Load Balanced Short Path Routing in Large-Scale Wireless Networks Using Area-Preserving Maps

ABSTRACT

Load balanced routing in a network, i.e., minimizing the maximum traffic load any node carries, is a well known NP-hard problem. Finding practical algorithms remains a long standing challenge. In this paper we propose greedy routing using virtual coordinates that achieves both small path stretch ratio (compared to shortest path) and small load balancing ratio (compared to optimal load balanced routing), in a large scale wireless sensor network deployed densely inside a geometric domain with complex shape. We first show that on a disk there are multiple greedy algorithm achieving constant path stretch and constant load balancing ratio. We provide one such routing scheme on the disk that has a stretch ratio of at most 2, and under which the maximum load is a factor $4\sqrt{2}$ smaller than the maximum load under shortest path routing. This is the first simple routing scheme with a small stretch that has been proven to outperform shortest path routing in terms of load balancing.

Then we transform a network of arbitrary shape to a disk by an area preserving map ϕ. We show that both the path length and the maximum traffic load in the original network only increases by an additional factor of d^2, where d is the maximum length stretch of ϕ. Combined with the result on a disk we again achieve both bounded stretch and bounded load balancing ratio. Our simulation results confirmed the superior performance over shortest path routing and prior greedy routing methods.

1. INTRODUCTION

In this paper we study the problem of greedy routing in a large scale wireless sensor networks and focus on the issue of load balancing. While reducing maximum traffic load is a general objective for most networking scenarios, in a battery-powered wireless sensor network this problem becomes more critical. Any subset of nodes used too much faces the risk of running out of battery prematurely; the functionality of the network may dramatically deteriorate even when many nodes still have ample battery life.

The maximum traffic load in a network depends on both the traffic pattern (i.e., the distribution of sources and destinations) and the topology of the network. When all messages are from the same source and/or same destination, the highest traffic is around the common 'sink'. This phenomena is termed the 'energy hole' problem and has been studied extensively in the literature [16]. In this paper we focus on the case of load balancing for general point to point traffic. This models the scenario when the sensor nodes are tightly embedded in the physical space in which human users reside. Users can query a nearby sensor and issue a message to a particular target sensor retrieving relevant information. This is also a particularly interesting scenario in theory as the topology of the network plays an important role in deciding the max load. Even when traffic pattern is uniform (source and destination chosen uniformly randomly) there can still be overly loaded nodes. In some cases the max load is mainly caused by the network topology (e.g., the nodes in a narrow neck of the network) – and there is little one can do to reduce the max load by different routing algorithms. In some other cases the max load is mainly caused by the routing algorithm (e.g., the overloaded center created by shortest path routing in a circular shaped network). This is what we focus on. We would like to design good routing scheme that achieve or approximate the best max load possible.

We observe that, in order to alleviate load accumulation, we need to spread out traffic, which leads to increased path length. Obviously using short paths is desirable for low delay. In fact, the two objectives, reducing max traffic load and reducing path lengths, have an interesting relationship. On one hand they are in agreement – using paths that are too long will elevate the total traffic load of all nodes, such that any routing algorithm has to induce heavy load on some nodes by pigeon hole principle. On the other hand, the two objectives are in conflict – if we can only use shortest paths, we have little freedom in choosing paths to spread out traffic. Indeed it has been shown that in a unit disk graph there is a trade-off between the two objectives. One cannot have both load balancing ratio and path stretch bounded by constants simultaneously, without additional constraints on the network topology [10]. This motivates us to investigate what type of network topology can support both good load balancing and short paths simultaneously and how to exploit such structural properties for a practical routing solution.

Our Approach We consider networks of wireless nodes uniformly deployed inside a complex geometric domain Ω and consider greedy routing using virtual coordinates. The case of densely deployed sensors inside a geometric domain is an important and natural scenario when sensor networks scale large. The use of greedy routing will make the solution to be practical.

In this paper we assume that the traffic pattern is modeled by a joint distribution Π of the source destination pair. We classify the network topology by the shape of Ω and ask for routing solutions with both good load balancing property and bounded stretch. Nevertheless, load balanced routing even for dense networks inside a simply shape with uniform traffic pattern, without constraints on

path stretch, is still very challenging. No good approximations (better than the general case) are known. Existing work mainly focused on simple shapes such as strips, disks or squares; or focused on reducing max load in certain parts of the network such as interior hole boundaries. We are the first to study load balancing of networks inside an irregular domain.

Throughout the paper we use the *path stretch* as the worst case ratio of the routing path length by our algorithm and the shortest path length for the same source destination pair. We use the *load balancing ratio* as the ratio of the maximum traffic load by our algorithm and the maximum traffic load of the *optimal* load balanced routing algorithm (i.e., the minimum of maximum load possible), for the same traffic pattern Π.

The main tool we use is *area-preserving maps*. Under an area-preserving map $\phi : \Omega \to \Omega'$, an ε-area ball is mapped to a piece (not necessarily round) with area ε. The main theoretical result in this paper is the following. Suppose we use a routing algorithm Γ on the original domain Ω with traffic load $\ell_\Gamma(p)$ on node p. Applying the area preserving map ϕ, we obtain a routing algorithm Γ' on the target domain Ω'. We prove that the traffic load $\ell_{\Gamma'}$ at point $\phi(p)$ is bounded by $d \cdot \ell_\Gamma(p)$ from above and $\ell_\Gamma(p)/d$ from below, in which d is the maximum length stretch of the map ϕ.

Applying the above theorem in different directions allows us to prove upper bound on the traffic load in our algorithm, as well as lower bound on the max traffic load in the optimal load balanced routing. Altogether this allows us to transfer a load balanced short path routing algorithm on Ω' to a load balanced short path routing algorithm on Ω. Specifically, if Γ' is has load balancing ratio c and path stretch of λ in domain Ω', we prove that Γ has load balancing ratio $c \cdot d^2$ and path stretch $\lambda \cdot d^2$ in Ω.

We first look at the case of a disk, which is an example of a good network topology. If we send one message between each pair of nodes in a disk shape network using shortest path routing, the maximum traffic load scales in the order of $O(n\sqrt{n})$, which is in fact already asymptotically optimal. We can further improve it by using the Lambert Azimuthal projection to map a unit disk to a sphere. By using greedy routing on the spherical coordinates we can reduce the maximum traffic load even further while only using paths of stretch at most 2. Furthermore, we prove that this routing scheme is a factor $4\sqrt{2}$ better than shortest path routingin terms of load balancing, making it the first routing scheme provably better than shortest path routing (using the definition of load in Definition 1).

For a network inside a simply connected domain Ω of an arbitrary shape, we again use area preserving maps to create virtual coordinates of good stretch and good load balancing ratio. We take a specific area preserving map ϕ from Ω to a disk and then apply the above mentioned routing algorithm in a disk. Therefore, we can compute virtual coordinate inside the disk for each node x in Ω, with which greedy routing achieves path stretch of $O(d^2(\phi))$ and load balancing ratio of $O(d^2(\phi))$. Here $d(\phi)$ is the maximum length stretch of the mapping ϕ. In particular, $d(\phi)$ gives an effective way of classifying the topology of the network. Any domain that admits an area preserving map ϕ with constant $d(\phi)$ will support greedy routing with constant path stretch and constant load balancing ratio. This provides a qualitative way to measure how the topology of the network affects its load balancing property.

We summarize our contribution below.

1. Establish connection of area-preserving maps to load balanced routing,
2. Provide a distributed, greedy routing scheme on any simply connected domain that has a worst case stretch factor of $O(d^2(\phi))$ and worst-case load balancing ratio of $O(d^2(\phi))$, where d_ϕ is the maximum length stretch of the area-preserving map from the domain to the disk.
3. Provide a routing scheme on the disk that uses routes at most twice the length of the shortest path, and is a factor $4\sqrt{2}$ better in load balancing than shortest path routing.

Note that the average load under the shortest path routing scheme is a lower bound on the maximum load under any routing scheme. Using this quantity, the load balancing ratios of both shortest path routing scheme and our routing scheme can be calculated, and they will differ by a factor $4\sqrt{2}$.

2. RELATED WORK

Load balanced routing on graphs. In the field of networking algorithms, load balanced routing on a graph with given source destination pairs is a long standing problem. One way to formulate this problem is to select routes that minimize congestion (the maximum number of messages that any node/link carries), termed the *unsplittable flow problem*. Solving this problem optimally is NP-hard even in very simple networks (such as grid). The best approximation algorithm has an approximation factor of $O(\log n/\log\log n)$ [22,23] in a network of n vertices. It is also shown that getting an approximation within factor $\Omega(\log\log n)$ is NP-hard [2]. Another popular way to formulate the problem is to consider node disjoint or edge disjoint paths that deliver the largest number of given source destination pairs. This is again NP-hard [15] and the best approximation factor known is $O(\sqrt{n})$ [6]. It is NP-hard to approximate within a factor of $\Omega(\log^{1/2-\epsilon} n)$ [7]. These approximation algorithms are mostly only of theoretical interest. They require global knowledge and are not suitable for distributed settings.

Load balancing routing on simple shapes. In [9], a greedy routing scheme that achieves both constant stretch factor and constant load balancing ratio is proposed, but only for nodes distributed in a narrow strip. In a general unit disk graph setting, there is a tradeoff between the stretch factor and load balancing ratio [10].

Popa *et al.* [21] studied load balanced routing in a disk. Under uniform traffic pattern, the center is more loaded than nodes near the boundary. By using numerical solvers, they had a numerical approximation of the optimal load balanced routing solution. They also proposed a practical algorithm by using stereographic projection to map the network on a hemisphere. Routing is guided by the spherical distance in a greedy manner. Improved load balancing is shown as the routes are made to 'curve' around the network center. But no theoretical analysis is given.

Mei and Stefa [18] studied load balancing for networks inside a square and propose to use the 'outer space' by creating four copies of the network, wrapped up as a topological torus. The destination has four images and greedy routing is done by using coordinates on the torus towards a randomly selected image of the destination. On the original network, it is as if we are reflecting on the boundaries. The idea is that through mapping on a torus the original boundary or center of the square are essentially removed so they should not present heavy load. By evaluations we found out that the traffic load on the nodes are indeed more even, but unfortunately the traffic at all nodes has been greatly elevated as routing paths through reflections are on average much longer than shortest paths.

Yu *et al.* [26] examined a network inside a simply connected domain and used Ricci flow to generate Thurston's embedding as the skeleton of a convex polytope. The intuition is to map the network on a sphere (or half sphere) as routing on a sphere has no congestion due to perfect symmetry. The choice of a convex polytope is

to ensure that greedy routing guarantees delivery [19]. This performs well in simulations but there is no theoretical guarantee on the worst case congestion.

Reducing congestion on centers or hole boundaries. A number of previous works focused on reducing traffic load on hole boundaries. This is because traditional geographical routing [8, 14] tends to send messages to nodes near hole boundaries. In [24], a network of multiple holes is converted to the covering space, such that one maps the network to the interior of each hole, filling it up. Again, since the boundaries are removed, greedy routing, when touching a boundary node, does not follow the boundary but get reflected away from the boundary. Simulation results show that the traffic load on boundaries are greatly reduced. But the average traffic load is increased as routing paths are made longer.

In [5], the focus is to reduce traffic load near the medial axis, which is a generalization of the center of a disk in a domain of general shape and is likely to attract traffic load. The proposed routing algorithm will follow a path parallel to the medial axis or orthogonal to the medial axis, minimizing the intersections with the medial axis. Again no theoretical guarantee is given.

3. THEORY OF LOAD BALANCING AND AREA-PRESERVING MAPS

3.1 Definitions

In this section we first describe load balancing in the continuous setting for which we will present our theoretical results. Throughout the paper we consider a simply connected domain Ω (i.e., no holes). We consider the traffic pattern as modeled by a joint distribution Π on the source destination pair. A routing scheme Γ describes how to find a path between two points inside Ω. For every pair of points (p, q) inside Ω, Γ specifies a path $\gamma_{p,q} \subset \Omega$ that connects p to q. The set Γ is the collection of all such paths.

In the discrete graph setting, the traffic load is taken as the number of messages delivered along each edge/through each node. In the continuous setting, however, there could be various definitions of traffic load, e.g. [11,12,20,21]. In [11] load is presented as a flux and load balancing is presented as a $\min\max$ problem. We will use the definition of load presented in [21]. The definition we present below is essentially the same, but described in a way more suited to our purposes. One should note that although these definitions are different, they nevertheless are comparable and the basic idea behind them is the same. Furthermore, the discrete definitions closely approximate the continuous ones in the case of a dense network.

Definition 1 (Load). *Given a traffic pattern Π and a routing scheme Γ for Ω, for any region $A \subset \Omega$, define the load of A (denoted as $\ell_\Gamma(A)$) under the traffic pattern Π by the following procedure:*

1. *Choose n source-destination pairs $\{(a_i, b_i)\}_{i=1}^n$ from the distribution Π. For all such pairs, find γ_i, the paths as determined by Γ.*
2. *Let $X_n(A)$ be the average length of intersection of the γ_i with A.*
3. *Define $\ell_\Gamma(A) = \lim_{n\to\infty} \frac{X_n(A)}{\text{Area}(A)}$. Here Area$(A)$ denotes the usual Euclidean area of the region A.*

Next, define the load at a point p in the domain by:

1. *Choose a nested sequence of neighborhoods A_n whose intersection is p and which satisfy Area$(A_n) \to 0$ as $n \to \infty$;*
2. *Define $\ell_\Gamma(p) = \lim_{n\to\infty} \ell_\Gamma(A_n)$.*

Clearly the traffic load depends on the specified traffic pattern Π. Throughout the paper we will fix the traffic pattern Π unless specified otherwise. Denote the maximum load over all $p \in \Omega$ under Γ as ℓ_Γ. The optimal load balanced routing problem is to find Γ^* which minimize ℓ_{Γ^*}. We define the *load balancing ratio* of a routing scheme Γ as $\ell_\Gamma/\ell_{\Gamma^*}$. For a particular routing scheme Γ, denote by $|\gamma_{pq}|$ the length of the routing path γ_{pq} from p to q and denote by $|pq|$ the shortest path length. We define the *path stretch* of Γ as

$$\max_{p,q\in\Omega} |\gamma_{pq}|/|pq|.$$

Definition 2 (Area Preserving Map). *Given two domains Ω and Ω' in $\mathbb{R}^n$ with the same area, a map ϕ from Ω to Ω' is area preserving if $\text{Area}(\phi^{-1}(A)) = \text{Area}(A)$ for every subregion A of Ω. $\text{Area}(A)$ is the area of the region A.*

The area-preserving map can be written in axis-parallel coordinates as $\phi : (x, y) \to (u, v)$. The Jacobian matrix $J(\phi)$ is a 2×2 matrix of the partials of u and v with respect to x and y. Now, ϕ is area-preserving, which is equivalent to the determinant of $J(\phi)$ being identically 1 at all points in the domain. Let $\lambda_1(x, y), \lambda_2(x, y)$ denote the two eigenvalues of $J(\phi)$ (not necessarily real). Note that $\lambda_1\lambda_2 = 1$ at every point, and hence $|\lambda_1| = 1/|\lambda_2|$.

Let $d(x, y) = \max(|\lambda_1(x, y)|, |\lambda_2(x, y)|)$ denote the maximum length distortion at point (x, y). When $p = (x, y)$, we will just write $d(p)$. Let

$$d(\phi_{\Omega,\Omega'}) = \sup_{p\in\overline{\Omega}} d(p) \tag{1}$$

$d(\phi)$ is the *maximum length stretch* of the mapping ϕ.

3.2 Bound on Max Load

Given a routing scheme Γ in Ω for the traffic pattern Π, by applying the map $\phi : \Omega \to \Omega'$ we get a routing algorithm Γ', the push-forward of the routing scheme Γ via ϕ in Ω'. $\Gamma^{'}$ allots the path $\phi(\gamma)$ to the pair (a, b), where γ is the path joining $\phi^{-1}(a)$ and $\phi^{-1}(b)$ as dictated by Γ. In particular, we can consider the traffic load of the routing scheme Γ' under traffic pattern $\Pi' = \phi(\Pi)$. In the following theorem we relate the maximum traffic load of Γ (under traffic pattern Π) and the maximum load of Γ' (under traffic pattern Π').

Theorem 3. *Given a routing scheme Γ on Ω and $\phi : \Omega \to \Omega^{'}$ area-preserving, denote by $\Gamma^{'}$ the routing scheme on $\Omega^{'}$ which allots the path $\phi(\gamma)$ to the pair (a, b), where γ is the path joining $\phi^{-1}(a)$ and $\phi^{-1}(b)$ as dictated by Γ. Then*

$$\frac{1}{d(p)}\ell_\Gamma(p) \le \ell_{\Gamma'}(\phi(p)) \le d(p)\ell_\Gamma(p) \;\forall p \in \Omega \tag{2}$$

PROOF. Fix an $\epsilon > 0$. By continuity of $J_\phi(x, y)$, $\exists\delta > 0$ such that for any $p^{'}$ in the disk of radius δ around p (denoted as $B_\delta(p)$),

$$d(p^{'}) < d(p) + \epsilon, \quad \text{and} \quad \frac{1}{d(p^{'})} > \frac{1}{d(p)} - \epsilon \tag{3}$$

Let $B_\delta(p) \supset B_{\delta_1}(p) \supset B_{\delta 2}(p) \supset \cdots$ be a sequence of nested neighborhoods of p such that $\cap_{i=1}^{\infty} B_{\delta_i}(p) = p$. We will calculate the load at $\phi(p)$ using the sequence $\phi(B_{\delta_i}(p))$. Note that $\phi(B_{\delta_i}(p))$ has the same area as $B_{\delta_i}(p)$.

Pick n source destination pairs in $\Omega^{'}$ by distribution Π'. Since ϕ is measure preserving, this amounts to picking n pairs in Ω by distribution Π and looking at their images under ϕ. Consider any

source destination path γ (joining two points a and b) intersecting $B_{\delta_i}(p)$ in Ω. The fact that ϕ is a homeomorphism implies that $\phi(\gamma)$ intersects the neighborhood $\phi(B_{\delta_i}(p))$ of $\phi(p) \in \Omega^{'}$.

Now let $\gamma_i = \gamma \cap B_{\delta_i}(p)$. As a consequence of Equation 3, the length of $\phi(\gamma_i)$, denoted as $|\phi(\gamma_i)|$ satisfies

$$\left(\frac{1}{d(p)} - \epsilon\right)|\gamma_i| < |\phi(\gamma_i)| < (d(p) + \epsilon)|\gamma_i| \quad (4)$$

as a consequence of which the load of $B_{\delta_i}(p)$ and $\phi(B_{\delta_i}(p))$ (both of which have the same area) satisfy

$$\left(\frac{1}{d(p)} - \epsilon\right)\ell_\Gamma(B_{\delta_i}(p)) < \ell_{\Gamma'}(\phi(B_{\delta_i}(p))) < (d(p)+\epsilon)\ell_\Gamma(B_{\delta_i}(p))$$

Taking limit as $i \to \infty$ gives

$$\left(\frac{1}{d(p)} - \epsilon\right)\ell_\Gamma(p) \leq \ell_{\Gamma'}\phi(p) \leq (d(p) + \epsilon)\ell_\Gamma(p) \ \ \forall p \in \Omega \quad (5)$$

Since $\epsilon > 0$ chosen above was arbitrary, this proves the theorem. □

By applying the above theorem we can now study maximum load of two algorithms in two domains of different shapes. We provide two theorems doing so, and we omit their straightforward proofs. First we study how the optimal load balanced algorithms in Ω and Ω' relate to each other. Recall that ℓ_Γ denotes the maximum load under Γ.

Theorem 4. *Let Γ^* and Ψ^* be the optimal load balanced routing schemes on domains Ω and $\Omega^{'}$ respectively. Let ϕ be an area-preserving map from Ω to $\Omega^{'}$ and define $d(\phi_{\Omega,\Omega'})$ as above. Then*

$$\frac{1}{d(\phi_{\Omega,\Omega'})}\ell_{\Gamma^*} \leq \ell_{\Psi^*} \leq d(\phi_{\Omega,\Omega'})\ell_{\Gamma^*} \quad (6)$$

Remarks on Theorem 4: Here the statement holds for *any* area-preserving homeomorphism between the two domains. Thus, we can find the area preserving map $\phi : \Omega \to \Omega'$ that achieves the minimum length stretch. Clearly the more similar Ω is to Ω' the smaller this length stretch could be.

Besides, the above statement gives non-trivial lower bounds for the best achievable load on any given domain. This has not been accomplished before. It can be intuitively understood that any routing algorithm on domains with narrow bridges/cuts necessarily creates high traffic load at the bridge area – the shape matters. This theorem makes it more precise. Take Ω' as the unit disk. We know that the minimum max load of any routing for uniform traffic pattern has maximum load of at least 0.45 – the average load of a node in the disk when one uses shortest path routing [21]. If the area-preserving mapping ϕ has length distortion at most say δ, then we know that the optimal load on the domain Ω is at least $0.45/\delta$.

The next theorem tells us how to obtain an approximate solution on the original domain Ω by using an approximate (or exact) solution on the target domain Ω'.

Theorem 5. *Let Ψ^c be a factor c approximation of the optimal solution to the* $\min\max$ *problem on $\Omega^{'}$, i.e. $\ell_{\Psi^c} \leq c\ell_{\Psi^*}$, where Ψ^* is as in Theorem 4. Let Γ be the routing scheme on Ω that allots the path $\phi^{-1}(\gamma)$ to the source-destination pair (a, b) inside Ω, where γ is the path between $\phi(a)$ and $\phi(b)$ as dictated by Ψ^c. Let Γ^* be the optimal routing scheme on Ω. Then $\ell_\Gamma \leq cd^2(\phi_{\Omega,\Omega'})\ell_{\Gamma^*}$.*

3.3 Length stretch bound

It is clear that any other routing scheme on the domain Ω will generate longer paths than shortest path routing. Now we will show that by using area-preserving map ϕ, we also bound the path stretch by a constant factor dependent on $d(\phi)$. Assume that Γ and Ψ are shortest path routing schemes on Ω and Ω' respectively, and let Ψ^c be a routing scheme on the disk such that the path it generates between a source-destination pair (u, v) is at most c times the length of the shortest path between u and v, for all pairs (u, v). We will write this as

$$|\mathcal{R}_{\Psi^c}(u, v)| \leq c|R_\Psi(u, v)|$$

here $\mathcal{R}_\Psi(u, v)$ denotes the route between u and v under Ψ and $|.|$ denotes its length. Now let ϕ be an area-preserving map from Ω to Ω', and define $d(\phi_{\Omega,\Omega'})$ as in Equation 1. Let $\Gamma^{'}$ be the pull-back of Ψ^c via ϕ.

Theorem 6. *Under the above hypothesis,*

$$|\mathcal{R}_{\Gamma'}(a, b)| \leq cd^2(\phi_{\Omega,\Omega'})|\mathcal{R}_\Gamma(a, b)| \ \ \forall a, b \in \Omega \quad (7)$$

PROOF. Let Ψ^0 be the push-forward of Γ (the shortest path routing on Ω) via ϕ. From the proof of Theorem 3, one can see that the image of a route $\mathcal{R}$ in Ω (denoted as $\phi(\mathcal{R})$) has length at most $d(\phi_\Omega)$ times that of $\mathcal{R}$. We now have the following string of inequalities:

$$\begin{aligned} |\mathcal{R}_{\Gamma'}(a, b)| &\leq d(\phi_\Omega)|\mathcal{R}_{\Psi^c}(\phi(a), \phi(b))| \\ &\leq cd(\phi_\Omega)|\mathcal{R}_\Psi(\phi(a), \phi(b))| \\ &\leq cd(\phi_\Omega)|\mathcal{R}_{\Gamma^*}(\phi(a), \phi(b))| \\ &\leq cd^2(\phi_\Omega)|\mathcal{R}_\Gamma(a, b)| \end{aligned} \quad (8)$$

where the second inequality follows by the hypothesis on Ψ^c, the third by the fact that shortest path routing Ψ will have shorter lengths than Γ^* and the last inequality follows by the same observations as the first (since Γ^* is the image of Γ, the shortest path routing on Ω). □

We will show in the simulation section that in all practical scenarios, the stretch is much smaller than the claimed constant in our proof.

4. LOAD BALANCED SHORT PATH ROUTING IN A DISK

In this section we study load balanced routing for a network inside a disk domain. We will first present the scaling laws of the maximum load for shortest path routing, under uniform traffic pattern, in a disk and on a sphere respectively.

4.1 Scaling Law in Disks, Spheres, Half Spheres

Suppose that we use uniform traffic pattern and deliver one message between each pair of nodes in the network of n nodes. When the network is uniformly distributed in a disk of radius R, the maximum traffic load by shortest path routing occurs in the center [21] and is in the order of $\Theta(n\sqrt{n})$ [13]. Notice that in fact the maximum traffic load is already *asymptotically optimal*. In particular, on average the shortest path between all pairs of nodes has length in the order of $\Theta(R) \approx \sqrt{n}$. Thus the total traffic load on all nodes is $\Theta(n^2\sqrt{n})$. By pigeon hole principle, the maximum traffic load in the network is at least $\Theta(n\sqrt{n})$. Therefore, shortest path routing in a disk achieves path stretch of 1 and load balancing ratio of $O(1)$.

For a network of n nodes uniformly deployed on a sphere of radius R, the shortest path routing follows the geodediscs on the

sphere. Due to perfect symmetry the traffic load is perfectly uniform everywhere on the sphere. Similarly, the total traffic load on all nodes is $\Theta(n^2\sqrt{n})$. Spreading the total traffic load on all nodes uniformly, the traffic load for each node is $\Theta(n\sqrt{n})$. Notice that this is also the optimal load balanced routing algorithm.

For a network of n nodes uniformly deployed inside the bottom half sphere of radius R, shortest path routing using spherical coordinates gives a maximum traffic load of $O(n\sqrt{n})$ as well. To see this, imagine $2n$ nodes uniformly spread inside a whole sphere and issue a message between all pairs of these $2n$ nodes. By the analysis above, the maximum traffic load is $\Theta(n\sqrt{n})$. Now, remove all the messages that involve any node in the top half sphere, the maximum traffic load does not increase and is thus $O(n\sqrt{n})$. The same scaling upper bound applies for other fractional caps of a sphere.

4.2 Improving Load Balancing

The discussion above suggested that for a network with a regular shape, shortest path routing already achieves asymptotically the best load balancing ratio. The improvement margin on reducing maximum load lies in improving the constant factor. This partly explains the limited improvement reported earlier by heuristic algorithms [21]. We will summarize options for greedy routing using virtual coordinates in the case of a disk. This will be used as subroutine for handling networks of general shape in the next section.

Using Lambert azimuthal projection To further reduce the center traffic load, we can map a disk to a sphere by using area-preserving maps and then adopt the spherical coordinates for greedy routing. The intuition is to push paths away from the crowded center. In particular, we use the *Lambert azimuthal equal-area projection*. Denote the (open) disk of radius r centered at the origin in the plane by $\mathbb{D}_r$, and set $\mathbb{D} := \mathbb{D}_1$. The Lambert azimuthal projection, $g : \mathbb{D}_2 \to S^2 \setminus \{(0,0,1)\}$ is the area-preserving map from the disk of radius 2 to the sphere of radius 1 (centered at the origin in $\mathbb{R}^3$) minus its north pole given by:

$$g(x,y) = \left(x\sqrt{1 - \frac{x^2+y^2}{4}}, y\sqrt{1 - \frac{x^2+y^2}{4}}, \frac{x^2+y^2}{2} - 1 \right) \tag{9}$$

One can see that g maps the disk of radius $\sqrt{2}$ to the lower hemisphere, the circle of radius $\sqrt{2}$ to the equator, and the remainder of the bigger disk with radius 2 to the upper hemisphere. Let $\mathcal{S}_H$ denote the lower half sphere, and Γ denote the greedy routing using spherical metric on it. For two nodes $a, b \in \mathcal{D}_{\sqrt{2}}$, we choose the path $g^{-1}(\mathcal{R}_{g(a),g(b)})$, where $\mathcal{R}_{g(a),g(b)}$ is the shortest path between $g(a)$ and $g(b)$ on $\mathcal{S}_H$ using Γ. Denote this routing scheme on $\mathcal{D}_{\sqrt{2}}$ as Ψ. Furthermore, let Δ denote shortest path routing on the disk. We can show that Ψ has maximum load a factor $4\sqrt{2}$ times smaller than the maximum load of Δ, and has path stretch at most two.

Theorem 7. *Let Ψ and Δ be two routing schemes on the disk of radius $\sqrt{2}$ as above. Then $\ell_\Psi = \ell_\Delta/(4\sqrt{2})$.*

PROOF. First, we need to calculate the proportionality constant in Theorem 1 in [21]. Redoing their integration, one finds that the maximum load under Δ is $4\sqrt{2}/\pi^2$.

Now we need to find the load on the south pole (of the unit lower half sphere), under the shortest path routing on the spherical metric. We set up an integral similar to the disk case, using a similar coordinate change. Thus the load on a small disk of radius ϵ at the south pole is found out to be $\frac{I(\epsilon)}{\pi\epsilon^2}$, where $I(\epsilon)$ is the integral mentioned. This integral does not have a closed-form expression.

We then use Taylor series to find out the load at the south pole as

$$\ell_\Psi = \frac{1}{2\pi} . \frac{\partial^2 I}{\partial \epsilon^2} \|_{\epsilon=0}$$

which turns out to be $1/\pi^2$. Comparing with the load of $4\sqrt{2}/\pi^2$ gives us the desired constant. □

Theorem 8. *For any source destination pair (a, b) inside $\mathcal{D}$,*

$$|\mathcal{R}_\Psi(a,b)| \leq 2|\mathcal{R}_\Delta(a,b)|.$$

PROOF. We first find out how much a curve on the disk gets streched when mapped to the lower half sphere via the area-preserving mapping. To do this, we write the Lambert's azimuthal mapping in term of polar coordinates (R, Θ) on the disk and cylindrical coordinates (r, θ, z) on the sphere:

$$(r, \theta, z) = \left(R\sqrt{1 - \frac{R^2}{4}}, \Theta, \frac{R^2}{2} - 1 \right) \tag{10}$$

Given a curve $(R(t), \Theta(t))$ on the disk we find its tangent vector v by differentiating equation 10. Calculating the length of v and normalizing $(R(t), \Theta(t))$ by requiring $\dot{R}^2 + R^2\dot{\Theta}^2 = 1$, we maximize the length keeping in mind that R ranges from 0 to $\sqrt{2}$ and $\dot{R}$ from 0 to 1.

This implies that any curve on the disk when translated to the sphere gets stretched by a factor of at most $\sqrt{2}$. The same statement applies if we interchange the roles of the disk and the sphere. Now consider any two points a and b on the disk, with shortest path $\gamma(a, b)$ of length $\ell(a, b)$. Now $|g(\gamma)| \leq \sqrt{2}\ell(a, b)$ by the above observation on the stretch, and the shortest path $\tilde{\gamma}$ on the lower half sphere from a to b certainly has a length shorter than that of $g(\gamma)$. By the argument for the inverse of the mapping, the pull-back of $\tilde{\gamma}$ on the disk is again stretched by a factor at most $\sqrt{2}$, and this gives us the combined factor of 2 claimed. □

Moreover, it is clear that just by scaling the original network to radius R disk, one can arrange for the map to cover different parts of the sphere. Thus we scale our disk to a radius close to 2 if we want to push routes more towards the boundary and closer to $\sqrt{2}$ if we want them pushed lesser. The former is more aggressive in reducing the center load. In our simulations we provide different values of R for which we apply this map.

Curveball – using stereographic projection Curveball routing uses the same intuition of mapping a disk to a sphere, except that the projection used is the stereographic projection. Stereographic projection preserves angles, not areas. We refer the reader to [21] for details. Again, routing is done greedily using spherical metric on these virtual coordinates. Curveball routing is a heuristic. It can be shown that if the disk is mapped to the bottom halfsphere the length stretch is a constant. But there is no proof on how much the maximum load is reduced compared to that of shortest path routing.

Using approximate optimal An approximate optimal solution to the load balancing problem on the disk was described in [21]. However, since optimal is complicated, one cannot hope to get closed formula for the route from a source to a destination. We refer the reader to Section 3.4 of [21] for details. In our implementation, we weight each edge by its distance r to the center of the disk using the formula $1.8r^3 - 3.1r^2 + 2.3$. We compute the shortest path in the weighted graph and compare the performance with our methods. Notice that this algorithm is not a greedy algorithm and is less practical than the two algorithms above.

5. LOAD BALANCED SHORT PATH ROUTING ON AN ARBITRARY DOMAIN

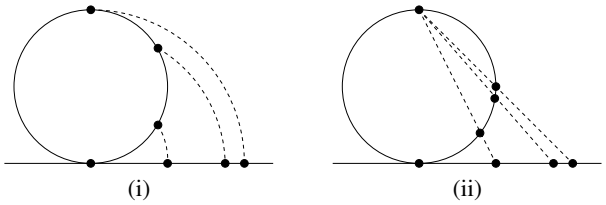


Figure 1. A cross sectional view of the sphere and a plane tangent to it at south pole. (i) Area-preserving map: each point on the sphere (except the north pole) is projected to the plane along a circular arc centered at the point of tangency between the sphere and plane. (ii) Stereographic map: a point p on the plane is mapped to the intersection of the line through p and the north pole with the sphere.

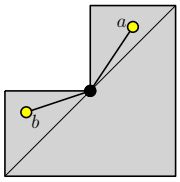


Figure 2. An L-shape domain. The shortest path from the top corner to the left corner will go through the nodes near the reflex vertex. The maximum traffic load near the reflex vertex is about $\Theta(n^2)$.

5.1 Area-Preserving Map to Disk

For an arbitrary domain Ω we again use area-preserving maps. This time we map Ω to a disk such that we can pull back our solution from the previous section. By applying our theory in Section 3 we obtain the following results immediately.

Theorem 9. *For any simply connected domain Ω and an area preserving map ϕ that maps Ω to a disk of the same area, we use Lambert's azimuthal mapping to map the disk to a half sphere and create the virtual coordinates of all nodes in the network as the spherical coordinates on the half sphere. Greedy routing using such spherical virtual coordinates has path stretch of $4d_\phi^2$ and load balancing ratio of d_ϕ^2, where d_ϕ is the maximum length stretch of ϕ.*

We would like to use an example to demonstrate the potential of this approach. Take a look at the L-shape domain with n nodes uniformly spread inside. With a message between every pair of nodes, the nodes near the reflex vertex of the domain have traffic load of $\Theta(n^2)$ by shortest path routing – there are quadratic many nodes whose shortest paths 'bend' at the reflex vertex. See Figure 2. Using the algorithm to be presented below, we can find an area preserving map that maps the L-shape domain to a disk with constant length stretch. Thus we can reduce the maximum load to be $\Theta(n\sqrt{n})$, by only increasing the path stretch by a small constant! In our simulation section (see Figure 7) we report the load reduction and its scaling in network size.

In the following we will present an distributed algorithm to compute an area preserving map.

5.2 Computing Area Preserving Maps

For some domains (e.g. the square) a closed form for an area-preserving map to the disk is available. Generally, this is not always possible and numerical methods are required. We will present distributed algorithms for computing such a map.

We start with a simply connected domain (with finite area) whose boundary is a piece-wise smooth curve. For our purposes, we can assume that the boundary is a polygon with k vertices. This is reasonable because any smooth boundary can be approximated by polygons. Note that k is often much smaller than the number of sensors, n, inside the domain.

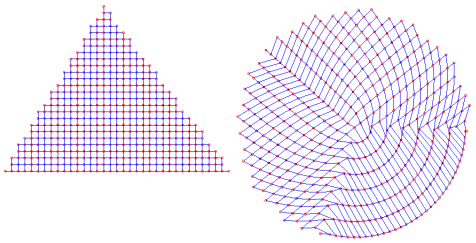
Let Ω denote the interior of such a polygonal region, with $\mathcal{P}$ as the boundary polygon. Our goal is to map Ω to a disk D centered at the origin (of area equal to that of Ω) in an area-preserving way. Consider a real valued function f defined on Ω in such a way that the level sets of f are simple closed curves that fill up Ω. We call f the *contour generating function*. Given f, we will first define an area-preserving mapping from Ω to the disk in terms of f. We will then show how to obtain f for any given Ω in a simple manner.

To describe our bounds, we need to consider the Jacobian of our mapping. Set $u := R\cos\Theta$ and $v := R\sin\Theta$ where R and Θ are the same as in Equation 13, but now we use the conformal mapping g and the contour function f described in the previous section. The new equations for R and Θ now become:

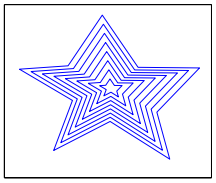
$$\begin{aligned} R(x,y) &= \tilde{f}(x,y) := \sqrt{\text{Area (Int }\{f^{-1}(f(x,y))\})/\pi} \\ \Theta(x,y) &= \frac{1}{\tilde{f}(x,y)} \int_{g^{-1}(f(x,y))}^{x,y} \frac{ds}{(\tilde{f}_x^2 + \tilde{f}_y^2)^{1/2}} \end{aligned} \tag{11}$$

5.2.1 *Area-Preserving map using the contour generating function*

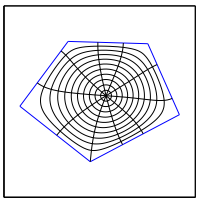
Assume that f has continuous first order partial derivatives everywhere except at $\mathcal{O}$, and not both of the partials are zero at any point. Let $\mathcal{C}$ be the family of contours of f, i.e. $\mathcal{C}$ is a set containing all simple closed curves $\gamma \subset \Omega$, such that there exists a constant c_γ with $f(x,y) = c_\gamma$ for all points $(x,y) \in \gamma$. Let $\mathcal{O}$ be a point interior to all the curves in $\mathcal{C}$, and let $\mathcal{L}$ be a path joining $\mathcal{O}$ to some point on $\mathcal{P}$ such that $\mathcal{L}$ intersects each curve in $\mathcal{C}$ once. See examples of contoured regions in Figure 3(b) and Figure 3(c).



(a) An example of area-preserving map from a triangle to a disk



(b) Contoured star-shaped polygon using boundary method



(c) Contoured pentagon using conformal mapping method

Figure 3. Example of area-preserving map and contoured polygons

The area-preserving mapping $\phi : \Omega \to \mathcal{D}$ we use was first described in [4]. It has the following properties:

1. Every curve $\gamma \in \mathcal{C}$ is mapped to a circle inside $\mathcal{D}$ centered at the origin. Hence the contours are mapped to concentric circles. See Figure 3(a) for an example.

2. The path $\mathcal{L}$ is mapped to any given radius of $\mathcal{D}$.

To construct ϕ, we first modify our f as follows. Define a function $\tilde{f}$ on Ω such that $[\tilde{f}(x,y)]^2$ at any point $(x,y) \in \Omega$ equals $1/\pi$ times the area inside the contour in $\mathcal{C}$ passing through (x,y), i.e.

$$\tilde{f}(x,y) = \sqrt{\text{Area}(\text{ Int } \gamma_{(x,y)})/\pi}, \tag{12}$$

where $\gamma_{(x,y)} \in \mathcal{C}$ is the contour passing through (x,y). Choose polar coordinates (R,Θ) in $\mathcal{D}$ where Θ is measured from the given radius to which we want to map $\mathcal{L}$. The map $\phi : (x,y) \to (R,\Theta)$ is now given by :

$$R(x,y) = \tilde{f}(x,y) \ , \ \Theta(x,y) = \frac{1}{\tilde{f}(x,y)} \int_{Y(x,y)}^{x,y} \frac{ds}{(\tilde{f}_x^2 + \tilde{f}_y^2)^{1/2}}, \tag{13}$$

where $Y(x,y)$ is the point of intersection of $\mathcal{L}$ with the curve in $\mathcal{C}$ through (x,y), $\tilde{f}_x$ and $\tilde{f}_y$ are the partial derivatives, and the integral is evaluated along the curve of $\mathcal{C}$ from $Y(x,y)$ to (x,y) in the direction for which the interior is on the left.

For the proof that this map is indeed area-preserving, we refer the reader to [4]. Observe that any contour γ passing through (x,y) gets mapped to a circle of radius $\tilde{f}(x,y)$ (see Figure 3(a) for an example), and by the definition of $\tilde{f}$, they have the same area. In general, there are many area-preserving mappings from Ω to $\mathcal{D}$; we choose the above one[1] because of its nice property of mapping contours to circles. Furthermore, we will see later that by choosing $\mathcal{O}$ to be the point with the maximum load, we can achieve good results for load balancing.

Now we describe how to map Ω to $\mathcal{D}$ (the disk of area equal to that of Ω) in a distributed manner. For ease of description we assume first that $\mathcal{D}$ is a unit disk; the method described below can easily be generalized for a larger disk. Recall that the nodes on the same contour (the same value under f) will be mapped to the same circle in $\mathcal{D}$. Thus there are two steps for computing the map. First we need to form the level set of f in a distributed way. Second we will compute the virtual coordinate for each node under the area-preserving map.

There has been previous work on finding contours, one of such robust algorithms in a distributed setting is to use the *cut locus* [25, 27]. In fact, in this setting the level sets are simple – f has only a single minimum inside Ω and no saddles. We discretize the interval $[0,1]$ into $1/\varepsilon$ intervals of width ε each. Here ε depends on the density of the nodes in the network and is in the order of $O(\pi/n)$. We now find one contour cycle γ_i for each interval $[i\varepsilon,(i+1)\varepsilon]$, $0 \le i \le n/\pi$. The set of nodes whose values are inside $[i\varepsilon,(i+1)\varepsilon]$, denoted as C_i, naturally occupies an annulus. Note that we will choose ε according to the network density such that C_i is connected. One node, denoted as the root of this contour, r_i floods within C_i. For the purpose of the computations later, we will choose the roots r_i for different levels along a path that maps to a radius under the conformal map g. That is, the nodes whose projection under g on the positive x-axis (or is the closest to the x-axis among neighbors) will be chosen as the roots for the contours. After the flooding from r_i in C_i, the cut locus is defined as a pair of neighboring nodes whose shortest paths to the root are different and far apart. Then connecting the shortest path for the cut pair will give us a closed cycle $\gamma_i = \{z_{ij}\}$ representing the contour at the range $[i\varepsilon,(i+1)\varepsilon]$. The nodes within the band that are not selected to be on the contour cycle will be rounded to the nearest node on the contour cycle and will be handled later. Remark that the flooding is only restricted to the nodes inside the annulus and flooding for different levels does not overlap. Thus the total communication cost is linear in the number of nodes.

At the end of this procedure, we have a closed contour γ_i with a root r_i. Now we can circulate another message along γ_i to calculate the area inside γ_i. Then r_i forms

$$\tilde{f}_i = \sqrt{\text{Area}(\gamma_i)/\pi}$$

and sends this value to all nodes in the contour γ_i. This gives the radius of the circle that γ_i is mapped to. In polar coordinates (R,Θ), R for node z_{ij} lying on contour γ_i is just $\tilde{f}_i$, i.e $R(z_{ij}) = \tilde{f}_i$, $\forall j$. Now we will find the angular coordinate for each node of γ_i by computing a (discrete) integral along γ_i, which approximates the integral in Equation 11.

First we need to calculate the partial derivatives. For every node z_{ij} on the contour γ_i:

1. Locates neighbors x_{ij}^+, x_{ij}^-, y_{ij}^+, and y_{ij}^+, where x_{ij}^+ (y_{ij}^+) is the closest neighbor in the positive direction of the horizontal (vertical) line passing through z_{ij}, x_{ij}^- (y_{ij}^-) is the closest neighbor in the negative direction of the horizontal (vertical) line passing through z_{ij}. Find their values under $\tilde{f}$ through local communications.
2. Computes

$$v_x(z_{ij}) = \frac{\tilde{f}(x_{ij}^+) - \tilde{f}(x_{ij}^-)}{\text{distance}(x_{ij}^+, x_{ij}^-)}, v_y(z_{ij}) = \frac{\tilde{f}(y_{ij}^+) - \tilde{f}(y_{ij}^-)}{\text{distance}(y_{ij}^+, y_{ij}^-)}$$

and $v_{ij} = 1/(\sqrt{(v_x(z_{ij}))^2 + (v_y(z_{ij}))^2})$.

For computing $\Theta(z_{ij})$, we start from the root of the contour $r_i = z_{i0}$, for which $\Theta(r_i) = 0$. Now we just discretize the integral in Equation 11. Let $V_{i0} = 0$. A node z_{ij} gets a value $V_{i(j-1)}$ from node $z_{i(j-1)}$. It then adds the quantity $v_{ij}(\text{distance}(z_{ij}, z_{i(j-1)}))$ to this value and passes it as V_{ij} to node $z_{i(j+1)}$. In other words, the sum

$$V_{ij} = \sum_{k=0}^{j} v_{ij}(\text{distance}(z_{ij}, z_{i(j-1)}))$$

is updated by node z_{ij}. We finally define $\Theta(z_{ij}) = (V_{ij})/(\tilde{f}_i)$. In this way all the nodes on contours find their positions inside the disk in polar coordinates. Setting $u = R\cos\Theta$ and $v = R\sin\Theta$ gets them the cartesian coordinates. Denote this map as $\phi : (x,y) \to (u,v)$.

For nodes that are not on any contour in the above method, a simple "interpolation" function can be used.

5.2.2 *Finding the contour generating function*

The previous section assumed the knowledge of f—the contour generating function. In this section we describe how we generate contours for an arbitrary domain.

For many shapes, the contours can be generated easily just by shrinking the boundary by appropriate factors. For instance, consider the case of star-shaped polygons $\mathcal{P}$; one can find the center of this polygon p (the point from which the entire polygon is "visible") and find the distance of every point x on the boundary to p. Let L_{px} denote the line segment between p and x and ℓ_{px} denote its length. Define the ith contour γ_i by

$$\gamma_i = \{q_x \in L_{px} : \text{distance}(q_x, p) = \epsilon_i \ell_{px}; x \in \mathcal{P}\}$$

where the $\epsilon_i \in (0,1)$ are constants. Figure 3(b) shows an example of this. Note that the contours attained in this way are non-differentiable, i.e. they will have corners.

We describe next one elegant and simple method to get a smooth contour generating function for any domain Ω. Let $g : \Omega \to \mathbb{D}$

[1]To the best of our knowledge, the map described in [4] has not been used before for load-balancing purposes.

be the *conformal mapping* from the polygon to the disk. Define $f : \Omega \to [0,1]$ by $f(x,y) = |g(x,y)|$ where $|g(x,y)|$ denotes the distance of $g(x,y)$ from the origin. It is now clear that $f^{-1}(a)$ for some $a \in [0,1]$ will be a simple closed curve γ inside Ω. Furthermore, the pre-image of the interval $[0,1]$ can be seen to be a curve joining $\mathcal{O}$ to $g^{-1}(1)$ which intersects every contour only once. We define $\mathcal{L} := g^{-1}([0,1])$ and then use the mapping described in the previous section. The second figure in Figure 3(c) illustrates this method for a pentagon. As mentioned, the contours are smoothed out. Care should be taken when deciding on which point inside the polygon is to be mapped to the origin on the unit disk; essentially the point should not be too close to the boundary of the polygon, since then the contours would no longer be uniformly dense.

In a distributed setting, every node z wants to find its image in the unit disk under the conformal mapping $g(z)$. The map g depends only on the polygon $\mathcal{P}$ - boundary of the domain Ω. Another freedom in the Schwarz-Christoffel mapping is that we can choose which node in Ω to be mapped to the origin of disk. For our purposes, we feel it is reasonable to map the node with expected high traffic load (e.g., the centroid of $\mathcal{P}$) under the shortest path routing scheme to the origin. This is because the method that we use to alleviate load in the disk assume the center to be overloaded and then try to route away from it. However, this node should not be very close to the boundary as stated earlier, for so the contours would not be uniformly dense; if this happens, we choose an arbitrary node closer to this node, but is farther from the boundary.

With all this information, a single node (e.g., the base station) computes the parameters of the conformal mapping. Only the information about the parameters in the Schwarz-Christoffel mapping g is relayed to all nodes via flooding. Using the functional form of g, the nodes individually compute their own pre-images and compose with e^{-1} as above to find their coordinates in the disk. Once every sensor node with original coordinate z has found $g(z)$ (its image under the conformal map to the disk), it calculates $f(z)$ which is just the distance of $g(z)$ from the origin. $f(z)$ is the contour generating function. In this method, the information relayed is only $O(k)$, where $k \ll n$ is the number of vertices of the polygon and is much smaller than n, the number of sensors in the domain. The contour function for the shrinking method for star-shaped polygons is just the distance of every node to the center of the star, which can be computed by each node individually by routing to the center node.

5.3 Analysis of the Algorithm

Computational Complexity and Memory Requirements The distributed algorithm provided above is very light-weight in terms of both computational complexity and memory requirements. The parameters of the Schwarz-Christoffel conformal mapping can be computed to ϵ accuracy in $O(k \log \frac{1}{\epsilon})$ computations [3], where k is the number of nodes on the boundary of the domain. Only one node, the base station has to undergo this cost. All the other nodes undergo a subsequent $\Theta(1)$ cost to compute the area-preserving map. Routing on the spherical metric in virtual coordinates also requires $\Theta(1)$ computations per node on the routing path.

The same holds for the memory requirements. The base station has to store the $\Theta(k)$ coordinates of the vertices on the boundary. After the conformal map is computed, all nodes require $\Theta(1)$ memory for storing quantities aiding in the computation of the area-preserving map. Once the virtual corodinates on the disk (and hence on the sphere) are obtained, a node only stores its virtual coordinate and that of its neighbors.

Guaranteeing Delivery For an arbitrary domain, greedy routing can get stuck at a local minima. However, because our coordinates are either in the disk or on the sphere (both convex shapes), for a dense network it is very unlikely that greedy routing gets stuck. Moreover, if the density is not high enough and greedy routing does get stuck, we can still ensure delivery by using the contours. The node at which the packet is stuck can always use the "contour" routes to route along its contour until it finds a node that can continue greedy routing. Although this might change the load function, the effect would not be drastic as 1) Such instances would be rare due to convexity of the target domain in which the virtual coordinates lie and 2) The expected detour would be very small. In the next section, we always compare our newly developed methods to shortest path routing on the original domain Ω because greedy routing can get stuck, in which case packets are not delivered, and our conclusions about the change in load function would be erroneous.

6. SIMULATIONS

We perform experiments[2] on several networks in domains of different shapes, including the disk. In our disk model, we place about $1500 - 2000$ nodes on a perturbed grid with an average degree of 7.3, and perform all-pairs shortest path to get the load for each node. We not only consider the unit disk graph model but also the more realistic quasi-unit disk graph model. For a dense network in an arbitrary simply connected domain, we first generate contours and then using the algorithm mentioned before to map the domain to the disk in an area-preserving way.

We use different routing schemes on the disk - using the Lambert's azimuthal mapping (and then routing greedily on the spherical coordinates), Curveball Routing [21], the optimal approximation [21] and shortest path routing. For a domain of arbitrary shape, we give it virtual coordinates in the disk using the area-preserving mapping and then implement all the aforementioned routing schemes. We then compare them with the shortest path routing on the original domain and also with each other. We can summarize our observations as follows:

1. For a domain of arbitrary shape (except the disk), using almost any of the above routing schemes on the disk after applying the area-preserving map gives considerably better results than shortest path routing on the original domain. This improvement occurs for a variety of domains (different shapes).
2. For the disk the optimum approximation (presented in [21]) gives very high load at the boundary - a problem which is resolved by both our application of the Lambert's azimuthal mapping and Curveball Routing. The azimuthal mapping technique gives results very similar to those of Curveball. The maximum stretch of the Lambert's azimuthal mapping is less than 1.9 (note that Theorem 8 gave a theoretical upper bound of 4).
3. The above statements also hold for a quasi-unit disk graph with reasonable degree variance and link asymmetry. Thus our method is still very effective when all nodes do not necessarily have the same radius of communication, and when links are asymmetric.
4. When we decrease the average degree, the difference between our method and shortest path routing begins to diminish, and at average degree around 5 they are very similar. This is because in this case, there are not many paths apart from the shortest one, and our algorithm is forced to use it most of the time.

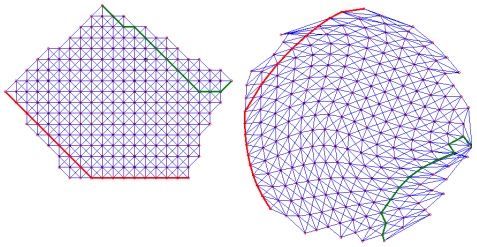
[2]All our experiments were performed on the $C\#$ Simulator, and for the conformal mapping we use the Schwarz-Christoffel toolbox [1] developed for MATLAB [17]

We provide an example of the area-preserving mapping from a pentagon with 238 nodes and average degree 7.24 to the disk in Figure 4(a). After getting coordinates on the disk, the comparison of various routing schemes on virtual and original coordinates (in the pentagon) is provided in Figure 4(b). The maximum load of shortest path routing decreased by about 19% when we use the area-preserving map to the disk and Curveball routing thereafter, and by similar percentages for area-preserving map followed by the azimuthal mapping. Note that this improvement comes only at the cost of maximum 4% increase in the average load (which is minimized by shortest path routing for obvious reasons). The optimum approximation in [21] still suffers from abnormally high load at the boundary, but performs similarly otherwise.

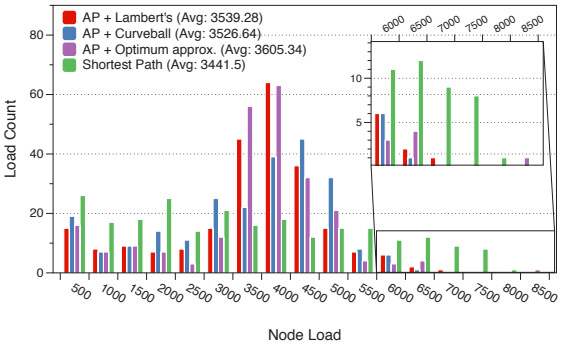
Fig. 5 presents the other result of the load balancing performance on a cross shape domain. With Lambert's mapping, the maximum load of the domain is reduced by 18%; while Curveball and optimum approximation achieve even better, up to 30%. This clearly shows that area-preserving mapping to a unit disk drastically reduced the load. Note that for "fat" and convex domains, the reduction in load was even more prominent, which is to be expected. We achieve similar figures for many complicated shapes (like the star); for simplicity we just presented a generic case of the cross shaped domain.

We also test the routing stretch in all domains. The maximum routing stretch factor was 1.96, which we believe is reasonable, considering the reduction in load.

The comparison of different routing schemes on the unit-disk graph with about 1500 nodes on a perturbed grid is provided in Figure 6, which corroborates observation (2) above. As mentioned, the above results were stable—for domains with different shapes and for the quasi-unit disk graph models we achieved similar performances.



(a) A pentagon with grid nodes and area-preserving map to disk using contours by conformal mapping method (238 nodes, avg degree 7.24)



(b) A comparison of the routing schemes. Note the decrease in maximum load on using area-preserving map followed by either azimuthal or stereographic mapping to sphere compared to shortest path routing

Figure 4. Load-balanced routing on a pentagon.

7. CONCLUSION

A clear open problem is to consider non-simple domains, i.e., domains with holes. Previous work mainly focused on how to alleviate the heavy traffic along hole boundaries. We remark that bounded load balancing would become much more challenging as one has to also consider paths of different homotopy types (getting around holes in different ways). In particular, an extreme case of the problem on a multi-connected domain could be load balanced routing on a planar graph, which is known to be NP-hard. New ideas are needed to solve that case and this remains one of the most interesting future directions.

8. REFERENCES

[1] *Schwarz-Christoffel toolbox for MATLAB*. `http://www.math.udel.edu/~driscoll/SC/`.

[2] M. Andrews and L. Zhang. Hardness of the undirected congestion minimization problem. *SIAM Journal on Computing*, 37(1):112–131, 2007.

[3] C. Bishop. Conformal mapping in linear time. *Discrete and Comput. Geometry*, 44(2):330–428, 2010.

[4] A. B. Brown and M. Halperin. On certain area-preserving maps. *Annals of Mathematics*, 36(4):833–837, Oct. 1935.

[5] J. Bruck, J. Gao, and A. Jiang. MAP: Medial axis based geometric routing in sensor networks. *Wireless Networks*, 13(6):835–853, 2007.

[6] C. Chekuri, S. Khanna, and F. B. Shepherd. An $o(\sqrt{n})$-approximation for EDP in undirected graphs and directed acyclic graphs. *Theory of Computing*, 2:137–146, 2006.

[7] J. Chuzhoy and S. Khanna. New hardness results for undirected edge-disjoint paths. Manuscript, 2005.

[8] Q. Fang, J. Gao, and L. Guibas. Locating and bypassing routing holes in sensor networks. In *Mobile Networks and Applications*, volume 11, pages 187–200, 2006.

[9] J. Gao and L. Zhang. Load balanced short path routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems, Special Issue on Localized Communications*, 17(4):377–388, April 2006.

[10] J. Gao and L. Zhang. Tradeoffs between stretch factor and load balancing ratio in routing on growth restricted graphs. *IEEE Transactions on Parallel and Distributed Computing*, 20(2):171–179, February 2009.

[11] E. Hyytiä and J. Virtamo. On load balancing in a dense wireless multihop network. In *2nd conference on Next Generation Design and Engineering (NGI), pp. 72-79, Valencia,Spain*, 2006.

[12] E. Hyytiä and J. Virtamo. Near optimal load balancing in dense wireless multi-jop networks. In *4th conference on Next Generation Design and Engineering (NGI). pp. 181-188, Karakow, Poland*, 2008.

[13] E. A. Jonckheere, M. Lou, F. Bonahon, and Y. Baryshnikov. Euclidean versus hyperbolic congestion in idealized versus experimental networks. *Internet Mathematics*, 7(1):1–27, 2011.

[14] 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.

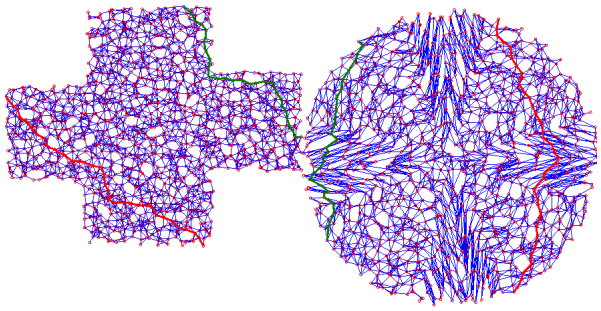
[15] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[16] J. Li and P. Mohapatra. Analytical modeling and mitigation techniques for the energy hole problem in sensor networks. *Pervasive and Mobile Computing*, 3(3):233–254, 2007.

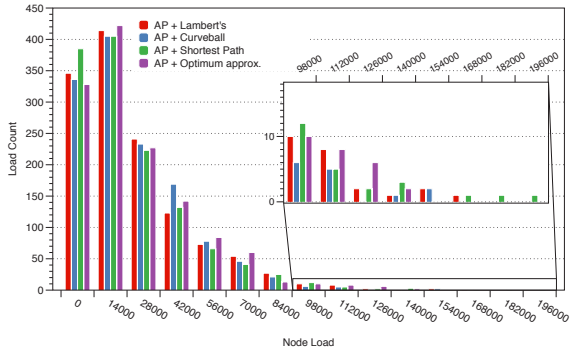
[17] MATLAB. version 7.10.0 (r2010a). *The MathWorks Inc.,Natick, Massachusetts*, 2010.

[18] A. Mei and J. Stefa. Routing in outer space: fair traffic load in multi-hop wireless networks. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 23–32, New York, NY, USA, 2008. ACM.

[19] C. H. Papadimitriou and D. Ratajczak. On a conjecture related to

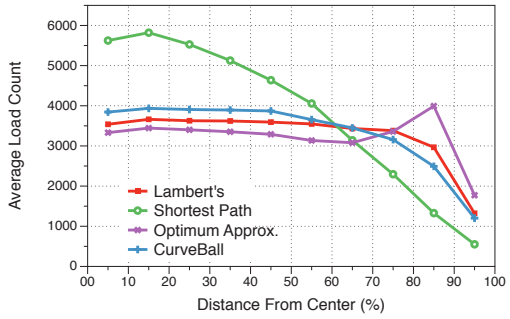


(a) Cross-shaped domain with 1302 perturbed grid nodes, avg. degree 7.38

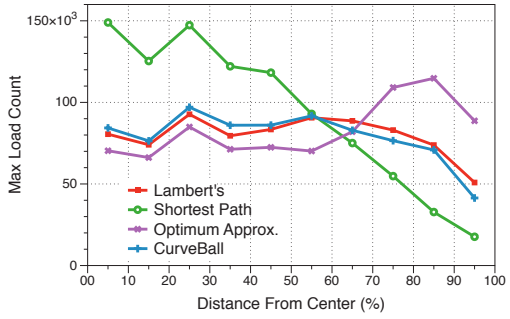


(b) The histogram comparison of load distribution over the cross-shaped domain. We first map the cross shape domain to the unit disk, then compare load balancing algorithm on the disk. Notice that all unit disk load balancing routing algorithms now perform better than shortest path routing over cross shape domain.

Figure 5. Load-balanced routing on a cross-shape domain.



(a) Histogram of the average load for a unit disk network, as a function of distance from the center.



(b) Histogram of the maximum load for a unit disk network.

Figure 6. Histogram of load distribution for a unit disk network

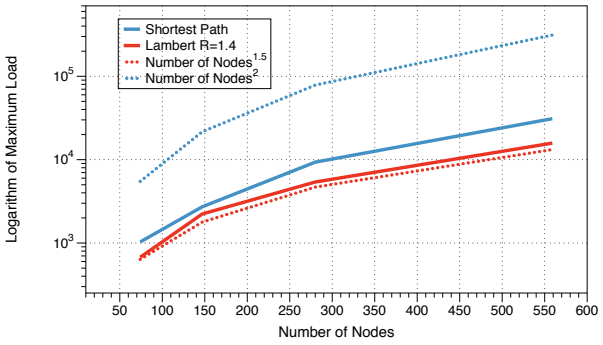


Figure 7. We tested the L-shape network for different network size. In particular we try shortest path routing on the original network as well as the Lambert based greedy routing approach. We calculate the highest traffic load by both routing algorithm. By our analysis, the first scales in the order of $O(n^2)$ and the second scales in the order of $O(n\sqrt{n})$. The figure shows the logarithm of the maximum load respectively. It is clear to see that the curve for shortest path routing has a bigger slope.

geometric routing. *Theor. Comput. Sci.*, 344(1):3–14, 2005.

[20] P. P. Pham and S. Perreau. Performance analysis of reactive shortest path and multipath routing mechanism with load balance. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE, Vol. 1 (2003), pp. 251-259 vol.1*, 2003.

[21] L. Popa, A. Rostamizadeh, R. Karp, C. Papadimitriou, and I. Stoica. Balancing traffic load in wireless networks with curveball routing. In *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 170–179, New York, NY, USA, 2007. ACM.

[22] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *J. Comp. and System Sciences*, pages 130–143, 1988.

[23] P. Raghavan and C. D. Thompson. Provably good routing in graphs: regular arrays. In *Proceedings of the 17th annual ACM Symposium on Theory of Computing*, pages 79–87, 1985.

[24] 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.

[25] Y. Wang, J. Gao, and J. S. B. Mitchell. Boundary recognition in sensor networks by topological methods. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 122–133, September 2006.

[26] X. Yu, X. Ban, R. Sarkar, W. Zeng, X. D. Gu, and J. Gao. Spherical representation and polyhedron routing for load balancing in wireless sensor networks. In *Proc. of 30th Annual IEEE Conference on Computer Communications (INFOCOM'11)*, pages 612–615, April 2011.

[27] X. Zhu, R. Sarkar, J. Gao, and J. S. B. Mitchell. Light-weight contour tracking in wireless sensor networks. In *Proceedings of the 27th Annual IEEE Conference on Computer Communications (INFOCOM'08)*, pages 960–967, May 2008.