1. Möbius transformations are all the bijective holomorphic (differentiable in the complex sense) mappings from $\widehat{\mathbb{C}}$ to itself. This also implies that they are conformal, or angle-preserving.

2. Möbius transformations carry circles and lines (which can be regarded as circles passing through ∞, point of infinity) to circles and lines. Thus, giving a circular domain, any Möbius transformation will map it to another circular domain.

3. To every Möbius transformation one can associate a matrix

$$M_f = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

   Any other matrix which is a (nonzero) scalar multiple of this matrix represents the same Möbius transformation. Composition of two Möbius transformations is equivalent to matrix multiplication, i.e., $M_{f\circ g} = M_f \cdot M_g$.

4. Given distinct $z_1, z_2, z_3 \in \widehat{\mathbb{C}}$ and distinct $w_1, w_2, w_3 \in \widehat{\mathbb{C}}$, there is a *unique* Möbius transformation f satisfying $f(z_i) = w_i$, $i = 1, 2, 3$. In other words, as there are unique circles C_1 and C_2, defined by z_1, z_2, z_3 and w_1, w_2, w_3 respectively, the transformation f maps the circle C_1 to C_2 and is unique. Determining f explicitly is equivalent to finding determinants of four 3×3 matrices.

It should be noted that there is a natural way to identify the real plane $\mathbb{R}^2$ with the complex plane $\mathbb{C}$, so for our purposes we can assume that nodes are in the complex plane.

3.3 Multipath Routing

In this section we describe how to generate multiple paths from a given source node s to a given target node t. We will give different embeddings of the domain in such a way that the route used by greedy routing in one embedding is likely to be different from the one used by greedy routing in any other embedding, and these routes are 'well spaced', a notion we will make precise soon. The algorithm we will present generates paths that are provable to be disjoint in the continuous case. In the discrete setting, we evaluate the performance by simulations.

We first present the algorithm for a network without holes. Then we discuss how to find disjoint paths in a network with holes.

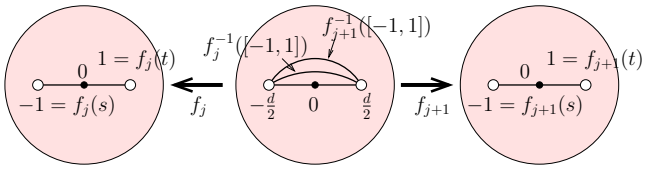

Figure 3. The multiple paths on the domain D (in the middle) are the greedy paths under transformations f_j. The figure shows two transformations f_j and f_{j+1} respectively.

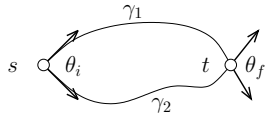

Figure 4. For two curves γ_1 and γ_2 from s to t, the initial directional spread is shown as θ_i and the final directional spread is shown as θ_f.

3.3.1 Network Without Holes

Without loss of generality we can assume that the outer boundary is a circle, and that the coordinates of source s are $(-d/2, 0)$ while those of destination t are $(d/2, 0)$, so that the line segment joining s to t is horizontal and of length d. This can easily be achieved by a rotation and translation. We denote the domain by D.

Consider a continuous domain D we can easily generate many disjoint paths – by essentially applying a different Möbius transformation each time. The greedy path under a Möbius transformation turns out to be a circular arc connecting s and t in the original domain D. See Figure 3. In the discrete case when the domain D is represented by a triangulation, the routing paths are found by using greedy routing in different embeddings, after proper Möbius transformations. We remark that potentially one can design greedy routing to follow any curve, i.e., as in the idea of routing along a curve [33]. But in general routing on a curve does *not* have any guarantee on the delivery. In our case, as we actually perform greedy routing in another circular domain after a proper Möbius transformation, this immediately shows a proof that such a route is guaranteed to reach the destination.

In a continuous domain D, obviously all such circular arcs are disjoint except at source and destination. In the discrete case the greedy paths are guided by the circular arcs but can definitely deviate from them due to discrete node distribution. Since typically sensor networks have upper bounded density, a major constraint on the number of node disjoint paths between s and t is due to the degree at s and t. A reasonable heuristic to minimize overlaps of multiple paths is to design paths that are evenly spread out at s and t. We use this heuristic to design our paths in the discrete setting.

Given two curves γ_1 and γ_2 joining s to t, we can define initial and final *directional spread* between γ_1 and γ_2 to be the angle between the tangent vectors of γ_1 and γ_2 at s and t respectively. We denote these by $d_i(\gamma_1, \gamma_2)$ and $d_f(\gamma_1, \gamma_2)$ respectively. See Figure 4 for an example. In the following we use Möbius transformations to generate circular arcs connecting s and t such that their directional spread at source and destination are as evenly spread as possible.

For a given $k \geq 1$, let $\theta = \frac{\pi}{2k}$ and define $\theta_j = \frac{\pi}{2}(1 - \frac{j-1}{k})$ for $1 \leq j \leq 2k+1$. Also, let $\alpha_j = d/2 \tan\theta_j/2$ for $1 \leq j \leq 2k+1$.

We next define a Möbius transformation $f_j(z)$ for $1 \leq j \leq 2k+1$ by

$$f_j(z) = \frac{zd - id\alpha_j}{z(-2i\alpha_j) + d^2/2}.$$

Theorem 3.1. *f_j has the following properties:*

1. $f_j(s) = -1, \quad \forall 1 \leq j \leq 2k+1.$
2. $f_j(t) = 1, \quad \forall 1 \leq j \leq 2k+1.$
3. *Let $\gamma_j = f_j^{-1}([-1,1])$. Then γ_j is a curve joining s to t. Moreover, it is the arc of the unique circle passing through s and t, such that the tangent vectors at s and t both make an angle of θ_j with the x-axis.*
4. $d_i(\gamma_j, \gamma_{j+1}) = d_f(\gamma_j, \gamma_{j+1}) = \theta, \quad \forall 1 \leq j \leq 2k+1.$

PROOF. (1) and (2) are trivial.

To prove (3) we use the property that Möbius transformations map circles to circles. Note that the point $(0, \alpha_j)$ is represented by the complex number $z = i\alpha_j$. One can check that $f_j(i\alpha_j) = 0$. Since $-1, 0$ and 1 lie on a line, it means that s, $(0, \alpha_j)$ and t lie on a circle. Therefore $f_j^{-1}([-1,1])$ is an arc of the circle passing through these three points. One has to now verify that the tangent to this (unique) circle at s and t makes an angle of θ_j with the horizontal axis.

(4) follows from (3) and the fact that $\theta_{j-1} - \theta_j = \theta$. □

What the above calculations mean is that if we define a new embedding of the domain D by mapping a point $z \in D$ to $f_j(z) \in f_j(D)$, then the source maps to -1 and the target maps to 1. In this new embedding, the shortest path from source to target is simply the straight line from -1 to 1. Following this path for some j is equivalent to following the arc of the unique circle passing through s and t whose tangents at s and t make an angle of θ_j with the horizontal axis. Any two such arcs have an initial and final directional spread of at least $\theta = \frac{\pi}{2k}$. Hence we have generated $2k+1$ node disjoint θ spread paths from s to t. See Figure 3 for an example.

All such paths lie in the circle with the line segment st as its diameter. We can also consider the case $\theta_j = \frac{\pi}{2}(1 + \frac{j-1}{k})$ to get $2k-2$ more node disjoint paths, with an angle spread of θ, getting $4k-1$ in total. For example if $k = 3$, we can get a total of 11 paths, such that the directional difference is at least $\frac{\pi}{6}$.

We remark that the above results hold only for source s and target t pairs for which the circle with line segment $\overline{st}$ as diameter, denoted as C_{st}, is contained inside the domain (i.e., in the interior of the outer circle C). When this is not the case, the paths will 'hit' the outer boundary circle. Such paths will merge as they follow along the outer boundary and are hence not disjoint. However, this is actually a case that will be handled by the algorithm in the following section as the outer boundary is also a topological hole.

The above analysis is done in the continuous setting. We will present the multiple path routing results in a discrete setting by simulations.

3.3.2 Network With Holes

Consider a circular domain D with k holes (including the outer hole, which can be regarded as a circle centered at ∞). In this case, finding disjoint paths is more complicated. This is precisely because if two paths both hit the same hole, they will start to follow the boundary of the hole and converge. This will create a long shared sub-path on that boundary. Therefore we would need to find paths such that either (i) they do not hit the same hole, or, (ii) when two paths hit the same hole, they hit the top and bottom of the circular hole respectively so they follow the upper boundary and lower boundary and do not converge. Take a look at Figure 5. For each hole C_i in the domain, we take three circular arcs through s and t, the one that is tangent to C_i internally (i.e., including C_i); the one that goes through the center of C_i; and the one that is tangent to

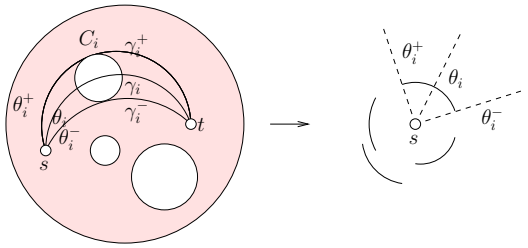

Figure 5. For a pair of source and destination, each hole C_i will produce two intervals θ_i^+ and θ_i^- such that any two paths falling in the same interval will hit the hole and share some segments of the boundary. Thus any set of disjoint paths can only select one path inside each interval.

C_i externally (i.e., excluding C_i). These three paths are denoted as γ_i^+, γ_i, γ_i^-, respectively. Now, any two circular arcs through s and t that fall in between γ_i and γ_i^+ (or γ_i and γ_i^-) will definitely merge on the boundary of C_i. Thus we can only allow one path selected within each angular range bounded by $[\gamma_i^+, \gamma_i]$, and $[\gamma_i, \gamma_i^-]$. In the following we present an algorithm that finds a maximum number of hole touching paths satisfying the above constraints.

Given a source s and a destination t, we can first assume without loss of generality that the x coordinate of t is larger than that of s and that there does not exist a circle which passes through s and t which is tangent to more than one of the holes. For a hole C_i with center c_i, there are three arcs of relevance: γ_i, γ_i^+ and γ_i^-, as defined earlier. Let θ_i,θ_i^+ and θ_i^- denote the angles that the initial tangent vectors to γ_i, γ_i^+ and γ_i^- make with the horizontal axis. Note that the convention is that all of them are contained in the interval $[-\pi, \pi]$.

Now we define the following angular intervals $\{T_i\}_{i=1}^{2m}$, two for each of the m holes: $T_{2h-1} = [\theta_i^-, \theta_i]$ and $T_{2h} = [\theta_i, \theta_i^+]$ ($1 \leq h \leq m$). For the outer hole, arc γ_i is simply the straight line joining s to t; while arcs γ_i^+ and γ_i^- are contained in circles that pass through s and t and are tangent to the outer boundary at the top and bottom.

Now any two circular arcs joining s to t, both of whose initial directions lie in the same interval T_{2h-1} or T_{2h}, will both traverse either the upper or the lower boundary of the hole C_i, and hence cannot be disjoint. We want to find the maximum number of arcs, all of which pass through s and t, and touch at least one hole.

Since all the T_i's lie in $[-\pi, \pi]$, we can think of them as subsets of the unit circle S^1. We can angularly sort the endpoints, to obtain a sequence $s_1, s_2, ..., s_{4k}$ where each s_i is an endpoint of T_j for some j. If some interval $[s_i, s_{i+1}]$ is not contained in any T_j, we are free to use it as there are no constraints associated to such an interval. Assume that we have collapsed all the $[s_i, s_{i+1}]$ that are not contained in any T_j, and now we are left with a sequence $s_1, s_2, ..., s_l$.

Now we want to find a maximum number of intervals such that any two intervals cannot be part of the same T_i for any i. That is, each interval in the solution can be used to generate one circular arc and all such circular arcs do not intersect with each other except at s or t.

The above problem has an optimal solution using a greedy algorithm that we now describe. Each interval $A_i = [s_i, s_{i+1}]$ now is contained in (covered by) some T_j's. We obtain a solution Q_i as follows. Q_i starts with a seed interval A_i. Move clockwise on the circle until the first interval $[s_{j_1}, s_{j_1+1}]$ which is not covered by any T_j that covers A_i. When this happens, include A_{j_1} in the solution Q_i and proceed greedily until we cannot include any more

intervals to Q_i.

After this process, we have l solutions $Q_1, Q_2, ..., Q_l$. We choose the one with a maximum number of intervals. This solution is optimal.

Theorem 3.2. *The number of intervals in the solution chosen above (i.e., the best amongst the $\{Q_i\}_{i=1}^{l}$) is equal to the number of intervals chosen in the optimal solution.*

PROOF. Pick any interval that the optimal solution chose, say A_{j_1}. Consider what the greedy algorithm performed in Q_{j_1}. Let the next interval chosen by the greedy algorithm be G_{j_2} while the one chosen by the optimal be O_{j_2}. Assume they are different (if they are the same the argument proceeds). If there does not exist j such that G_{j_2} and O_{j_2} are both contained in T_j, then adding the interval G_{j_2} to the optimal solution increases the number of intervals in the optimal, which is a contradiction. So assume T_j contains both G_{j_2} and O_{j_2}. Therefore by the end of T_j we have not done any worse than the optimal, as both have added one interval each. The argument now proceeds in a similar fashion. G_{j_3} and O_{j_3} would both have to be contained in some T_j again, by the end of which we are again no worse than the optimal and so forth. Inductively we can show that our solution is no worse than the optimal and thus must be optimal. □

Using this greedy method we can thus find a maximum number of node disjoint paths in the domain all of which pass through a hole. We can use results of the previous section to generate node disjoint θ spread paths in the intervals $[s_i, s_{i+1}]$ which were not covered by any T_j. Thus putting together, we can find a maximum number of disjoint paths that do touch some hole and depending on the spread we can find disjoint paths that do not touch any hole using previous results.

To summarize, our multipath algorithm will generate k disjoint paths in a network with and without holes by applying different Möbius transformations, with provable results for the continuous case. When the nodes have high density, the greedy paths in the discrete case will better approximate the circular arcs. When the node density drops, the multiple paths may overlap in the middle. We evaluate in the simulation section the dependency of the performance on the network density.

3.4 Recovery From Failure

In this section we describe how to deal with en-route node failures or link failures. Recall that our embedding produces a circular domain that guarantees delivery when links are assumed to be reliable. When a link may fail, greedy routing no longer guarantees delivery. For example, a node may discover that all the neighbors that are closer to the destination are not reachable. In this case we aim to find an alternative path. The freedom of applying Möbius transformations on a circular domain provides great flexibility for this task.

Assume s is the source node that wants to transmit a package to t ($s, t \in D$). Let the degree of s be ν. Furthermore, assume that s has sorted its neighbors in increasing order of their distances from t such that

$$||p_1t|| \leq ... \leq ||p_kt|| \leq ||st|| \leq ||p_{k+1}t|| \leq ... \leq ||p_\nu t||.$$

$||uv||$ is the Euclidean distance between u, v. If the link between s and any of the $\{p_i\}_{i=1}^{k}$ is functional, s routes the message to that neighbor just as in greedy routing. Assume that the only links available to s are those in $\{p_i\}_{i=k+1}^{\nu}$. Then s picks a functional link from this set, say the link to $p = p_{k+1}$. Now the idea is to find a Möbius transformation such that in the new embedding, p is closer to t than s, so that greedy routing would then continue by using p as the next hop. The details are presented below.

As soon as node s finds that greedy routing can no longer continue, it does the following:

1. It finds the coordinates of a neighbor p. Assume that s knows the coordinates of p and the destination t.

2. s finds a Möbius transformation that maps s to -1, p to 0 and t to 1. The explicit formula for f is
$$f(z) = \frac{z(s-t) + p(t-s)}{z(2p-(t+s)) - p(t+s) + 2st}.$$

3. s then sends the package to p along with the information about this Möbius transformation. p calculates new coordinates for all of its neighbors and for t.

4. Greedy routing then continues (since now $f(p)$ is clearly closer to $f(t)$ than $f(s)$), until we get stuck at another node. When this happens, we repeat the entire process, i.e., find another Möbius transformation and compose it with the previous one.

As will be shown later in the simulation section, our failure recovery mechanism is compared with random walk – simply pick a random 'live' link until greedy routing can be performed again. Basically our scheme makes big jumps and chooses a vastly different alternative path while random walk can only make local adjustments. This benefit of using long de-tours is significant for failures that exhibit spatial patterns.

4. SIMULATIONS

In the experiments, we perform greedy routing with Möbius transformations to achieve multipath routing and link failure recovery. Our simulations are performed on unit disk graph topologies potentially with holes inside, and in the following are our key observations:

Multipath routing: By using Ricci flow with different Möbius transformations, we can generate a substantial fraction of node disjoint paths. With reasonable sensor density (average degree around 20), the average number of disjoint paths we find using our algorithm is consistently more than 70% of the input parameter m (the desired number of disjoint paths). We can consistently find two node disjoint paths even in very sparse networks. We also observed that when the network is sparse, the bottleneck for finding node disjoint paths is often near the source and destinations.

Recovering route from link failures: Under a spatial failure model in which the nodes in a geometric failure region have a much higher failure rate, our method of using greedy routing on the virtual coordinates in a circular domain with Möbius transformations as the recovery scheme performs consistently better than all other methods (on virtual or original coordinates, using random walk as the recovery scheme). The advantage of using Möbius transformations rather than random walk as a recovery scheme diminishes when the failure pattern is no longer spatially correlated.

4.1 Multipath Routing

After we generate the sensor network $G = (V, E)$, we randomly choose two vertices s and t from V as the source and the destination. We then calculate the maximum number of node-disjoint

$\kappa(s,t)$	m	Disjoint paths generated	Approximation factor
6	3	2	67.7%
	5	4	
8	3	3	62.5%
	5	4	
	7	5	
11	5	4	72.7%
	9	7	
	11	8	

Table 1. Results of different sources and destinations in a uniform distributed graph with average edge links 20.

Nodes	Average link number	Input m	Average output paths / m
1000	20.00	3	83.3%
		5	79.0%
		7	71.4%
600	12.02	3	76.7%
		5	64.0%
		7	51.4%
400	5.62	3	63.3%
		5	46.0%
		7	35.7%

Table 2. Results of graphs with different sensor densities.

paths between s and t, called the vertex connectivity $\kappa(s,t)$, as a reference for comparison, using the centralized maximum flow algorithm [8]. To test our multipath routing algorithm, we generate m (no larger than $\kappa(s,t)$) paths from the source to the destination, and count how many of them are node-disjoint. We also try different m parameters to further observe the performance of the algorithm.

Figure 6 shows the proposed routing scheme on a sensor network with 1000 vertices and 10006 links. We first apply Ricci flow to embed the network into a circular domain where each node is given a virtual coordinate. We use our multipath routing algorithm to seek $m_1 = 3$, and $m_2 = 5$ paths from s (in yellow) to t (in red) respectively (in this graph $\kappa(s,t) = 6$). Those paths are not necessarily node-disjoint, and all the shared nodes/edges are marked in purple in the figure. We also show the paths on the original network, and a different circular domain obtained by a Möbius transformation as well. In each of the three embeddings, the paths with the same color and number are identical. We can see that in different circular domains, the greedy paths, or the straight lines from s to t are also different, which demonstrates that Möbius transformations together with greedy routing give us flexibility in choosing multiple paths.

More results are shown in Table 1. From the table we can see two facts. First, as a distributed algorithm, the Möbius transformation method gives us a good approximation of the number of node-disjoint paths in a dense graph. Second, the number of disjoint paths we can get is usually smaller than m, or equivalently, some paths we generate share common nodes or edges. This is due to the discrete nature of graphs. From s, we can only send packets to its neighbors, which is a restriction in choosing the first few hops of the transmission; the hops near t suffer from the similar problem. But in the middle segments, the paths generally follow the shape of a circle arc connecting s and t, which is desired. When the network becomes denser, the situation becomes more similar to that of the continuous case.

To further explore the differences between the discrete and continuous settings, we also simulate under different graphs. In graphs with different densities (with uniform sensor distribution), we randomly pick 10 pairs of sources and destinations, give the different inputs m as 3, 5 and 7, and calculate the average numbers of disjoint output paths we get. The results are shown in Table 2. From the table, we observe that the algorithm performs better in a denser graph with more links. This is reasonable since the gap between the discrete and continuous settings is smaller with a denser sensor distribution. Moreover, when the input m is smaller, the algorithm gives better approximation. This is simply because given a smaller input, the arcs span further away with each other (this result does not conflict with Table 1 where $\kappa(s,t)$ is known). We also notice that when m exceeds the average node degree, the percentage drops drastically, where the input – the number of disjoint paths we are trying to find – often exceeds the optimal value.

When the sensor distribution is non-uniform, the bottleneck of the performance lies in the sparse regions, especially when those regions cover the neighborhood of the source or the destination.

Figure 7 shows the result of a network region with holes. Some paths go along the inner boundary, but the heavy load along the boundaries is avoided.

4.2 Routing with link failures

In sensor networks, links are likely to fail, especially in an adversarial environment. Since greedy routing requires that for each step, there exists one link leading to a node closer to the destination, the performance of greedy routing will drop quickly when a link failure happens. What is more, link failures often have a property of spatial locality, which means that a group of nearby links are likely to fail at the same time. Therefore, when greedy routing hits this region, the message may 'get trapped in the mud'. We use the freedom of Möbius transformations to recover from this situation.

Based on the above observations, we adopt the setting of clustered random link failures. We first test a simple setting in which there is a region with arbitrary size and shape. All the links within this region have a link failure rate p, while all the links outside the region or crossing the boundary of the region will not fail. A message following greedy routing in the circular domain may not have guaranteed delivery as the link to the next hop can suddenly fail. Our strategy is to adopt a different Möbius transformation. We compare it with another simple strategy that recovers from failure by performing a random walk. Although random walk is simple, it is time-consuming for the routing path to jump out of a large link failure region, due to its locality and randomness. In our following experiment, we will compare greedy routing with Möbius transformations with other greedy routing techniques, in terms of routing delivery rate and routing path length. The experiment network size is 1000 nodes with a varying number of links. The link failure region is a rectangle lying in the network.

In the experiments, we compared the following methods:

Greedy routing on the original coordinates: Simple greedy routing on the original coordinates, which fails to route to the destination easily due to link failures. We call this method **Greedy** in short.

Greedy routing on the virtual coordinates: Greedily route to the destination using coordinates computed by Ricci Flow. We call this method **Ricci** in short.

Greedy routing on the original coordinates with random walk:

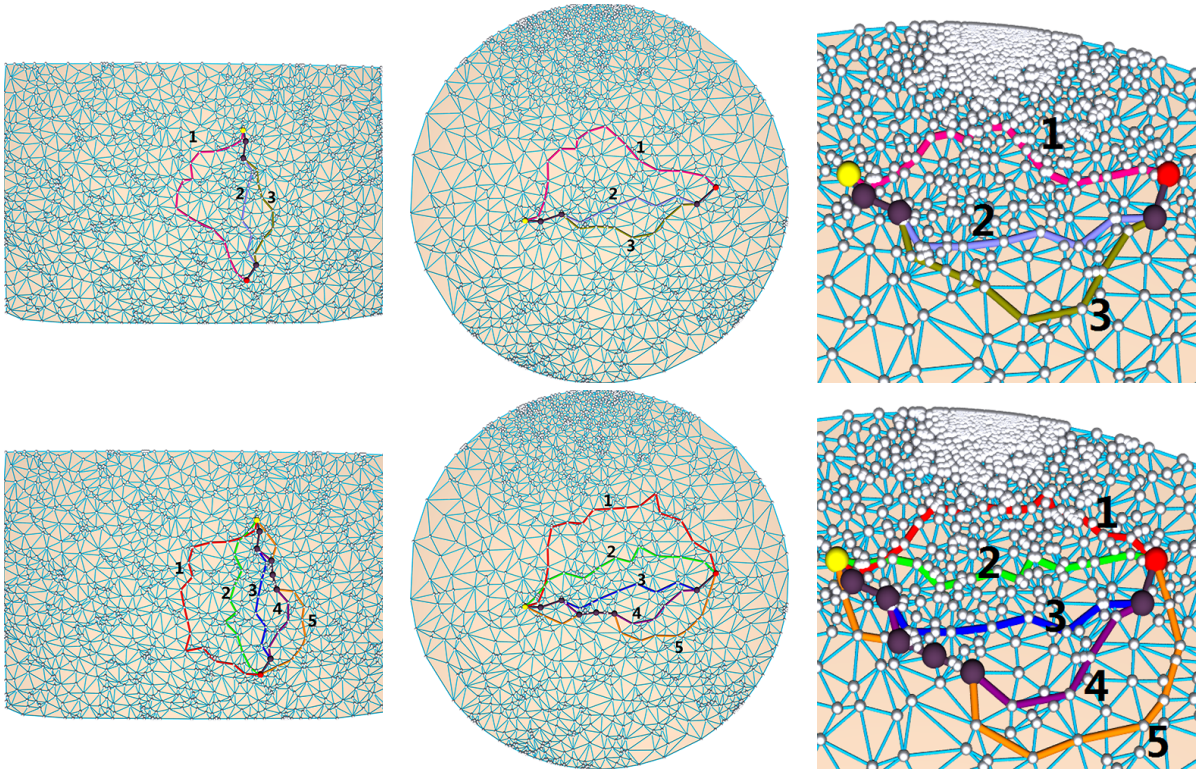

Figure 6. Multipath Routing Algorithm. Left column: original network; middle column: network applying ricci flow; right column: network applying ricci flow and a Möbius transformation (zoomed in). First row: $m = 3$; second row: $m = 5$.

Route based on the original coordinate set and perform random walk to recover from failure. We call this method **GreedyRnd** in short.

Greedy routing on the virtual coordinates with random walk: Greedily route using virtual coordinates and perform random walk to recover from link failures. We call this method **RicciRnd** in short.

Greedy routing with Möbius transformations: Our method performs greedy routing based on the virtual coordinates in a circular domain. If the route gets stuck in the middle due to link failures, it performs a Möbius transformation to get a new path towards the destination. We call this method **Möbius** in short.

Various parameters will affect the success rate of routing. In our experiment, we focus on the average degree of the network, the link failure rate and the TTL (time-to-live) of the packet. If the connectivity of networks becomes better as the average degree increases, routing will be easier for all methods. Obviously a higher link failure rate will make all methods suffer. We also include a TTL with each packet to stop a packet from roaming aimlessly in the network, in particular in the random walk method.

Routing result and analysis. In Figure 8, 9 and 10, we show the message delivery rates by varying the network density, the TTL and the link failure rate respectively. In all settings, we can see our method has a significantly higher delivery rate than the other methods, which shows that by using a new Möbius transformation, we can effectively find an appropriate path which leads the route out of the failure region. In general the performance of different methods, in decreasing order, follows the trend of **Möbius > RicciRnd > Ricci** $\gg$ **GreedyRnd** $\simeq$ **Greedy**. Greedy routing using the original coordinates nearly does not work, no matter whether it is augmented with random walk or not.

p_1	p_2	Möbius	Ricci	RicciRand	Greedy	GreedyRand
0.3	0.0	0.962	0.573	0.779	0.021	0.028
0.6	0.0	0.912	0.402	0.651	0.014	0.016
0.6	0.3	0.738	0.207	0.472	0.003	0.009
0.9	0.0	0.823	0.271	0.511	0.008	0.013
0.9	0.3	0.632	0.148	0.335	0.002	0.006
0.9	0.6	0.472	0.037	0.193	0.001	0.004
0.6	0.6	0.587	0.094	0.263	0.002	0.004
0.3	0.3	0.802	0.254	0.513	0.007	0.011
0.3	0.6	0.591	0.119	0.285	0.002	0.005
0.3	0.9	0.224	0.017	0.104	0.001	0.002
0.6	0.9	0.152	0.011	0.063	0.001	0.002

Table 3. Comparison of different p_1 and p_2 settings.

Figure 8 shows the performance of all methods on networks of different node average degrees. The performance of all methods deteriorates when the network becomes sparse. Still our method is the leader. Note that in all cases we must first make sure that the network is connected and has a triangulation for computing the virtual embedding as a circular domain.

Figure 9 shows the importance to choose an appropriate TTL. As TTL grows, the delivery rate of our routing method grows rapidly and then stays close to 1, while the other methods do not exhibit a growing trend with TTL. This shows that the Möbius transfor-

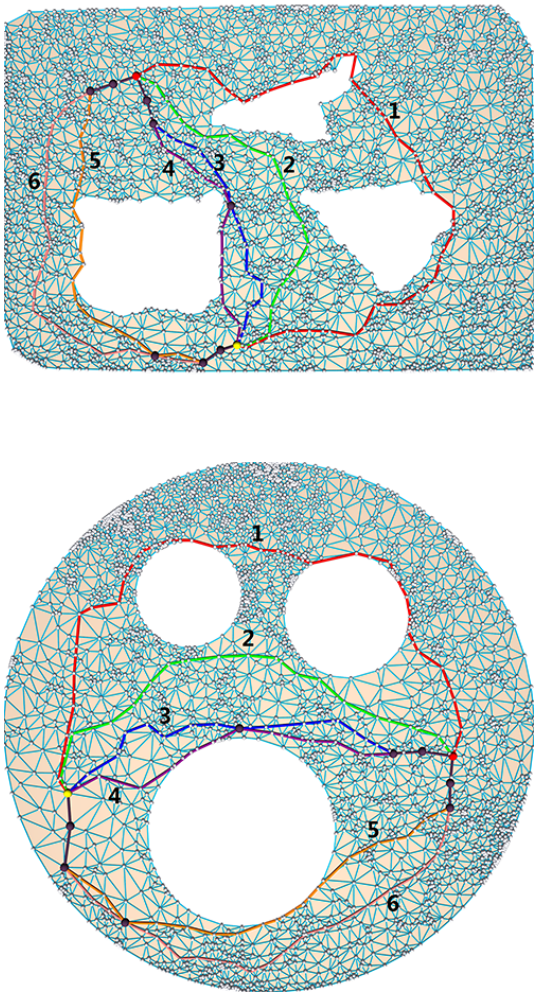

Figure 7. Multipath Routing Algorithm in a region with holes. Up: original network; bottom: network applying ricci flow. Here $\kappa(s,t) = 9$.

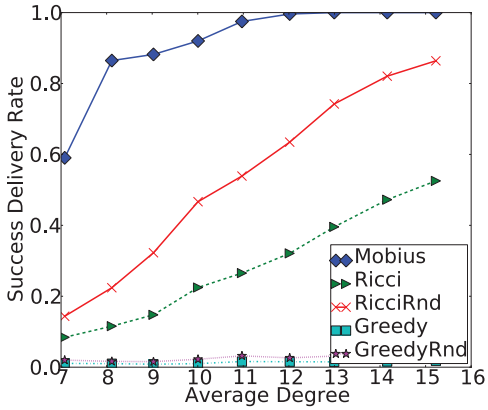

Figure 8. Routing delivery rate versus average degree (TTL = 500; link failure rate = 0.8). **Möbius** is our method. **Greedy** and **Ricci** are greedy routings on the original and Ricci Flow coordinates respectively. **GreedyRand** and **RicciRand** are greedy routings on the original and Ricci Flow coordinates with random walk respectively.

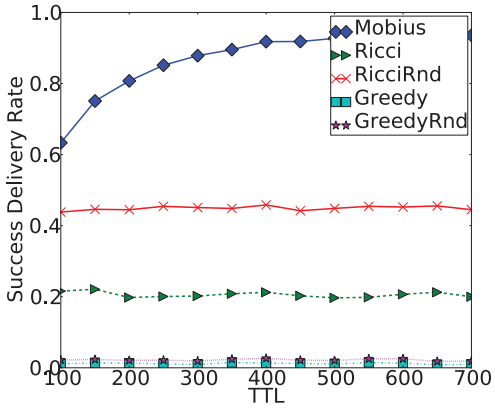

Figure 9. Routing delivery rate versus TTL (time-to-live) of packets (AvgDegree = 10; link failure rate = 0.8).

mation scheme does recover from link failures and makes progress towards the destination, while the other methods still get stuck in the middle. Figure 11 shows the distribution of path length under different methods. Indeed our method gradually delivers more messages when TTL is increased.

Another potential factor affecting the routing performance is the distribution of link failure rates. We test the situation where links lying inside and outside the failure regions have failure rates p_1 and p_2, respectively. We evaluate the performance by varying the difference between p_1 and p_2. From the simulation results shown in Table 3, we can see that our method consistently outperforms the other methods in different settings. When p_1 and p_2 are getting close, **RicciRnd** starts to catch up. In these experiments we also vary the shape and positions of failure regions. While the exact values may vary, the trend is clear: our method makes progress towards the destination despite the existence of high link failure rate, and is unlikely to get stuck in the middle; making long detours is generally better than taking local random walks.

5. CONCLUSIONS

In this paper we presented a method that uses circular domain embeddings and Möbius transformations to switch between them for multipath routing and improving routing resilience. This shows the power of a geometric transformation that regulates a network shape — difficult routing problems due to lack of global knowledge can benefit significantly from such transformations. We expect to extend the intuition to more problems on distributed network setting in the future.

Our multipath routing algorithm in the discrete case uses a heuristic method that maximizes the angular spread of the paths at source and destination. It would be a very interesting problem to exploit the freedom of using such routing scheme to improve network throughput, or optimize energy usage, etc.

We would also like to mention that the paper borrows heavily intuitions that arise from the continuous domain. Our provable results are in the continuous case and the performance of the algorithm in the discrete case is only evaluated by simulations. The problem

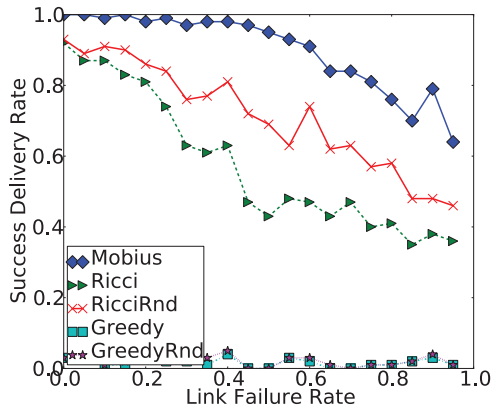

Figure 10. Routing delivery rate versus link failure rate (AvgDegree = 10; TTL = 500).

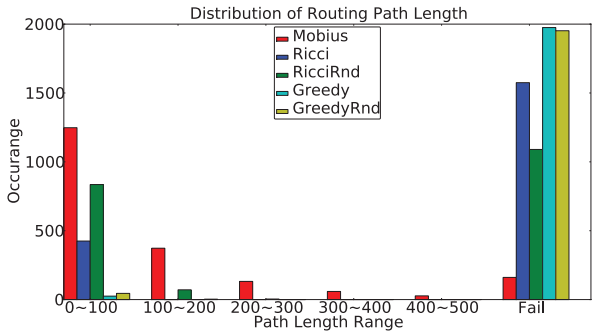

Figure 11. Distribution of routing path lengthes (AvgDegree = 10; TTL = 500; link failure rate = 0.8).

of addressing the gap between the continuous space and a discrete graph, theoretically, is yet an open problem. It would be an interesting and challenging problem to come up with a suitable discrete model and derive minimum density bound. We leave this for future work.

6. ACKNOWLEDGMENTS

Ruirui Jiang, Wei Zeng, Mayank Goswami and Xianfeng Gu have been supported by NSF CNS1016829, IIS0916286, CCF083-0550. Xiaomeng Ban and Jie Gao would like to acknowledge the support from NSF CNS0643687 and CNS1016829.

7. REFERENCES

[1] G. Alandjani and E. Johnson. Fuzzy routing in ad hoc networks. In *Proceedings of the 2003 IEEE International on Performance, Computing, and Communications*, pages 525 – 530, April 2003.

[2] P. Angelini, F. Frati, and L. Grilli. An algorithm to construct greedy drawings of triangulations. In *Proc. of the 16th International Symposium on Graph Drawing*, pages 26–37, 2008.

[3] A. Atlas and A. Zinin. Basic specification for IP fast reroute: Loop-free alternates. IETF RFC 5286, September 2008.

[4] A. Banerjea. Simulation study of the capacity effects of dispersity routing for fault tolerant realtime channels. In *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, pages 194–205, New York, NY, USA, 1996. ACM.

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

[6] J. Cai and W. Wu. Degraded link-disjoint multipath routing in ad hoc networks. In *ISWPC'09: Proceedings of the 4th international conference on Wireless pervasive computing*, pages 149–153, Piscataway, NJ, USA, 2009. IEEE Press.

[7] B. Chow and F. Luo. Combinatorial ricci flows on surfaces. *Journal Differential Geometry*, 63(1):97–129, 2003.

[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1994.

[9] S. De, C. Qiao, and H. Wu. Meshed multipath routing with selective forwarding: an efficient strategy in wireless sensor networks. *Comput. Netw.*, 43(4):481–497, 2003.

[10] 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.

[11] P. Djukic and S. Valaee. Reliable packet transmissions in multipath routed wireless networks. *IEEE Transactions on Mobile Computing*, 5(5):548–559, 2006.

[12] D. Eppstein and M. T. Goodrich. Succinct greedy graph drawing in the hyperbolic plane. In *Proc. of the 16th International Symposium on Graph Drawing*, pages 14–25, 2008.

[13] 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.

[14] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(4):11–25, 2001.

[15] 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.

[16] M. T. Goodrich and D. Strash. Succinct greedy geometric routing in $\mathbb{R}^2$. Technical report on arXiv:0812.3893, 2008.

[17] K. Ishida, Y. Kakuda, and T. Kikuno. A routing protocol for finding two node-disjoint paths in computer networks. In *ICNP '95: Proceedings of the 1995 International Conference on Network Protocols*, page 340, Washington, DC, USA, 1995. IEEE Computer Society.

[18] F. Javadi and A. Jamalipour. Multi-path routing for a cognitive wireless mesh network. In *RWS'09: Proceedings of the 4th international conference on Radio and wireless symposium*, pages 223–226, Piscataway, NJ, USA, 2009. IEEE Press.

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

[20] 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.

[21] 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.

[22] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker. On the pitfalls of geographic face routing. In *DIALM-POMC '05: Proceedings of the 2005 joint workshop on Foundations of mobile computing*, pages 34–43, 2005.

[23] R. Kleinberg. Geographic routing using hyperbolic space. In *Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM'07)*, pages 1902–1909, 2007.

[24] K.-W. Kwong, L. Gao, R. Guerin, and Z.-L. Zhang. On the feasibility and efficacy of protection routing in IP networks. In *INFOCOM'10*, March 2010.

[25] W. K. Lai, S.-Y. Hsiao, and Y.-C. Lin. Adaptive backup routing for ad-hoc networks. *Comput. Commun.*, 30(2):453–464, 2007.

[26] T. Leighton and A. Moitra. Some results on greedy embeddings in metric spaces. In *Proc. of the 49th IEEE Annual Symposium on Foundations of Computer Science*, pages 337–346, October 2008.

[27] X.-Y. Li, G. Calinescu, and P.-J. Wan. Distributed construction of planar spanner and routing for ad hoc networks. In *IEEE INFOCOM*, pages 1268 – 1277, 2002.

[28] C. Liu, M. Yarvis, W. S. Conner, and X. Guo. Guaranteed on-demand discovery of node-disjoint paths in ad hoc networks. *Comput. Commun.*, 30(14-15):2917–2930, 2007.

[29] M. K. Marina and S. R. Das. Ad hoc on-demand multipath distance vector routing. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(3):92–93, 2002.

[30] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala. Path splicing. *SIGCOMM Comput. Commun. Rev.*, 38(4):27–38, 2008.

[31] R. Musaloiu-E. and A. Terzis. Minimising the effect of wifi interference in 802.15.4 wireless sensor networks. *International Journal of Sensor Networks*, 3(1):43–54, 2008.

[32] A. Nasipuri and S. Das. Demand multipath routing for mobile ad hoc networks. In *Proceedings of the 8 th Annual IEEE Internation Conference on Computer Communications and Networks (ICCCN)*, pages 64–70, October 1999.

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

[34] Y. Ohara, S. Imahori, and R. V. Meter. Mara: Maximum alternative routing algorithm. In *Proc. IEEE INFOCOM*, 2009.

[35] C. H. Papadimitriou and D. Ratajczak. On a conjecture related to geometric routing. *Theor. Comput. Sci.*, 344(1):3–14, 2005.

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

[37] S. Ramasubramanian, H. Krishnamoorthy, and M. Krunz. Disjoint multipath routing using colored trees. *Comput. Netw.*, 51(8):2163–2180, 2007.

[38] A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 96–108, 2003.

[39] S. Ray, R. Guérin, K.-W. Kwong, and R. Sofia. Always acyclic distributed path computation. *IEEE/ACM Transactions on Networking*, 2009.

[40] L. Reddeppa Reddy and S. V. Raghavan. Smort: Scalable multipath on-demand routing for mobile ad hoc networks. *Ad Hoc Netw.*, 5(2):162–188, 2007.

[41] C. Reichert, Y. Glickmann, and T. Magedanz. Two routing algorithms for failure protection in IP networks. In *Proc. ISCC*, 2005.

[42] C. Reichert and T. Magedanz. Topology requirements for resilient IP networks. In *Proc. 12th GI/ITG Conf. on Meas., Mod. and Eval. of Comp. and Comm. Sys*, 2004.

[43] J. Ruppert and R. Seidel. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18:548–585, 1995.

[44] 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)*, pages 97–108, April 2009.

[45] R. Sarkar, W. Zeng, J. Gao, and X. D. Gu. Covering space for in-network sensor data storage. In *Proc. of the 9th International Symposium on Information Processing in Sensor Networks (IPSN'10)*, pages 232–243, April 2010.

[46] G. Schollmeier, J. Charzinski, A. Kirstädter, C. Reichert, K. Schrodi, Y. Glickman, and C. Winkler. Improving the resilience in IP networks. In *Proc. HPSR*, 2003.

[47] C. Sengul and R. Kravets. Bypass routing: An on-demand local recovery protocol for ad hoc networks. *Ad Hoc Netw.*, 4(3):380–397, 2006.

[48] J. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1–3):86–95, 2002.

[49] K. Stephenson. *Introduction To Circle Packing*. Cambridge University Press, 2005.

[50] H. Suzuki and F. Tobagi. Fast bandwidth reservation scheme with multi-link and multi-path routing in atm networks. In *Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 2233 –2240 vol.3, may 1992.

[51] W. P. Thurston. *Geometry and Topology of Three-Manifolds*. Princeton lecture notes, 1976.

[52] H. Q. Vo, Y. Y. Yoon, and C. S. Hong. Multi-path routing protocol using cross-layer congestion-awareness in wireless mesh network. In *ICUIMC '08: Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pages 486–490, New York, NY, USA, 2008. ACM.

[53] W. Xu, K. Ma, W. Trappe, and Y. Zhang. Jamming sensor networks: attack and defense strategies. *Network, IEEE*, 20(3):41–47, May-June 2006.

[54] D. Zappala. Alternate path routing for multicast. *IEEE/ACM Trans. Netw.*, 12(1):30–43, 2004.

[55] W. Zeng, R. Sarkar, F. Luo, X. D. Gu, and J. Gao. Resilient routing for sensor networks using hyperbolic embedding of universal covering space. In *Proc. of the 29th Annual IEEE Conference on Computer Communications (INFOCOM'10)*, pages 1694–1702, March 2010.

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

[57] Z. Zhou and J.-H. Cui. Energy efficient multi-path communication for time-critical applications in underwater sensor networks. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 221–230, New York, NY, USA, 2008. ACM.