Compact Conformal Map for Greedy Routing in Wireless Mobile Sensor Networks

Siming Li, Wei Zeng, *Member, IEEE,* Dengpan Zhou, Xianfeng Gu, *Member, IEEE,* and Jie Gao, *Member, IEEE*

Abstract

Motivated by mobile sensor networks as in participatory sensing applications, we are interested in developing a practical, lightweight solution for routing in a mobile network. While greedy routing is robust to mobility, it may get stuck in a local minimum, which then requires non-trivial recovery methods. We find an embedding of the network such that greedy routing using the virtual coordinates guarantees delivery, thus eliminating the necessity of any recovery methods. Our contribution is to replace the in-network computation of the embedding by a preprocessing of the domain before network deployment and encode the map of network domain to virtual coordinate space by using a small number of parameters which can be pre-loaded to all sensor nodes. As a result, the map is only dependent on the network domain and is independent of the network connectivity. Each node can directly compute or update its virtual coordinates by applying the locally stored map on its geographical coordinates. This represents the first practical solution for using virtual coordinates for greedy routing in a sensor network and could be easily extended to the case of a mobile network. The paper describes algorithmic innovations as well as implementations on a real testbed.

Index Terms

Greedy Forwarding, Ricci Flow, Conformal Mapping, Sensor Networks

Siming Li, Dengpan Zhou, Xianfeng Gu, Jie Gao are with the Computer Science Department, Stony Brook University, Stony Brook, NY 11794. E-mail: {silli, dpzhou, gu, jgao}@cs.stonybrook.edu

Wei Zeng is with the School of Computing and Information Sciences, Florida International University, Miami, FL, 33199. E-mail: wzeng@cs.fiu.edu

A preliminary version of this article appeared in the IEEE International Conference on Computer Communications - INFOCOM '13.

Compact Conformal Map for Greedy Routing in Wireless Mobile Sensor Networks

I. Introduction

THIS paper is motivated by applications of participatory sensing, in which participants such as human beings or vehicles carry sensors and may move around inside a domain of interest. We are particular interested in the case of participants in which the density of participants ensures a minimum level of coverage of the domain, as well as a minimum level of quality guarantee for sensing data. The sensor data can go beyond traditional automatic measurements and may exploit voluntary user intervention and inputs. Many interesting applications can be formed in such a setting. Of particular interest to us is the spontaneous sensing opportunities due to incidental spatial proximity. Exploiting the spatial proximity is one of the main ideas in participatory sensing. The participants at the right place and at the right time can conveniently provide required data. Such a system is able to provide data that is very selective and personalized, location specific and time-sensitive, the type of data that would be otherwise difficult to get, and not cost efficient to gather or archive on the web.

While it still remains a major open issue as how to give incentives to participants and how to bootstrap the system, our focus in this paper is on the critical component of how to support efficient routing for queries and answers. It remains an arguable issue whether existing communication infrastructures such as WiFi and 3G/4G connections can support the potentially large number of exchanges of these location-specific, fleeting information. The limited coverage of WiFi signals, the high cost and limited bandwidth of 3G links are nevertheless major hurdles to cross. Here we would like to explore the possibilities of relying on unlicensed spectrum with short-ranged wireless communications for delivering queries and data in such a highly mobile network. Bandwidth aside, we would like to see whether it is possible to manage a mobile ad hoc network for reliable, efficient and low-cost routing for this particular scenario.

A. Prior Work on Routing in Mobile Networks

Routing in a wireless mobile network has been a long standing, active research problem for quite a number of years. While the traditional proposals of dynamic distance vector routing and on-demand routing suffer from high overhead of either maintaining the routing tables or discovering a route to the destination by flooding, geographical routing that relies on local operations based on the geographical positions appears to be an appealing solution for its simplicity. Unfortunately, although geographical routing works nicely in theory [3], [18] for a dynamic, mobile network, it fails in practice, as shown by experiments in real testbeds [19], [27]. Geographical routing has two components, the greedy routing in which a message is delivered to the neighbor closer to the destination, and the recovery method of face routing, which is executed when greedy routing gets stuck and delivers a message along the faces of a planarized graph. In a real world setting, there are a number of complications that make geographical routing challenging. The node location is not always accurate. Wireless communication model does not follow the idealistic unit disk graph model and has various spatial and temporal radio irregularities [14], [30]. Although greedy routing is very robust to such real world issues [25], face routing fails. In practice, the planar subgraph extracted may become disconnected or still contain crossing edges. Face routing on such a graph may either get into a loop or fail to deliver the message although a path exists.

In the last few years, a number of alternative recovery methods have been proposed. In particular, quite a number of them suggest using virtual coordinates such that greedy routing always works. This completely eliminates the necessity of face routing and of course all the practical issues that come with it. This family of work includes both heuristic algorithms NoGeo [25], centralized and theoretical constructions for 3-connected planar graphs [2], [10], [12], [17], [21], [23], embedding in high dimensional spaces [13], embeddings in hyperbolic spaces [12], [20], [22], embedding into circular domains (all the holes are circular) which is referred as CircularEmbedding [26] in this paper . Most of existing work are mainly of theoretical interest. All of these are only for static networks. When nodes move and the network changes its topology, the virtual coordinates need to be recomputed or updated, which is highly non-trivial.

At last we note that since nodes are mobile so theoretically speaking delivery is not an issue — the source can simply hold the message and wait until it gets sufficiently close to the destination. Although being completely impractical, this routing scheme is used in a number of theoretical models to study capacity issues in mobile networks [15]. Similarly, by using greedy routing on geographical locations or any virtual coordinates, in the case of a message getting stuck, the node holding the message could simply wait until mobility brings a new neighbor closer to the destination. Thus message delivery could be ensured at the cost of high delay. Therefore, there is a natural trade-off of delay versus delivery and algorithm efficiency in routing design.

In this paper we investigate compact, lightweight schemes for obtaining virtual coordinates in mobile sensor networks that will help to reduce the delay substantially, or, improve delivery rate when the maximum delay is fixed.

B. Pre-computed Compact Map for Greedy Routing

In this paper we consider a domain R in which a collection of mobile nodes reside. Each node has GPS or other localization schemes to obtain its geographical position. A routing request is formed by specifying the destination's geographical location, rather than the destination's ID – this is because we are interested in finding the nodes near the specified location that has an opportunity to acquire the desirable data, instead of any particular node that may move elsewhere. The nodes may move freely within the domain but we assume the domain R is typically uniformly covered.

Our approach is to use essentially greedy routing for its simplicity and robustness to network dynamics, but avoid face routing completely.

To do so, we would like to understand routing "holes" in a mobile network. There are two types of holes that cause face routing to fail. The first type is due to geometric feature, e.g., obstacles or other domain features in which sensor deployment is impossible. Nodes near such "geometric holes" may run out of neighbors with shorter distance to the destination. This kind of hole is long term and stable. The second type of hole is temporal "routing void" due to irregularity of density. The current node may not have any neighbor in the right direction of destination temporarily. In the mobile network, this kind of hole is short term, fleeting, and may disappear by themselves over time, so the most critical work is to address the holes of the first type. The previous work in static networks [25], [26] generate virtual coordinates based on the *network graph topology* in order to address the holes of both types. In this work we focus on mobile networks and the graph topology is constantly changing. Therefore we propose to use virtual coordinates that mainly address issues caused by holes of the first type (i.e., due to the underlying geometric features). Instead of taking the (constantly changing) network graph topology as input, we take input as the underlying *geometric domain* in which the mobile nodes reside and is more stable than the graph structures.

In particular, we take the geometric domain R and map it in a disk D such that all holes of R are circular. Therefore, we eliminate the first type of holes under the domain D and greedy routing under the virtual coordinates in a network that uniformly sampled on R is unlikely to get stuck. Of course if the density of sensor nodes is non-uniform, a message may still get stuck at the boundary of the second type of holes, but under mobility this situation is not going to last long. This is the first new idea we introduce in this paper.

A second new idea we introduced in this paper is to pre-compute the virtual coordinates for *all* points of the domain R, represent them a compact way by a mapping f, and encode the mapping f within each sensor node. With the help of the hard-coded mapping f, each node p can easily compute its virtual coordinate in real time, by simply taking the value of f on its current GPS coordinates. We also get the virtual coordinates of the destination location as well as that of the neighbors. The next hop is chosen by the greedy rule in the virtual coordinate space.

Our routing method is especially practical for mobile setting, as this algorithm is very light with the calculation of the virtual coordinate totally decided by the encoded mapping function with *current* GPS coordinates as input. The previous methods CircularEmbedding [26] and NoGeo [25] both depend on the network graph topology. When nodes move around and the connectivity of graph changes, the embedding is no longer valid. As a result, the embedding must be recomputed, which could be prohibitively costly. The encoded map we computed ensures that the geometric domain R is transferred to a circular domain D, under which greedy routing itself with the virtual coordinates can be substantially improved. The details are explained below.

The shape of the domain R can be obtained through external sources such as a map and can be represented by a polygon. We will compute the mapping offline (on a centralized machine) and then use a small number of parameters to compactly represent the map. The number of parameters is the same as the complexity of describing the geometric domain R (e.g., the number of vertices of the polygon describing R), and is typically a small number in practice[1] Thus the parameters for the conformal map can be preloaded and programmed on the sensor nodes before deployment and each node can by itself calculate the virtual coordinates with its current geographical location. This representation is compact and stable. It does not depend on the network topology and does not change when the nodes move around. Besides the application in the mobile setting, the new method is also the first practical solution for applying virtual coordinates in a static sensor network.

C. Comparison with Previous Work on Geographic Routing in Mobile Network

While most of the previous work on greedy routing with guaranteed delivery are mainly for theoretical interest, in this section, we specifically compare with [18], [25], [26] as they are the representative work with practical values. We will compare the computational efficiency, and performance of the algorithms in the mobile setting under both dense and sparse densities.

1) Comparison on computational efficiency: [18] first tries greedy routing, when failing, it follows the rule of face routing based on the planar graph of the network. Method such as NoGeo [25] and CircularEmbedding [26] are to compute virtual coordinates by running iterative algorithms according to network topology. In either case, when topology changes due to mobility, the network state needs to be updated. The cost of maintenance is moderate to heavy. Instead, our algorithm only needs the node locations and the pre-defined mapping function. The caculation cost of the virtual coordinates under our algorithm is totally decided by the pre-loaded encoded mapping function which is linear.

[1]There are numerous work on the simplification and approximation of a polygon geometric [8]. By allowing reasonable approximation one can substantially reduce the complexity of a polygon.

2) Comparison on performance on dense network: For a dense network, the method in NoGeo is with decent results on delivery by using virtual coordinates for routing, while CircularEmbedding always guarantees delivery with enough iterations. However, the routing path calculation is on an instantaneous basis, which means the existing path is based on the topology at the current moment. Unless the nodes move extremely slowly, such instantaneous delivery guarantee is unnecessary as a message may not get to the destination before the graph topology already changed.

When nodes densely and uniformly cover the underlying geometric domain R, the holes of the second type rarely exist and the issues of routing with respect to holes of the first type can be handled by the mapping we had. Thus we expect the performance of our algorithm to be fairly nice. If the sensors do not uniformly cover the domain R, our algorithm does not provide instantaneous delivery guarantee as holes of the second type may exist. That is said, a message may be stuck temporarily at a node with no neighbor in the correct direction. However, such cases do not persist in a mobile network.

As shown in our simulations and testbed emulations, our method can guarantee message delivery for densely and uniformly covered networks. Although we can not always deliver under a non-uniformly covered network, we strike for a good balance between maintenance cost and routing quality in terms of delivery rate and delay.

3) Comparison on performance on sparse network: For NoGeo, the algorithm itself can work on sparse networks, however the quality of the coordinates and routing performance could vary a lot. When a network is sparse or the holes are big, the network tend to be largely stretched and nodes are attracted to the outer boundaries. For CircularEmbedding, the input must be a triangulation representing a discrete manifold, which typically requires an average degree around 10 or higher in the experiment. So if the network is not dense enough, we can not even utilize this method. Of course all the algorithm performance will be impacted by network density given that the network might even not be connected under the sparse case. However, our algorithm calculation by itself is not affected by network density, and compared to NoGeo, the performance results under our algorithm are much better.

4) Outline of the paper: The key of the paper is how to compute the compact conformal map which ensures *any* node in geometric domain R is mapped to a virtual node in a circular domain D which guarantees greedy delivery. In the following we first briefly go through the theory of conformal mapping. Then we explain our main contribution in terms of theory, which is how we encode a conformal map of any node in R to a virtual node in D by using Schwarz-Christoffel formula and its inverse. We report results in our simulations and emulations. We implemented and tested the new method on a real testbed (the Orbit testbed [1]), a wireless network emulator with 400 nodes on a 20 by 20 grid. We compared routing using geographical locations and virtual coordinates, in both static and mobile network settings. Experimental results confirm that using the compact conformal map one can substantially improve the delivery rate and largely reduce the packet delay.

II. Background of Conformal Mapping

Given a domain R with an irregular shape and possibly holes, we explore mappings that take it to a circular domain D, in which all boundaries are circular. The mapping we talk about are conformal mappings. A conformal map between two surfaces preserves angles. For any two arbitrary curves γ_1, γ_2 on the surface S, a conformal map ϕ maps them to $\phi(\gamma_1)$, $\phi(\gamma_2)$ with the same intersection angle as that of γ_1, γ_2 on S. In this section, we briefly go through the mathematical theory that describes this mapping and explain the details in the appendix.

A. Surface and Deformation

To deform the domain, two measures are changed, the *metric* that defines distances and the length of a curve, and the *curvature*, which describes how much a geometric object deviates from being flat in the case of a surface, or straight in the case of a curve. A point in the interior of a flat surface has curvature zero; a point on a surface with positive curvature locally has the shape of a hill/valley; a point on a surface with negative curvature locally has the shape of a saddle. Regarding the curvature of a point on the boundary of a surface, the curvature is zero if and only if the boundary locally at the point is straight. All points on the same circular boundary of a planar domain have uniform curvature equals to $1/r$ where r is the radius of this circular curve. The smaller the value r, the more 'curved' this curve is.

In our particular case we need to deform an irregular shaped planar domain R to a circular planar domain D. Since both R and D are flat, all points in the *interior* have zero curvature in both cases. But we need to change the curvature on all *boundary* points in R to be uniform (and thus being circular in D). Consider the simple case of deforming a simple polygon to a disk, as shown in Figure 1. For the boundary of a polygon, the points in the interior of boundary edges have curvature zero; and the vertices have non-zero curvature, defined as the *turning angles* at this vertex between the two adjacent edges. In particular, the curvature at vertex w_1 is $\beta_1\pi$. To make the curvatures at all these vertices the same we need to change the metric as well, i.e., stretching the points in the polygon in certain ways.

B. Ricci Flow

One tool to describe such a deformation is Ricci flow. In particular, Ricci flow deforms the (Riemannian) metric [2]proportional to the local curvature, such that the curvature evolves according to the heat diffusion process. Eventually, curvature at all points converge and the surface has uniform

[2]Riemannian metric g is a symmetric $(0,2)-$tensor that is positive definite (i.e. $g(X,X) > 0$ for all tangent vectors $X \neq 0$).

curvature. In order to deform a surface to be a particular shape, we can specify the target curvature and define Ricci flow by deforming the metric according to the *difference* of the local curvature and the target curvature. When Ricci flow converges we get the shape with the target curvature. In our case for example, if we specify $\mathcal{D}$ such that all points in the interior have zero curvature and points on the same (interior/outer) boundary have uniform (negative/positive) curvature, as Ricci flow converges the surface is deformed to a circular one. The process of Ricci flow defines a map from the original surface R to the deformed surface D. This map is conformal.

Ricci flow is a ground breaking concept and a useful tool in geometric analysis. The surface Ricci flow theory is originally developed by Hamilton [16] for continuous surfaces. It has been applied in the proof of Poincaré conjecture by Perelman in [24]. However, being mathematically defined using notions in continuous geometry, it cannot be directly used in computer applications. The version for discrete surfaces (a surface represented by triangulations) is developed by Chow [6]. The earlier work by Sarkar *et al.* [26] developed the distributed algorithm for implementing Ricci flow in a static sensor network.

In the work by Chow [6] and Sarkar *et al.* [26], discrete Ricci flow is defined based on *discrete curvatures* on a triangulated surface. The surface is represented by a triangulation which describes a two-dimensional manifold (i.e., locally the surface looks like a flat 2D disk). A vertex v in the interior of the triangulation has curvature defined by 2π minus the sum of corner angles (the angles of the triangles with v as one vertex). A vertex v on the boundary has curvature defined by π minus the sum of corner angles. Further, the discrete Ricci flow and conformality of such a discrete mapping are re-defined for the discrete setting, using the notion of circle packing metric (see Appendix).

The discretization of classical Ricci flow theory allowed for applications in computer science domains and thus is very much celebrated. Nevertheless, in this paper we actually go back to conformal map in the original continuous setting which is particlarly suited for our mobile network settings.

C. Our Work

In this work, we are not using a discrete conformal map of a triangulation extracted from the connectivity graph of the mobile nodes, since the connectivity graph keeps changing when nodes move. Instead, we resort to the classical notion of conformal mapping for the continuous geometry. We take the map from the underlying *geometric* domain R and map it to a circular domain D. This mapping f is a continuous mapping which is fundamentally different from the work done by Sarkar *et al.* [26]. The mapping f can be defined by a small number of parameters — the number of these parameters is the same as the complexity of describing R and in practice is small.

We focus on mapping the *whole* network domain R to a circular domain D, which will be explained in detail in the next section. Note that the discrete conformal mapping, e.g., Ricci flow, could be used to map the *boundary* of R to a domain D, which is the prerequisite of the our work. However, this is not the focus of our work.

III. Schwarz-Christoffel Transformation and Laurent Series

In this section, we will explain how we could encode a conformal map function to compute the virtual coordinate within D for *any* node in R. We introduce the Schwarz-Christoffel formula and the Laurent series [9], [11], for simply connected domains (i.e., domains without holes) and multiply connected domains (i.e., domains with holes), respectively, to compute the encoded mapping.

A. Simply Connected Domain

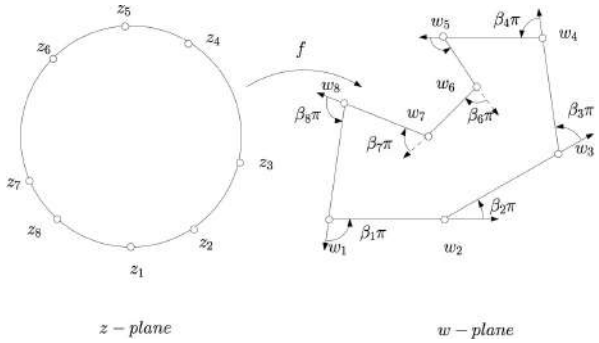


Fig. 1: The Schwarz-Christoffel transformation for a simply connected domain. $f(z_i) = w_i$.

In the case of sensors in a simply connected domain R, as shown in Fig. 1, we would like to find a mapping to a unit disk $\mathbb{D}$. We first look at the conformal mapping from the unit disk to a polygon R, which is a holomorphic function[3], $f : \mathbb{D} \to R$. The mapping we would like to encode is the inverse of f, $f^{-1} : R \to \mathbb{D}$.

Suppose the vertices of the polygon are $\{w_1, w_2, \cdots, w_n\}$. The Schwarz-Christoffel formula [11] describes the mapping of a disk onto the interior of a simple polygon.

$$f(z) = A \int^{z} \prod_{k=1}^{n} (\zeta - z_k)^{-\beta_k} d\zeta + B, \qquad (1)$$

where the pre-image of w_k on the circle is z_k, i.e., $w_k = f(z_k)$; the polygon tangent turning angle at w_k is $\beta_k\pi$; A and B are two constants.

In our case, we need to compute the inverse $f^{-1} : R \to \mathbb{D}$. For convenience, we denote $g(w) = f^{-1}(w)$. Suppose $w_0 = f(0)$. For each $w \in R$, we choose a path connecting w_0 and w, then

$$g(w) = f^{-1}(w) = \int_{w_0}^{w} \frac{dg(w)}{dw} dw + 0.$$

[3]A holomorphic function is a complex-valued function of one or more complex variables that is complex differentiable in a neighborhood of every point in its domain.

where the derivative is given by

$$\frac{dg(w)}{dw} = \left(\frac{df(z)}{dz}\right)^{-1} = \prod_{k=1}^{n}(z-z_k)^{\beta_k} = \prod_{k=1}^{n}(g(w)-z_k)^{\beta_k}.$$

Therefore we can calculate $g(w)$ as:

$$g(w) = \int_{w_0}^{w}\prod_{k=1}^{n}(g(\tau)-z_k)^{\beta_k}d\tau + 0. \quad (2)$$

This is a typical ODE (ordinary differential equation) problem, and can be easily solved by Runge-Kutta method [5]. Therefore, each sensor only requires the parameters $\{z_k, \beta_k\}_{k=1}^n$ and the constants A, B. With such information every sensor can solve the ODE and obtain its own location. The number of parameters stored on each sensors is linear in the complexity to describe R. If we approximate R with simpler polygon we can reduce the storage requirement accordingly.

The evaluation of the conformal map is equivalent to performing a path integration, the shorter the path is, the faster the evaluation is. Therefore in the mobile network setting we can update the virtual coordinates of a mobile node based on its current location, the past location and the past virtual coordinates. This will be faster than direct computation. Suppose at the j-th step, a node's position is $w^{(j)}$, its virtual coordinates is $z^{(j)}$. Then at the $j+1$-th step, if the position is $w^{(j+1)}$, the virtual coordinate $z^{(j+1)}$ can be updated as

$$z^{(j+1)} = \int_{\gamma} f'(\zeta)d\zeta,$$

where γ is the path connecting $w^{(j)}$ to $w^{(j+1)}$.

B. Multiply Connected Domain

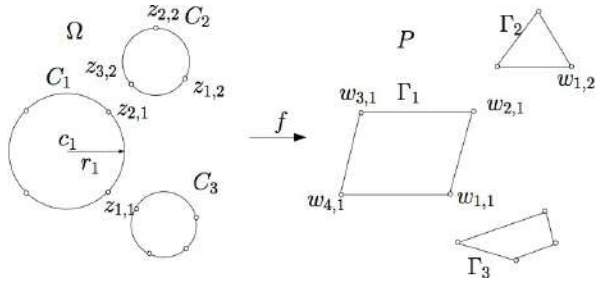


Fig. 2: The Schwarz-Christoffel transformation for a multiply connected domain. $f(z_{i,j}) = w_{i,j}$.

The formulation for conformal mappings for multiply connected domains is very similar. Recall that by using Ricci flow on the polygon R (with $m-1$ holes) describing the domain of interest, we map it to a circular domain $\mathcal{D}$ with $m-1$ circular interior holes. Now we apply a reflection on $\mathcal{D}$ to make it an unbounded domain with m circular holes. A reflection through a circle maps the interior of the circle to the exterior. Suppose p is a point inside the circle $C_j(c_j, r_j)$, then it is reflected to q, where

$$q = \frac{p-c_j}{|p-c_j|^2}r_j^2 + c_j$$

In particular, we can reflect the circular domain $\mathcal{D}$ with respect to the outer boundary and project it to an unbounded domain with m circular holes. This extra reflection is also conformal. For the ease of description, we will assume that $\mathcal{D}$ is an unbounded domain with m circular holes.

Similar to the simply connected case, there is a description of a conformal mapping f from an unbounded domain $\mathcal{D}$ with m circular holes to an unbounded domain R with m polygonal holes. Again what we want to use is the inverse of f, say $g = f^{-1}$.

As shown in Fig. 2, let C_j be the j-th circular hole, with center c_j and radius r_j, which is mapped to a polygonal hole Γ_j, $f(C_j) = \Gamma_j$. The vertices of Γ_j are $\{w_{1,j}, w_{2,j}, \cdots, w_{k_j,j}\} \subset \Gamma_j$. Their pre-images are $\{z_{1,j}, z_{2,j}, \cdots, z_{k_j,j}\} \subset C_j$.

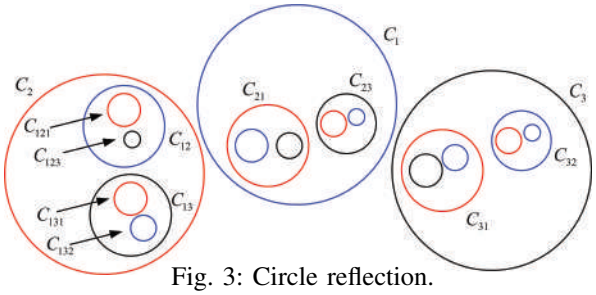


Fig. 3: Circle reflection.

On the circular domain $\mathcal{D}$ we will apply reflections with respect to the circles. As shown in Fig. 3, a circle C_j is reflected through circle C_i to get a circle C_{ij}. A circle C_k is reflected through the circle C_{ij} to get a circle C_{ijk}. More generally, a circle C_{i_n} is reflected through the circle $C_{i_1 i_2,\cdots i_{n-1}}$ to get $C_{i_1 i_2,\cdots,i_n}$. The multi-index $i_1 i_2, \cdots, i_n$ tracks the sequence of reflections. The set of all multi-indices with length n is denoted as σ_n.

The conformal mapping $f : \mathcal{D} \to R$ can be computed using the following formula, called the Laurent series:

$$f(z) = A\int^{z}\prod_{j=1}^{m}\prod_{k=1}^{K_j}\left[\prod_{\substack{n=0\\ \nu\in\sigma_n}}^{\infty}\left(\frac{\zeta - z_{k,\nu j}}{\zeta - s_{\nu j}}\right)\right]^{\beta_{k,j}} d\zeta + B. \quad (3)$$

where $z_{k,\nu j}$ are reflections through circles of prevertices $z_{k,j}$; $s_{\nu j}$ are reflections of circle centers $s_j = c_j$; ν is a multi-index tracking reflections; A and B are two constants.

Similar to the simply connected domain case, the inverse of f can be evaluated by solving an ODE. In practice, instead of using infinitely many reflections, one can reflect only N times, for a reasonable N. In practice N is less than 5. Therefore, the Laurent series could be approximated by:

$$f(z) = A\int^{z}\prod_{j=1}^{m}\prod_{k=1}^{K_j}\left[\prod_{\substack{n=0\\ \nu\in\sigma_n}}^{N}\left(\frac{\zeta - z_{k,\nu j}}{\zeta - s_{\nu j}}\right)\right]^{\beta_{k,j}} d\zeta + B. \quad (4)$$

TABLE I: Parameters of Schwarz-Christoffel transformation.

w	β	z
1 + 1i	0.5	-0.98993 + 0.14155i
20 + 1i	0.5	-0.99478 + 0.10202i
20 + 8i	0.5	-0.99480 + 0.10184i
5 + 8i	-0.5	-0.99978 - 0.02078i
5 + 13i	-0.5	0.97883 - 0.20466i
20 + 13i	0.5	0.99872 - 0.05063i
20 + 20i	0.5	0.99873 - 0.05041i
1 + 20i	0.5	1.00000 + 0.00000i

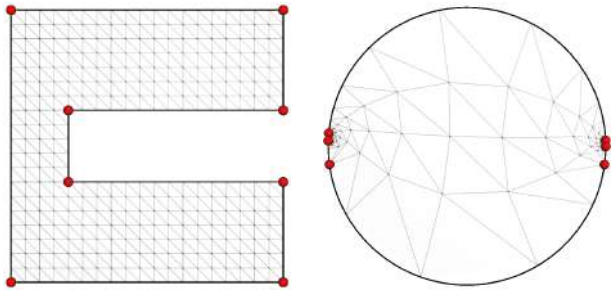
A	-0.16623648 + 2.5343952i
B	3.15 + 10.65i

Note: w - polygon domain, z - disk domain, $\beta\pi$ - tangential turning angle, A - constant scalar in transformation, B - conformal center on w.

In the formula, the prevertices $\{z_{k,j}\}$ and the circle domain $C_j(c_j, r_j)$ are computed using the Ricci flow method offline. Then these values are preloaded to all the sensors with the tangent turning angle $\{\beta_{k,j}\}$ and the constants A, B. Each sensor's virtual coordiantes can be computed through the conformal mapping $f^{-1} : R \to \mathcal{D}$ by solving an ODE.

IV. Examples

In this section, we elaborate the computation of the virtual coordinates using specific examples. These two examples are what we used in the following experiments.



(a) Vertices of R (b) mapped to a circle

Fig. 4: Conformal map from a circle to R using Schwartz-Christoffel formula.

A. Simply Connected Domain

Suppose the sensors move within a simply connected domain as shown in Fig. 4(a). A conformal map f maps a unit disk as shown in Fig. 4(b) to R such that $f(z_i) = w_i$, where w_i are vertices of R. Table I shows the coefficients for the Schwarz-Christoffel transformation, where w are the coordinates of the polygon vertices, z are the prevertices on the unit disk domain, A is a constant scalar, B is the conformal center which is mapped to the origin of the unit disk, β is the tangential turning angle of the input polygon boundary. The prevertices for the interior sensors can be computed by Equation 2 using the obtained coefficients above.

TABLE II: Parameters of Laurent series.

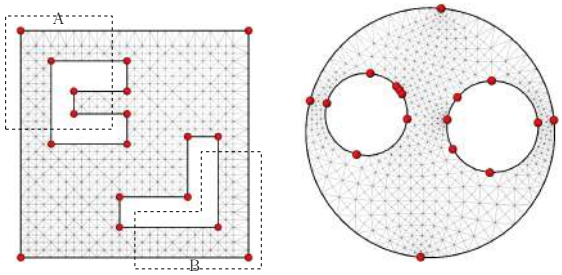
w	β	z
0	0.5	-0.0769204 -0.997173i
30 + 1i	0.5	0.994966 + 0.100535i
30 + 30i	0.5	0.0887658 + 0.995941i
30i	0.5	-0.966776 + 0.256636i
4 + 15i	-0.5	-0.592622 - 0.175069i
4 + 26i	-0.5	-0.832789 + 0.236657i
14 + 26i	-0.5	-0.482482 + 0.47661i
14 + 22i	-0.5	-0.270788 + 0.371865i
7 + 22i	0.5	-0.246845 + 0.342856i
7 + 19i	0.5	-0.246771 + 0.342755i
14 + 19i	-0.5	-0.226475 + 0.311561i
14 + 15i	-0.5	-0.184719 + 0.109243i
13 + 4i	-0.5	0.479915 - 0.319252i
13 + 8i	-0.5	0.182277 - 0.133036i
22 + 8i	0.5	0.138806 + 0.104056i
22 + 16i	-0.5	0.218467 + 0.283291i
26 + 16i	-0.5	0.494831 + 0.417102i
26 + 4i	-0.5	0.867876 + 0.0811398i

(c_1, r_1)	(-0.507853 + 0.159686 i, 0.327225)
(c_2, r_2)	(0.513089 + 0.054973 i, 0.358639)
A	-0.21564355 + 2.27355436i
B	16.45 + 11.65i

Note: w - polygon domain, z - circle domain, $\beta\pi$ - tangential turning angle, (c_k, r_k) - center and radius of hole k on circle domain, A - constant scalar in transformation, B - conformal center on w.

B. Multiply Connected Domain

Suppose the mobile sensors move within a multiply connected domain as shown in Fig. 5 (a). We first use Ricci flow to conformally map the domain to a circle domain, as shown in Fig. 5(b). Through this initial mapping, we obtain the coefficients in Equation 4, as shown in Table II. The prevertices for the interior sensors can be evaluated by solving an ODE to compute the inverse function of Equation 4 with the obtained coefficients.



(a) Input domain R. (b) Mapped to a circular domain.

Fig. 5: Ricci flow for conformal mapping a multiply connected polygon.

V. Experimental Results

In this section, we compare the proposed virtual-coordinate based greedy routing with the traditional greedy routing based on the original geographical locations, and NoGeo routing in both static and mobile network settings.

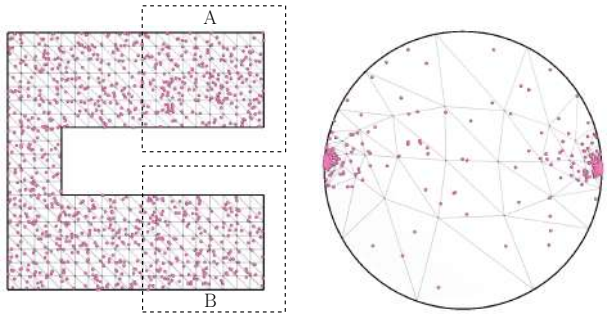
We only compared with greedy routing based on original geographical locations without implementing any recovery scheme (e.g., face routing [18]) when greedy routing (based on whatever coordinates) fails at a local minimum. The reasons for such consideration are the following: first, such recovery schemes are more complicated and are usually developed for an idealistic setting; second, for mobile networks it makes less sense to try to recover from local minimum as nodes move around and a packet may only get stuck temporarily; third, in our applications for participatory sensing and opportunistic query, it is not so critical that every message must be delivered – a scheme with sufficiently high success ratio that emphasizes on efficiency and low overhead will be more desirable. In NoGeo, a set of virtual coordinates are computed by fixing the network outer boundary at a square box and running iterative computations to put each non-boundary node at the center of mass of its neighbors until convergence. We compared our method with CircularEmbedding before, however, we did not specifically compare the performance with it in the experiment part as we already understood from [26] that CircularEmbedding could achieve 100% delivery rate under the condition of non-trivial iteration (which is on the computational scale of 10^3 under the scenarios below) and valid triangulation of the network.

The experiments are performed by both computer simulations and real testbed emulations. The computer simulation is based on a computer generated network with 1000, 400 and 100 nodes. The emulation is deployed on Orbit, an open accessed testbed [1]. We evaluated three performance metrics, namely, *message delay* (how long it takes to deliver a packet from source to destination; *the number of hops* (the number of nodes the packets visited when it is indeed delivered); and *delivery rate* (the fraction of successfully delivered packets).

In the following sections, we first report the results under *dense* networks, in the case of without holes and with holes respectively. We then present the performance of different methods under *sparse* networks. In all the above cases, we can see our method performs better than traditional greedy routing based on the original geographical locations, and NoGeo routing. Specifically, we observed that 1) our method could provide delivery guarantee under the simulation in dense and uniform network for the case without holes reported in section A and with holes in section C. 2) The delivery rate under our algorithm on Orbit is also higher. We measurement of delay on Orbit platform in terms of time instead of hops in the simulation. The new observation compared to simulation results is that we found that the queue formed under the frequently invoked nodes will drive the large deviation between hops and delay as there needs to be a queue to manage the conflict. 3) Under the sparse setting, CircularEmbedding method could not work as no valid triangulation of the whole topology exists. Under the embedding of NoGeo, the nodes are stretched to clearly suboptimal positions which results in less attractive results. The performance of our method on sparse case is much better though.

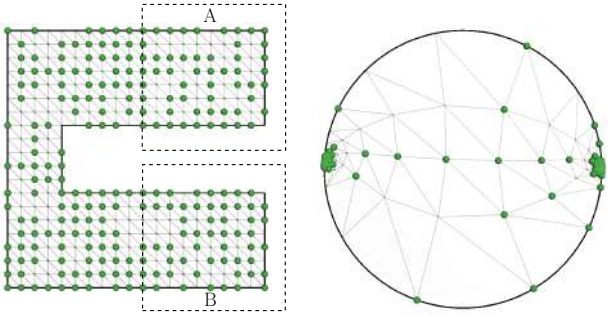
A. Simulations on Dense Network

In the simulations, we considered both static and mobile networks.



(a) mobile sensors (b) virtual coordinates

Fig. 6: Mobile sensors at one snapshot.



(a) active nodes on orbit (b) virtual coordinates

Fig. 9: Active nodes on orbit.

1) Experimental setup: We place 1000 nodes uniformly randomly inside a polygon with the boundary of [1,1], [20,1], [20,8], [5,8], [5,13], [20,13], [20,20], [1,20], as showed in Fig. 6 (a). The communication links are generated by using the Unit Disk Graph (UDG) model with radius of 2. The average degree in the resulted communication graph is about 40. The corresponding domain under conformal mapping to a unit disk is shown in Fig. 6 (b).

All the results here are based on 400 pairs of delivery attempts, which are chosen in two different ways: 1) uniformly and randomly selected from the entire network; 2) uniformly randomly chosen from the special areas A and B in Fig. 6 (a) respectively, under which greedy routing based on geographical locations would have difficulty in delivering the packets, particularly in static setting. The 400 pairs are the same for the two algorithms for fairness.

In the following we only reported the results for simply connected domain. The performance comparison for non-simple domain is similar and thus omitted due to space constraint.

2) Static setting: In this setting nodes are stationary. We set TTL high enough such that packets can be delivered as long as there are valid routes. Thus, the only reason a packet could not be delivered is due to local minimum. The simulation results are shown in Table III.

We can see from the results that the delivery rate of our method is much better than greedy routing with geographical location, especially for the pairs in the specially chosen areas, for which greedy routing using geographical locations always gets stuck. We also have better delivery

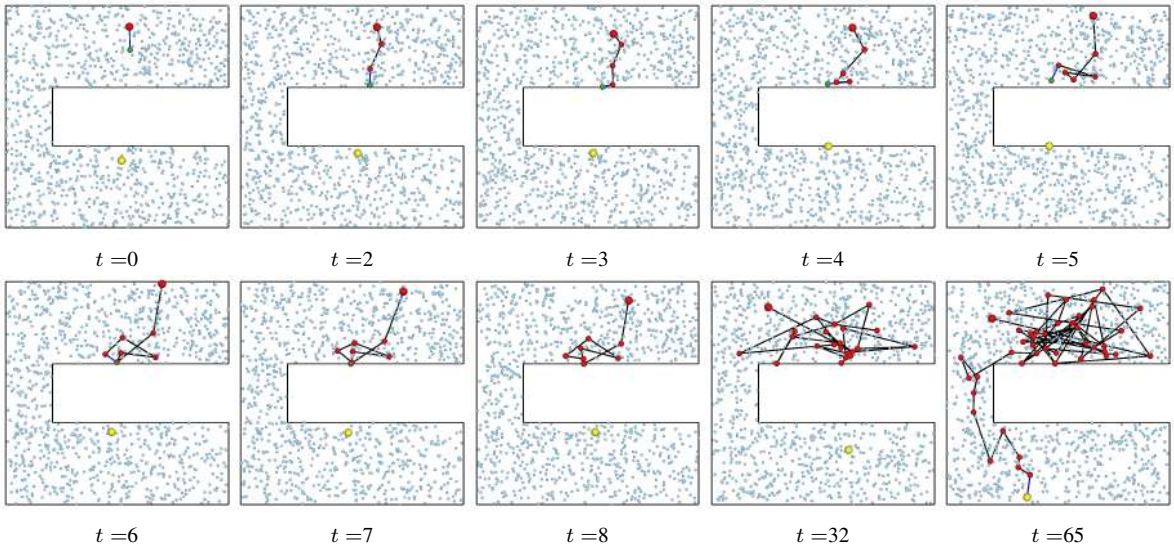


Fig. 7: Greedy routing using real coordinates from the large red node (source) to the yellow node (destination). The green node on each frame shows the node holding the packet at time t. Greedy routing got stuck at time frame 8 and 32.

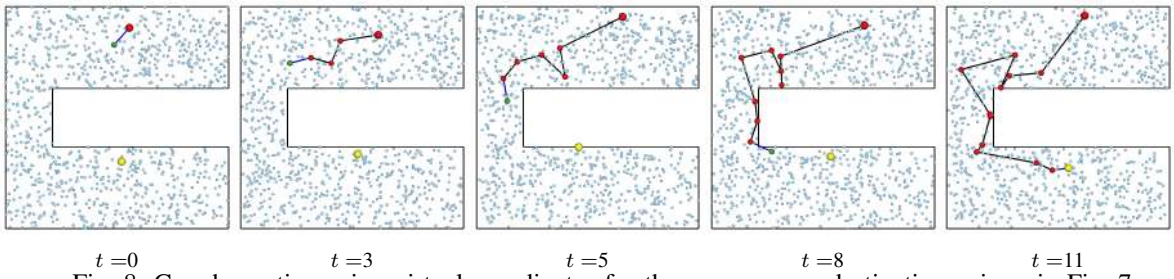


Fig. 8: Greedy routing using virtual coordinates for the same source destination pair as in Fig. 7.

TABLE III: Simulation results on static networks.

Uniformly and randomly selected pairs		
Methods	delivery rate	hops
Geographical locations	275/400	5.6
NoGeo	390/400	8.21
Our method	400/400	8.52
Specially chosen pairs		
Methods	delivery rate	hops
Geographical locations	0/400	0
NoGeo	363/400	16.67
Our method	398/400	17.23

rate than NoGeo method. The number of hops listed in the table is the average value of hops for successfully delivered message. The hops for our method is higher than that of greedy routing with geographical locations, simply because our method can successfully deliver more messages, including the far apart pairs in the special area for which greedy routing using geographical locations fails. For the 275 pairs that could be delivered by greedy routing using geographical locations, the average number of hops using our method is only 5.88, only slightly higher than 5.6 using geographical locations.

3) Mobile setting: To evaluate our algorithm for the mobile setting, we adopt the popular Random Waypoint (RWP) mobility model [4]. Each node moves to a destination, uniformly randomly chosen in the domain R, along a straight line with fixed velocity. Upon arrival at the destination it stays for a period of time t (uniformly randomly drawn from $[0, 2]$), chooses the next destination and moves there along a line. The moving speed for node i is drawn from $[0.5, 2.5]$ uniformly randomly. We sample the time axis to generate a sequence of snapshots, such that we only construct the communication neighbors and perform routing attempts at each snapshot. The sampling rate is uniformly distributed in the range $[1, 3]$.

When the node with the packet is within transmission range of the destination location, the packet is considered delivered successfully. If the package gets stuck, it remains at the current node for another trial in the next time slot. Notice that in the next snapshot, the node with the packet moves to a different location and may have a new set of neighbors. See Fig. 7 and Fig. 8 for examples. We use TTL to control how long the packets could wander in the network. When a packet is delivered, we measure the delay as the number of snapshots experienced when the packet arrives at the destination. We measure the number of hops of a successfully delivered packet as the number of relay nodes on the path from its source to the destination. The results are concluded in Table IV.

We can observe from the results that the delivery rate is substantially improved when mobility is allowed, both for greedy routing using geographical locations, NoGeo method and our method. Nevertheless we could deliver the

TABLE IV: Performance of the methods under mobile setting of computer simulation.

Geographical locations method on uniformly randomly chosen pairs				Our method on delivered pairs under geographical locations		
TTL	delivery rate	hops	delay	delivery rate	hops	delay
10	238/400	5.15	5.15	215/238	5.02	5.02
20	286/400	6.28	6.29	286/286	6.58	6.58
30	295/400	6.69	6.76	295/295	6.79	6.79
200	338/400	17.39	19.36	338/338	7.95	7.95
Geographical locations method on specially chosen pairs				Our method on delivered pairs under geographical locations		
TTL	delivery rate	hops	delay	delivery rate	hops	delay
10	0/400	0	0	0/0	0	0
20	1/400	18	19	1/1	18	18
30	2/400	22.5	23.5	2/2	19	19
200	173/400	95.62	110.8	173/173	19.02	19.02
Our method on randomly chosen pairs				NoGeo method on randomly chosen pairs		
TTL	delivery rate	hops	delay	delivery rate	hops	delay
10	231/400	5.28	5.28	220/400	5.13	5.13
20	379/400	8.43	8.43	382/400	8.68	8.68
30	400/400	9.12	9.12	400/400	9.22	9.22
Our method on specially chosen pairs				NoGeo method on specially chosen pairs		
TTL	delivery rate	hops	delay	delivery rate	hops	delay
10	0	0	0	0	0	0
20	254/400	16.50	16.50	248/400	16.67	16.69
30	400/400	18.40	18.40	399/400	18.43	18.46

messages with much smaller delay and hops. In particular, notice that in our experiments the values for hops and delay are always the same for our method. This means our method never gets stuck — in every time slot we are able to find a neighbor that is closer to the destination; the message is never held at a node for more than one time unit. This explains the reduced delay and the high delivery rate. For the messages that are indeed delivered within small TTL in greedy routing method using geographical locations, the delay and hop counts experienced are similar to that in our methods. When the TTL is large, the pairs delivered under geographical locations method have more hops and delay compared to our method. This basically says that for messages that start at positions whose straight line paths to the destination are not 'blocked', and they can be delivered fairly quickly and easily, using both methods. But for messages that get stuck, using geographical locations they will need to wait for a long time for the node holding them to move out of that region; using our methods they have no problem to find good neighbors.

Although in terms of delivery rate our method and NoGeo method are comparable, the computational cost under NoGeo is far heavier than ours. Under our method, for each TTL the virtual coordinate for each node is obtained locally using a small number of pre-calculated parameters; while in NoGeo one needs to recompute the virtual coordinates for around 160 iterations. Note that, if we aim for lower computational cost, say, updating the coordinates every 5 TTL under NoGeo, the delivery performance is not as good as the results we show in Table IV, with delivery rate as $246/300$ for special chosen pairs. Thus our method has superior delay and cost tradeoffs.

B. Emulation on Orbit

In this section, we demonstrate results from emulation experiments on a real testbed to validate our algorithm and compare to greedy routing with geographical locations. The emulation is deployed on the publicly accessible Orbit testbed [1]. This testbed consists of 400 nodes (standard Linux PCs), and it supports both wired and wireless experiments. The nodes are placed in a two-dimensional rectangular grid with 1 meter spacing and wireless antennas mounted on the sides.

1) Experimental setup: The network we created on Orbit has the same boundary as what we used in the simulations. There are 340 nodes in this C-shape area. Among them 233 nodes (shown in Fig. 9 (a)) of the grid are active nodes on Orbit. The rest nodes are artificially removed from our experiments. In our experiments, not all nodes selected are available to use and nodes may go down for no reason. So we do not have uniformly distributed nodes per our observation. The corresponding map of the network and virtual coordinates is shown in Fig. 9 (b).

The network topology is created with a unit disk graph model with communication range of 2.5. For two nodes within communication range, we use the wired communication channel on Orbit for packet transmission in the experiments, since the wireless radios on Orbit were set up with very large communication range, essentially covering the entire network. UDP is chosen as the communication protocol between the nodes. Although the communication graph is selected from a unit disk graph model, the packet is not always delivered even if the sender and receiver are within transmission range. This is observed from our experiments and could be observed from the results reported later.

Each of the node is equipped with two functions. 1) Initialize a packet as the sender and send it to the next node. 2) Listen and forward the message to the next node. The two functions are designed as threads, so the nodes can have the two functions at the same time. This simulates the scenario of a node in full duplex mode. We utilized a link table to filter packets at the application layer so that each node only receives the messages from its neighbors within communication range.

Since the local clocks at the Orbit nodes are not consistent, we cannot directly take the difference of the packet departure and arrival time. Instead, we work around this issue by sending a confirmation from the destination back to the source. Therefore the delay reported in our result actually includes this extra 'acknowledgement' step. Notice that since this extra step is added to both methods the comparison is still fair. The value for delay reported later is the median value.

TABLE V: Performance of the methods under static setting of Orbit emulation.

Uniformly randomly chosen pairs			
Methods	delivery rate	hops	delay (ms)
Geographical locations	27/40	5.8	25.669
Virtual coordinates	40/40	8.23	29.631
Specially chosen pairs			
Methods	delivery rate	hops	delay (ms)
Geographical locations	0/40	0	0
Virtual coordinates	38/40	16	33.270

2) Static setting: 40 pairs of source and destination are chosen among the above 233 nodes for routing. The nodes' geographical coordinates are their grid coordinates. Same as in computer simulation, the pairs are uniformly randomly chosen and chosen from area A and B in Fig. 9 respectively in two different ways.

Each node can obtain the locations of other nodes from the global node location table. And nodes can get their neighbor information from the global link table. To run our experiments, we issue 40 pairs of packets through the central controller. The sender initializes a packet with destination information and sends it to one of the neighbors according to the greedy algorithm. Since all the nodes listen in the network. If a packet is sent to a node, the node catches it. The node will then forward the packet to the next node until the packet arrives at the destination. The choice of which neighbor to send packet to depends on the node's location and connection specified in the global node locations table and global link table. The experiment setups for both greedy routing with geographical coordinates and virtual coordinates are the same except the different set of coordinates.

We compare delay, hops and delivery rate under the two methods, with randomly chosen pairs and specially chosen pairs. The results are listed in Table V.

We simulate the experiment with all the same setting on the computer, and the delivery rate for virtual coordinates routing under special chosen area is 40/40. Notice that some of the delivery rate under our simulations (the ideal setting) is higher than the results we get from Orbit experiments. The reasons are 1) the Orbit network may experience unknown stability issues and packets may get lost for no reason (since UDP is used); 2) two packets may arrive at a node at the same time. Thus due to competition (similar to wireless interference) one of them may be lost; 3) the nodes may be go up and down during the experiments Nevertheless this represents a practical setting in which transmission failures may as well occur.

Under the same network condition, our method consistently works better than greedy routing with geographical locations, especially for pairs in the special chosen areas. The results are similar to the results obtained in our computer simulations.

3) Mobile setting: The nodes on Orbit are static. In order to mimic a mobile network on Orbit, we generate a mobile network scenario using simulations and then map the mobile nodes in our simulations onto the Orbit nodes. A mobile node is mapped to the closest Orbit node. In this case, we can use the movement of nodes from computer simulation to emulate the mobility of nodes on Orbit. Thus, the nodes move in the same way as reported earlier and the actual transmission are done on Orbit.

TABLE VI: Performance of the methods under mobile setting of Orbit.

Greedy routing using geographical locations on uniformly randomly chosen pairs			
TTL	delivery rate	hops	delay (ms)
10	21/40	5.20	31.215
20	24/40	6.12	32.275
30	25/40	6.71	33.130
200	29/40	10.56	43.873
Greedy routing using geographical locations on specially chosen pairs			
TTL	delivery rate	hops	delay (ms)
10	0/40	0	0
20	0/40	0	0
30	1/40	12	93.978
200	10/40	50.72	629.985
Our method using virtual coordinates on uniformly randomly chosen pairs			
TTL	delivery rate	hops	delay (ms)
10	21/40	5.22	31.582
20	33/40	8.27	32.232
30	37/40	9.01	32.775
Our method using virtual coordinates on specially chosen pairs			
TTL	delivery rate	hops	delay (ms)
10	0/40	0	0
20	21/40	14.28	52.536
30	32/40	16.21	71.700

40 pairs of packets are chosen randomly uniformly and from the special area same as the above experiments respectively. Different from computer simulation, the delay on real testbed of Orbit is the real time difference between the message sent and received. The results are reported in Table VI.

The observation that mobility helps to improve delivery rate still holds for the testbed experiments, for both geographical greedy routing and our method. Again our method gives much smaller delay and fewer number of hops. We may notice that there could be a large deviation between the hop counts and packet delay, in particular for geographical greedy routing for specially selected pairs. As in geographical greedy routing, many packets go through some small number of critical nodes. In our simulations we assume that all these packets go through with ease – an unrealistically ideal case. But on Orbit, when an Orbit node is called by several packets (invoked frequently), a queue is formed and this could substantially increase the delay.

TABLE VII: Performance of the methods under static setting of computer simulation with holes.

Uniformly randomly chosen pairs without additional boundary nodes		
Methods	delivery rate	hops
Geographical locations	315/400	6.63
Virtual coordinates	374/400	7.40
Specially chosen pairs without additional boundary nodes		
Methods	delivery rate	hops
Geographical locations	73/400	15.6
Virtual coordinates	136/400	15.54
Uniformly randomly chosen pairs with additional boundary nodes		
Methods	delivery rate	hops
Geographical locations	326/400	6.64
Virtual coordinates	400/400	7.62
Specially chosen pairs with additional boundary nodes		
Methods	delivery rate	hops
Geographical locations	0/400	0
Virtual coordinates	400/400	15.58

C. Simulations on Networks With Holes

We also run simulations in a domain with holes for both static and mobile network setting.

1) Experimental setup: In the simulations, we throw 1000 nodes uniformly and randomly inside a domain with holes as showed in Fig. 5 (a). 1000 nodes are uniformly distributed inside the domain. The communication range of the nodes also follows the Unit Disk Graph (UDG) model. The corresponding conformal mapping results of the network under our algorithm is shown in Fig. 5 (a).

We run routing tests for 400 pairs of source and destinations. The 400 pairs of source and destination are also chosen in two different ways mentioned above, they are randomly and uniformly selected from the special areas A as source and B as destination in Fig. 5 (a) respectively. The 400 randomly chosen pairs are the same for the two algorithms for fairness.

In the experiments, we also include additional 222 static sensor nodes on the hole boundaries to the network, in addition to the 1000 nodes in the interior. This improves the performance of our algorithm, as shown later. To understand this, recall that our conformal map is applied for the *continuous* geometric domain, while the sensor network is a discrete network that may not always cover the entire domain nicely. In particular, there may not be sensors on the circular hole boundaries and thus the real 'hole' in the sensor network may have tiny sawtooth-type variations. This may create cases when a packet can locally get stuck on such a varied hole boundary. Placing additional fixed sensor nodes on the hole boundaries greatly help to reduce the number of such scenarios.

2) Static setting: We present the simulation results in Table VII. We can see from the results that our observations still hold – the delivery rate of our method is much better than greedy routing with geographical location. Placing additional nodes on the domain boundary also helps to further improve the delivery rate of our algorithm. What is interesting (and a bit surprising) is that placing these additional boundary nodes may make geographical greedy routing worse, especially for the specially chosen pairs! Because additional boundary nodes make sure that geographical greedy routing definitely direct the messages to the 'pockets' in the domain and get stuck there.

3) Mobile setting: In the mobile setting, the movement of nodes follows random waypoint mobility model [4], the same as the previous simulations but with different parameters. Specifically, we set the moving disk radius r for all nodes to be 3. The moving speed v_i for each node i is drawn from $[1, 4]$ uniformly and independently. When node i comes to stay stage, stay duration t_i is uniformly drawn from $[0, 3]$. Finally, the sample time duration for the next snapshot is uniformly distributed in the range $[1, 4]$.

The results are concluded in Table VIII and Table IX. For the mobile network setting including the static boundary nodes always helps our proposed methods.

D. Simulations on Sparse Mobile Network

In this section, we study the performance of our algorithm, NoGeo and greedy routing on original geographical location on sparse mobile network. Note that CircularEmbedding algorithm by itself cannot work when the network is sparse as no valid triangulation exists.

1) Experimental setup: In this section, we examine the performance of different algorithms on sparse networks. We use the same network domain as shown in Fig. 6 (a). The communication links are also generated by using the Unit Disk Graph (UDG) model with radius of 2. We uniformly randomly place 100 and 300 nodes in respective to generate two networks, which are relatively sparse compared to the network we studied above and with average degree 0.5 and 3.8 respectively. The node trajectory follows the same Random Waypoint (RWP) mobility model as adopted in the dense network above. Same as the dense network case, the virtual coordinates under our algorithm are calculated by the pre-loaded parameters and the original coordinates. We would like to emphasize that the pre-loaded parameters are independent of network density and only decided by the underlying geographic domain.

2) Results on mobile setting: Table X shows the results of both networks under 400 randomly chosen pairs as source and destination, and Table XI is for the results of both networks under 400 specially chosen pairs from special area A to special area B in Fig. 6 (a). For the performance of randomly chosen pairs in Table X, although the delivery rates of geographical location method are comparable to our method at the beginning, our delivery rates increase much faster for both networks. For the performance of specially chosen pairs in Table XI, our performance is much better than geographical location routing, no matter under very sparse network with 100 nodes

TABLE VIII: Performance of the methods without boundary nodes participation under mobile setting of computer simulation with holes.

Greedy routing using geographical locations on uniformly randomly chosen pairs, without boundary nodes			
TTL	delivery rate	hops	delay
10	230/400	5.15	5.15
20	341/400	7.60	7.66
30	356/400	8.19	8.35
50	369/400	9.16	9.50
100	378/400	10.42	11.15
150	385/400	11.91	13.29
200	387/400	12.57	14.23
Greedy routing using geographical locations on specially chosen pairs, without boundary nodes			
Methods	delivery rate	hops	delay
10	0/400	0	0
20	82/400	14.96	15.21
30	106/400	16.39	16.75
50	142/400	22.03	23.99
100	200/400	32.32	38.02
150	279/400	48.14	61.80
200	316/400	57.63	74.94
Greedy routing using virtual coordinates on uniformly randomly chosen pairs, without boundary nodes			
TTL	delivery rate	hops	delay
10	247/400	5.35	5.36
20	394/400	8.14	8.16
30	398/400	8.29	8.32
50	399/400	8.34	8.37
Greedy routing using virtual coordinates on specially chosen pairs, without boundary nodes			
Methods	delivery rate	hops	delay
10	5/400	9	9
20	266/400	15.03	15.10
30	389/400	17.25	17.40
50	398/400	17.59	17.78

TABLE IX: Performance of the methods with boundary nodes participation under mobile setting of computer simulation with holes.

Greedy routing using geographical locations on uniformly randomly chosen pairs, with boundary nodes			
TTL	delivery rate	hops	delay
10	242/400	5.21	5.21
20	352/400	7.41	7.52
30	358/400	7.60	7.80
50	367/400	8.04	8.57
100	376/400	8.71	10.26
150	382/400	9.17	12.09
200	385/400	9.50	13.40
Greedy routing using geographical locations on specially chosen pairs, with boundary nodes			
Methods	delivery rate	hops	delay
10	0/400	0	0
20	65/400	14.58	15.05
30	72/400	15.10	15.82
50	119/400	20.78	26.13
100	187/400	25.00	43.17
150	273/400	29.27	67.44
200	311/400	31.58	80.85
Greedy routing using virtual coordinates on uniformly randomly chosen pairs, with boundary nodes			
TTL	delivery rate	hops	delay
10	258/400	5.44	5.44
20	397/400	7.91	7.91
30	400/400	8.02	8.02
Greedy routing using virtual coordinates on specially chosen pairs, with boundary nodes			
Methods	delivery rate	hops	delay
10	3/400	9	9
20	340/400	15.28	15.28
30	400/400	16.24	16.24

or less sparse network with 300 nodes. Our performance is also better than NoGeo method, especially under the very sparse network with 100 nodes. What is interesting is that the improvement of our algorithm compared to NoGeo is higher on sparse networks than on dense case. This is because NoGeo is more sensitive to the network density. On sparse networks, the virtual coordinates under NoGeo are attracted to the suboptimal positions near the outer boundaries.

We also analyzed different types of holes in the network to understand how often they appear in routing. We examine when a message temporarily get stuck (having no immediate neighbors making progress towards the destination) whether the node is currently on the domain boundary or not. In the former case we say it is a type-1 hole; in the latter case we say it is a type-2 hole. Note that under our method there is no type-1 hole, and all local minima are due to type-2 holes. Table XII reports the percentage of pairs among all the delivered pairs ever experiencing type-1 and type-2 holes under geographical routing and our method respectively. We can see from the data that the percentage of type-2 holes under our method and geographical routing are similar with the same TTL counts. This is reasonable as we are not optimizing for type-2 holes. We can also observe from the results that when the network is not extremely sparse, e.g., under 300 nodes with 3.8 degree, type-2 hole is rarely seen. We also did an experiment for 400 nodes in the same network with 6 degree when we didn't see any type-2 hole under our method.

In summary, under the sparse cases, our results in terms of delivery rate are always better than geographical method and NoGeo. Compared to the other methods, our algorithm is robust to network density and especially good on sparse network. Since type-2 hole is not prominent at all when network is not extremely sparse, our method has great advantage to work on sparse networks.

TABLE X: Performance on sparse simulated network by uniformly randomly chosen pairs.

Geographical locations method on the network with 100 nodes				Geographical locations method on the network with 300 nodes			
TTL	delivery	hops	delay	TTL	delivery	hops	delay
10	109/400	4.23	4.96	10	194/400	4.37	4.40
30	285/400	9.64	12.84	30	308/400	8.02	8.49
50	308/400	10.5	14.70	50	316/400	8.54	9.25
100	329/400	11.98	18.24	100	330/400	10.18	11.58
Our method on the network with 100 nodes				Our method on the network with 300 nodes			
TTL	delivery	hops	delay	TTL	delivery	hops	delay
10	110/400	4.41	5.10	10	197/400	4.66	4.69
30	286/400	10.65	13.62	20	352/400	8.76	8.81
50	347/400	13.39	18.17	30	400/400	10.37	10.42
100	397/400	16.36	23.66	100			
NoGeo method on the network with 100 nodes				NoGeo method on the network with 300 nodes			
TTL	delivery	hops	delay	TTL	delivery	hops	delay
10	73/400	3.66	4.27	10	157/400	4.57	4.64
30	172/400	9.80	12.67	30	386/400	11.89	12.30
50	248/400	14.97	20.60	50	400/400	12.50	13.10
100	321/400	22.63	32.08	100			

TABLE XI: Performance on sparse simulated network by uniformly specially chosen pairs.

Geographical locations method on the network with 100 nodes				Geographical locations method on the network with 300 nodes			
TTL	delivery	hops	delay	TTL	delivery	hops	delay
30	0/400	0	0	30	2/400	20.50	28.00
50	3/400	19.33	31.00	50	14/400	33.21	40.36
70	9/400	29.22	53.00	70	37/400	37.89	46.05
100	16/400	37.63	71.38	100	74/400	48.85	66.64
Our method on the network with 100 nodes				Our method on the network with 300 nodes			
TTL	delivery	hops	delay	TTL	delivery	hops	delay
30	0/400	0	0	10	0/400	0	0
50	190/400	28.81	41.11	20	77/400	17.51	17.51
70	368/400	34.07	49.63	30	392/400	22.64	22.68
100	400/400	35.09	51.53	50	400/400	22.78	22.83
NoGeo method on the network with 100 nodes				NoGeo method on the network with 300 nodes			
TTL	delivery	hops	delay	TTL	delivery	hops	delay
30	0/400	0	0	10	0/400	0	0
50	21/400	29.7	41	20	51/400	17.75	17.90
70	47/400	36.02	51.17	30	257/400	21.86	22.63
100	122/400	49.56	71.03	50	400/400	25.44	27.17

VI. Conclusion

This paper describes a compact way to represent a conformal map and applies it to routing in both static and mobile networks. As the first practical solution to using virtual coordinates with greedy routing, we expect to see implementations of our method in real world applications.

Acknowledgment

The authors would like to acknowledge the support from NSF (DMS-1418255, DMS-1221339, CNS-1217823, CNS-1016829, CCF-1544267), AFOSR (FA9550-14-1-0193, FA9550-10-1-0294), and the Natural Science Foundation of China (under no. 61328206).

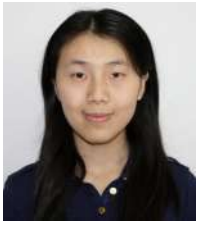
TABLE XII: Type of holes analysis on sparse simulated network by uniformly randomly chosen pairs.

Geographical locations method on the network with 100 nodes				Geographical locations method on the network with 300 nodes			
TTL	delivery	Type 1	Type 2	TTL	delivery	Type 1	Type 2
10	109/400	12/109	38/109	10	194/400	2/194	4/194
30	285/400	85/285	199/285	30	308/400	44/308	22/308
50	308/400	105/308	222/308	50	316/400	52/316	25/316
100	329/400	123/329	243/329	100	330/400	66/330	30/330
Our method on the network with 100 nodes				Our method on the network with 300 nodes			
TTL	delivery	Type 2		TTL	delivery	Type 2	
10	110/400	44/110		10	197/400	5/197	
30	286/400	202/286		20	352/400	15/352	
50	347/400	263/347		30	400/400	17/400	
100	397/400	313/397					

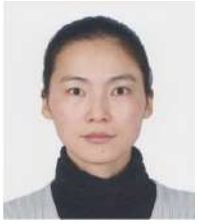
References

[1] Orbit testbed. http://www.orbit-lab.org/
[2] P. Angelini, F. Frati, and L. Grilli, "An algorithm to construct greedy drawings of triangulations," *Proc. of the 16th International Symposium on Graph Drawing*, pp. 26–37, 2008.
[3] 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.
[4] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," *Proc. of the ACM/IEEE Mobile Computing and Networking(MobiCom)*, pp. 85–97, 1998.
[5] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, New York: John Wiley & Sons, 2002.
[6] B. Chow, "The ricci flow on the 2-sphere," *J. Differential Geom.*, 33(2):325–334, 1991.
[7] B. Chow and F. Luo, "Combinatorial ricci flows on surfaces," *Journal Differential Geometry*, 63(1):97–129, 2003.
[8] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, Berlin, 1997.
[9] T. DeLillo and E. Kropf, "Numerical computation of the schwarz-christoffel transformation for multiply connected domains," *SIAM J. on Scientific Computing*, 33:1369–1394, 2011.
[10] R. Dhandapani, "Greedy drawings of triangulations," *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 102–111, 2008.
[11] T. A. Driscoll and L. N. Trefethen, *Schwarz-Christoffel Mapping*, volume 8, Cambridge University Press, 2002.
[12] D. Eppstein and M. T. Goodrich, "Succinct greedy graph drawing in the hyperbolic plane," *Proc. of the 16th International Symposium on Graph Drawing*, pp 14–25, 2008.
[13] R. Flury, S. Pemmaraju, and R. Wattenhofer, "Greedy routing with bounded stretch," *Proc. of the 28th Annual IEEE Conference on Computer Communications (INFOCOM)*, April 2009.
[14] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "Complex behavior at scale: An experimental study of low-power wireless sensor networks," Technical Report UCLA/CSD-TR 02-0013, UCLA, 2002.
[15] M. Grossglauser and D. Tse, "Mobility increases the capacity of ad hoc wireless networks," *IEEE/ACM Transactions on Networking*, 10(4):477–486, August 2002.
[16] R. S. Hamilton, "Three manifolds with positive ricci curvature," *Journal of Differential Geometry*, 17:255–306, 1982.
[17] X. He and H. Zhang, "On succinct convex greedy drawing of 3-connected plane graphs," *Proceedings of the ACM-SIAM symposium on Discrete algorithms*, pp 1477–1486, January 2011.

[18] B. Karp and H. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pp 243–254, 2000.
[19] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "On the pitfalls of geographic face routing," *DIALM-POMC '05: Proceedings of the 2005 joint workshop on Foundations of mobile computing*, pp 34–43, 2005.
[20] R. Kleinberg, "Geographic routing using hyperbolic space," *Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM'07)*, pp 1902–1909, 2007.
[21] T. Leighton and A. Moitra, "Some results on greedy embeddings in metric spaces," *Proc. of the 49th IEEE Annual Symposium on Foundations of Computer Science*, pp 337–346, October 2008.
[22] P. Maymounkov, "Greedy embeddings, trees, and euclidean vs. lobachevsky geometry", manuscript, 2006.
[23] C. H. Papadimitriou and D. Ratajczak, "On a conjecture related to geometric routing," *Theor. Comput. Sci.*, 344(1):3–14, 2005.
[24] G. Perelman, "The entropy formula for the ricci flow and its geometric applications," Technical Report arXiv.org, November 11 2002.
[25] A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," *Proceedings of the 9th annual international conference on Mobile computing and networking*, pp 96–108, 2003.
[26] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. D. Gu, "Greedy routing with guaranteed delivery using ricci flows," *Proc. of the 8th International Symposium on Information Processing in Sensor Networks (IPSN'09)*, pp 97–108, April 2009.
[27] K. Seada, A. Helmy, and R. Govindan, "On the effect of localization errors on geographic face routing in sensor networks," *IPSN '04: Proceedings of the third international symposium on Information processing in sensor networks*, pp 71–80, 2004.
[28] K. Stephenson, *Introduction To Circle Packing*, Cambridge University Press, 2005.
[29] W. P. Thurston, *Geometry and Topology of Three-Manifolds*, Princeton lecture notes, 1976.
[30] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Impact of radio irregularity on wireless sensor networks," *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pp 125–138, 2004.



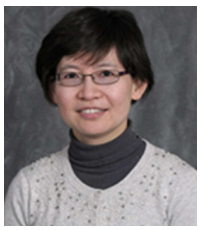
Siming Li is Ph.D candidate at Stony Brook University. Her research interests include algorithm design, wireless sensor network, and machine learning.



Wei Zeng is an assistant professor in School of Computing and Information Sciences at Florida International University, Miami, Florida. She received her Ph.D. degree in Computer Science and Technology from Chinese Academy of Sciences in 2008 and had her postdoc training at Stony Brook University during 2010-2012. Her research interests include computational conformal geometry, Ricci flow, surface registration, and shape analysis. Her research areas span over medical imaging, computer vision, computer graphics and visualization, wireless sensor network, geometric modeling and computational topology. She has published numerous papers in peer-reviewed journals and conferences and a book by Springer titled Ricci Flow for Shape Analysis and Surface Registration: Theories, Algorithms and Applications, and has two U.S. patents on virtual colonoscopy techniques.

Dengpan Zhou received the Ph.D degree from Stony Brook University, in 2012, advised by Dr. Jie Gao. His research interests include algorithm design and wireless sensor network. He is currently working in hedge fund industry.

Xianfeng Gu is an associate professor of computer science and the director of the 3D Scanning Laboratory in the Department of Computer Science, State University of New York at Stony Brook, Stony Brook, New York. He received his Ph.D. degree in Computer Science from Harvard University in 2003. He is one of the major founders of Computational Conformal Geometry. He invented computational methodologies for computing surface conformal structure based on Hodge theory and discrete surface Ricci flow. His research interests include computer vision, graphics, geometric modeling and medical imaging. His major works include global conformal surface parameterization in graphics, tracking and analysis of facial expression in vision, manifold splines in modeling, brain mapping and virtual colonoscopy in medical imaging and computational conformal geometry. He has published more than 200 papers in peer-reviewed journals and conferences, 2 books on computational conformal geometry, and has 6 U.S. patents on 3D geometric acquisition, biometrics and virtual colonoscopy techniques. He won the US National Science Foundation CAREER award in 2004.



Jie Gao is an associate professor at the Department of Computer Science, Stony Brook University. She received her Ph.D degree from the Department of Computer Science, Stanford University in 2004 and B.S. degree from the Special Class for the Gifted Young at University of Science and Technology of China in 1999. She is a recipient of the Research Excellence Award, Computer Science department, Stony Brook University, 2012; Best Paper Award from ACM SIGCOMM Internet Measurement Conference, 2009; National Science Foundation (NSF) Faculty Early Career Award, 2006; IBM Ph.D Fellowship 2003-2004. She serves on the editorial board of ACM Transactions on Sensor Networks and Journal of Discrete Algorithms.