Face Tracing Based Geographic Routing in Nonplanar Wireless Networks

Fenghui Zhang, Hao Li, Anxiao (Andrew) Jiang, Jianer Chen, and Ping Luo
Dept. of Computer Science, Texas A&M Univ. College Station, TX 77843
Email: {fhzhang, hao, ajiang, chen, pingluo}@cs.tamu.edu

***Abstract*— Scalable and efficient routing is a main challenge in the deployment of large ad hoc wireless networks. An essential element of practical routing protocols is their accommodation of realistic network topologies. In this paper, we study geographic routing in general large wireless networks. Geographic routing is a celebrated idea that uses the locations of nodes to effectively support routing. However, to guarantee delivery, recent geographic routing algorithms usually resort to *perimeter routing*, which requires the removal of communication links to get a planar sub-network on which perimeter routing is performed. Localized network planarization requires the wireless network to be a unit-disk graph (UDG) or its close approximation. For networks that significantly deviate from the UDG model, a common case in practice, substantially more expensive and non-localized network planarization methods have to be used. How to make such methods efficiently adaptable to network dynamics, and how to avoid the removal of an excessive number of links that leads to lowered routing performance, are still open problems. To enable efficient geographic routing in general wireless networks, we present *face-tracing based routing*, a novel approach that routes the message in the *faces* of the network that are virtually embedded in a *topological surface*. Such faces are easily recognizable and constructible, and adaptively capture the important geometric features in wireless networks — in particular, holes, — thus leading to very efficient routing. We show by both analysis and simulations that the face-tracing based routing is a highly scalable routing protocol that generates short routes, incurs low overhead, adapts quickly to network dynamics, and is very robust to variations in network models.**

I. Introduction

It is a challenging task to design practical routing schemes for large-scale *ad hoc* wireless networks (e.g., sensor networks). Limited energy and memory are often bottlenecks for such networks. And the complexity of connectivity and topology is key to the design of the routing protocols.

To support efficient and scalable routing, geographic routing has been extensively explored in recent years as a major technique. Geographic routing uses *greedy forwarding*: a relay node greedily forwards the message to a neighbor that is closer to the destination in Euclidean distance [3], [9], [12].[1] Such a step utilizes the close relation between a large-scale wireless network's topology and its deployment field, and greatly simplifies the design of the routing algorithm. Greedy forwarding, however, fails when the message reaches a *dead-end* node, a node that is closer to the destination than all its neighbors are. One method to solve this problem is to use local flooding (e.g., by expanding ring search) to find a node that is closer than the current dead-end node to the destination. This method turned out to be costly for networks that are relatively sparse or have holes.

In recently years, a celebrated idea called *perimeter routing* (or face routing) has been proposed and adopted in numerous routing algorithms [3], [9], [12]. The idea is to planarize the network by removing crossing edges. Then, when greedy forwarding fails in the original network, the message is routed from face to face in the planar sub-network toward the destination. That step, termed perimeter routing, is localized and nearly stateless. However, perimeter routing has its serious limitations. It relies on the planarization of the network. Localized network planarization requires the wireless network to be a unit-disk graph (UDG) — defined as a network where two nodes can directly communicate if and only if their Euclidean distance is below a fixed value R — or its close approximation (e.g., a quasi-UDG where the communication range varies by a ratio of at most $\sqrt{2}$ [2]). In practice, however, such idealized connectivity models significantly deviate from many real wireless networks, due to reasons including antenna design, multi-path fading, etc. In addition, the errors in the node positions that the wireless nodes learn from the positioning system (e.g., Global Positioning System or localization methods [5]) also moves the connectivity model away from the UDG model. It is not uncommon to observe stable long links that are five times or more longer than unstable short links in real wireless networks [7].

When a wireless network substantially deviates from the UDG model — a common case in practice, — it becomes provably infeasible to planarize it in a localized and efficient way. Also, planarizing such networks may force them to be disconnected. Recently, a nice attempt has been made to tackle this problem, where the Cross-Link Detection Protocol (CLDP) was proposed [10]. The idea of CLDP is to repeatedly probe the links of the network to remove crossing links (unless removing a link leads to problems such as network partition). Then in the network (nearly) planarized by CLDP, face routing algorithms, such as the well known Greedy Perimeter Stateless Routing (GPSR) algorithm [9], can be used. CLDP, however, does not resolve the *fundamental disadvantage* of network planarization: it can remove a large number of edges such that the distance distortion becomes large in the perimeter routing phase. Our experiments show that when the network deviates

[1] A source node can obtain a destination node's location based on its ID by using location service. And in some applications, such as data-centric storage, a message only needs to be sent to a location without knowing the destination node's ID.

substantially from the UDG model, even if all the edges are short compared to the size of the network-deployment region, the action of planarizing the network requires the removal of a very large number of edges, leading to large distance distortion. Such a disadvantage appears hard to avoid for any routing algorithm based on direct planarization approaches. Besides the high communication complexity of planarization and the concerns over the distance distortion, how to make such methods adaptable to network dynamics (insertion and removal of links or nodes) is also a difficult open problem.

In this paper, we present a novel routing approach for ad hoc wireless networks. We present *face-tracing based routing*, an efficient routing protocol that guarantees delivery for general wireless connectivity models. Similar to existing perimeter routing algorithms, the face-tracing based routing protocol combines greedy forwarding with a mechanism called *face tracing*: when greedy forwarding fails, the message uses face tracing to route out of the dead-end region. The fundamental difference between face tracing and perimeter routing is that with face tracing, the *faces* are not the faces of a planarized sub-network, but the faces of the network itself[2] embedded in a high-genus topological surface. Every edge is in one or two such faces. All the faces can be easily found, without any network embedding or planarization. The faces exhibit a prominent property: they automatically surround holes (regions where no node exists due to node sparsity or obstacles, around which dead-end nodes most likely appear) with high likelihood, and they tend to be localized in regions with no holes. Such a property is very useful for routing a message out of dead-end regions, which is similar to the key reason for the success of perimeter routing in planar graphs. No edge removal is required for the correctness of the protocol. But to improve the performance, the protocol does remove some edges in an efficient way. The number of edges removed, however, is much less than planarization, which makes face tracing much more efficient than perimeter routing due to its small distortion. We show that the face-tracing based routing protocol is highly efficient, scalable, adapts quickly to network dynamics, and is very robust to variations in the network connectivity models.

There has been numerous geographic routing protocols based on perimeter routing, including GPSR [9], the work by Bose et al. [3], Compass routing [11], GOAFR [12], etc. There has also been routing protocols that do not use node locations, but assign *virtual coordinates* to nodes for routing. Examples include GLIDER [6], MAP [4], GEM [13], etc. The latter protocols do not require the network to be a UDG, which is similar to the face-tracing based routing protocol. Comparatively, the face-tracing based routing protocol uses node locations obtained from position systems or localization methods, but does not require the embedding or the building of infrastructures to obtain virtual coordinates.

The rest of the paper is organized as follows. In Section II, we introduce face tracing and study its properties. In Section III, we present the face-tracing based routing protocol. In Section IV, we evaluate the protocol's performance through simulations. In Section V, we present the conclusion.

II. Faces Tracing and Its Properties

In this section, we study face tracing and its properties for wireless networks. Faces can be easily determined, and they exhibit very nice locality properties. We will present the routing protocol based on face tracing in Section III.

A. Faces and face tracing

The concept of the *faces* of a network corresponds to an embedding of the network in a high-genus topological surface. Although our routing protocol *does not* embed the network in any way, understanding the relationship between face tracing and embedding is key for proving the correctness of our protocol and its properties. In the following, we regard a network as a graph $G = (V, E)$ deployed in a plane, with V being the set of nodes and E the undirected edges.

A *topological surface* is an orientable 2-dimensional manifold in which each point has a neighborhood homomorphic to an open disk.[3] Informally speaking, a topological surface is the surface of a solid that contains no "infinitely thin joints". The simplest topological surfaces include spheres and toruses. (See Fig. 1 for examples.) On the other hand, the surface of two balls "glued" at a point does not make a topological surface.

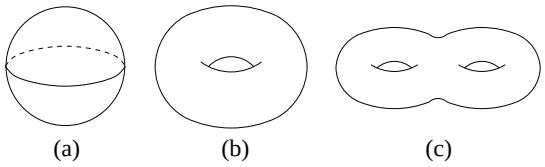


Fig. 1. Topological surfaces. (a) Sphere (genus=0). (b) Torus (genus=1). (c) Two-holed torus (genus=2).

Let G be a connected graph. An *embedding* of the graph G in a topological surface S is a "drawing" of G on S with no edge crossings. We will only consider "cellular embeddings" in which each face of the embedding is homomorphic to an open disk.

To study graph embeddings, the concept of *graph rotation scheme* has to be introduced. Let v be a vertex in the graph G. A *rotation* at v is a cyclic labelling of the edges incident to v. That is, if v has p incident edges $[vu_0]$, $[vu_1]$, $\cdots$, $[vu_{p-1}]$, the rotation at v labels them by $\Pi(0)$, $\Pi(1)$, $\cdots$, $\Pi(p-1)$, where $\Pi(\cdot)$ is some permutation on $\{0, 1, \cdots, p-1\}$. We say that the edge labelled by $(i+1) \mod p$ *follows* the edge labelled by $i \mod p$ or, equivalently, the edge labelled by $i \mod p$ *precedes* the edge labelled by $(i+1) \mod p$. A list of rotations, one for each vertex of G, is called a *rotation scheme* of the graph G. An example of a graph with a rotation scheme is shown in Fig. 2 (a), where the numbers beside edges are their labels.

[2]To be precise, in our protocol implementation, we consider the faces in a "cluster graph" derived from the network, which will be defined later.

[3]Formally, by "X is homomorphic to Y", we mean there is a 1-to-1 mapping π from X to Y such that both π and its inverse are continuous.

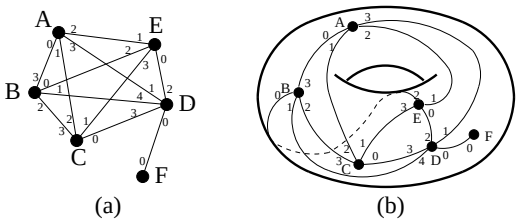


Fig. 2. Graph and its embedding. (a) A graph G with a rotation scheme. (b) Embedding of G in a topological surface.

An embedding of a graph G in a topological surface S naturally induces a rotation scheme for the graph G, as follows. For each vertex v of G, we take a sufficiently small neighborhood D of v on the surface S such that D is homomorphic to a (planar) open disk. Then for the edges incident to v in D, we label them with $0, 1, 2, \cdots$ in the counterclockwise order, which defines a rotation scheme. (See Fig. 2 (b) for an example.) Conversely, by the classical Heffter-Edmonds Principle [8], *every rotation scheme of a graph G induces a unique embedding of G in a unique topological surface.* (See Fig. 2 for an example illustrating the correspondence between rotation scheme and embedding, where Fig. 2(a) is the graph G, and Fig. 2(b) is G's embedding in a topological surface.)

Therefore, as long as a rotation scheme of a graph G is given, we conceptually obtain an embedding of the graph G on some topological surface S. In our routing protocol, we always use the following rotation scheme: we label the edges incident to a node with $0, 1, 2, \cdots$ by the counterclockwise order of the edges *in the plane* where the wireless network is deployed. Note that the embedding corresponding to that particular rotation scheme is still highly non-trivial, because the network itself is usually not planar.

The edges of G partition the topology surface it is embedded in into *faces*. (See Fig. 2 (b).) By *face tracing*, we refer to the process of walking along the edges on the boundary of a face following the right-hand rule. For example, by walking along the edges from A to D to E to A to $D \cdots$ in Fig. 2, we are tracing a face. We can, in fact, do face tracing in the original graph G without finding out its embedding, as the following algorithm **FaceTrace** shows.

First we define a few notations. Each edge $e = [u, v]$ in a graph G has two directions: one is from u to v and the other is from v to u. We will call them "*edge-directions*" and denote them by $\langle u, v\rangle$ and $\langle v, u\rangle$, respectively. Let $\pi(G)$ be a rotation scheme of the graph G. To trace a face starting from an edge-direction $\langle u_0, v_0\rangle$, we apply the following algorithm:

FaceTrace$(\pi(G), \langle u_0, v_0\rangle)$
1. $u \leftarrow u_0$; $v \leftarrow v_0$;
2. **repeat**
output edge direction $\langle u, v\rangle$;
let $[v, w]$ be the edge *following* the edge $[v, u]$ in the rotation at vertex v;
$u \leftarrow v$; $v \leftarrow w$;
until $(u = u_0)\ AND\ (v = v_0)$.

We give some remarks on the **FaceTrace** algorithm. The algorithm traces a sequence of edge-directions, following the orders in the rotations of the vertices appearing in the sequence, and stops when the first edge-direction is encountered again. It should be noted that *the first edge-direction must be encountered again and no edge-direction may appear more than once in the sequence.* To see this, we present a proof here by contradiction. Let $\langle u', v'\rangle$ be the first edge-direction that repeats in this sequence (such an edge-direction must exist because there are only finitely many edge-directions in the graph), and assume that $\langle u', v'\rangle$ is not $\langle u_0, v_0\rangle$. By the algorithm, in order to trace the edge-direction $\langle u', v'\rangle$, we must first follow the edge-direction $\langle w', u'\rangle$, where $[w', u']$ is the edge preceding the edge $[u', v']$ in the rotation at vertex u'. Since $\langle u', v'\rangle$ is not the first edge-direction in the sequence, in order to trace $\langle u', v'\rangle$ twice, we must first trace $\langle w', u'\rangle$ twice. This contradicts the assumption that $\langle u', v'\rangle$ is the first edge-direction that repeats in the sequence. This contradiction proves that the first edge-direction $\langle u_0, v_0\rangle$ must be the first repeated edge-direction in the sequence traced by the algorithm. In consequence, no edge-direction appears more than once in the sequence constructed by the algorithm **FaceTrace**.

Therefore, the sequence of edge-directions constructed by the algorithm **FaceTrace** forms a closed walk, which is the boundary of a face in the embedding $\pi(G)$ of the graph G.

The **FaceTrace** algorithm can start with any edge direction and trace the face that it is in. So clearly, each edge direction in a graph is contained in exactly one face. An edge is involved in either one or two faces, because its two edge directions may or may not be in the same face. If a vertex v is on the boundary of a face f — which we shall call "*the vertex v is in the face f*" in the rest of the paper — a tracing of f must enter and leave v at least once each. So the number of faces that a vertex is in is upper bounded by its degree.

B. Face optimization for geographic routing

In our face-tracing based routing protocol, when greedy forwarding fails, we route the message along faces to get out of the dead-end region. Our extensive simulations show that to improve the routing performance, it is very beneficial to have small faces because of their good *locality* property: small faces tend to surround holes tightly, so they can guide messages to efficiently route around holes to escape from the dead-end regions. In the following, we present three methods for reducing the sizes of faces, which have proved to be very effective in practice.

The first method splits a face into two smaller faces by removing an edge. Assume that a vertex v has p incident edges, which are labelled by $0, 1, \cdots, p-1$ in the rotation scheme. When we remove the edge labelled by j ($0 \le j \le p-1$), the rotation at v changes in this way: now the edge labelled by $(j+1) \mod p$ *follows* the edge labelled by $(j-1) \mod p$, and the '*follow*' relationship for the other edges remain unchanged. For a vertex v, we denote its neighboring vertices by $N(v)$. The first method is as follows:

- **First method:** Let G be a connected graph with a rotation scheme. We remove an edge $[u, v]$ if it satisfies these two conditions: (1) there exists a face f that contains both the edge directions $\langle u, v\rangle$ and $\langle v, u\rangle$; (2) there exists another face g ($g \neq f$) that contains a vertex in $N(u) - v$ and a vertex in $N(v) - u$ (those two vertices can be the same).

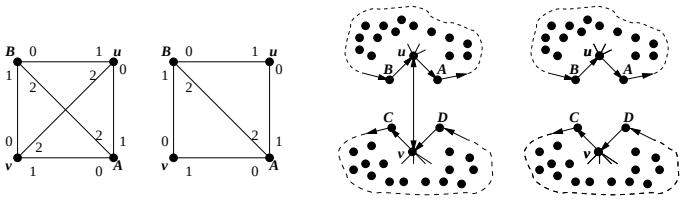


Fig. 3. Split a face by removing an edge. (a) A graph before removing an edge $[u, v]$. (b) After removing edge $[u, v]$. (c) A graph embedded in a topological surface, before removing edge $[u, v]$. (d) After removing edge $[u, v]$.

An example of the above method is shown in Fig. 3 (a), (b). The edge $[u, v]$ in Fig. 3 satisfies the two conditions in the **first method**, where the corresponding face f is $u \to v \to B \to A \to v \to u \to A \to B \to u \to v \to \cdots$, and the face g is $u \to B \to v \to A \to u \to B \cdots$. So $[u, v]$ can be removed. After the removal, the graph is shown in Fig. 3 (b), where the face f has been split into two smaller faces, g remains intact, and the graph is still connected. More generally, we have:

Theorem 1: After removing an edge $[u, v]$ using the **first method**, (1) the face f is split into two smaller faces; (2) the faces in G other than f all remain unchanged; (3) G remains connected.

Proof: Let's say that the face g goes through $a \in N(u) - \{v\}$ and $b \in N(v) - \{u\}$. Face g is a cyclic walk, so g contains a walk from a to b. So naturally we get a walk $u \to a \to \cdots \to b \to v$, and clearly that walk does not contain the edge $[u, v]$. So $[u, v]$ is not a cut edge, removing which will not disconnect the graph.

The face f contains both $\langle u, v\rangle$ and $\langle v, u\rangle$. Since f is a cyclic walk, without loss of generality, the embedding of the graph G in the topological surface is as shown in Fig. 3(c). In Fig. 3(c), We are only showing the edges in the face f, not any other edge in the graph G. (A vertex may appear multiple times in the shown face f. The direction of the walk along f using right-hand rule is shown in Fig. 3(c) by arrows. Note that the topological surface has 'bridges'; so the nodes in the two seemingly closed regions can be connected through the 'bridges'.) Before removing edge $[u, v]$, face f is $\{u \to A \to \cdots B \to u \to v \to C \to \cdots D \to v \to u\}$. After removing edge $[u, v]$, f is replaced by two smaller faces: $\{u \to A \to \cdots B \to u\}$ and $\{v \to C \to \cdots D \to v\}$. Removing edge $[u, v]$ does not affect other faces, because they do not contain the edge directions $\langle u, v\rangle$, $\langle v, u\rangle$, $\langle B, u\rangle$, $\langle u, A\rangle$, $\langle D, v\rangle$ or $\langle v, C\rangle$. ■

The **second method** is simple: if there is a triangle in the graph, we remove its longest edge. Its benefit for creating small faces is validated through experiments.

The **third method** is to work on a *cluster graph* $H = (V_H, E_H)$ instead of the original graph $G = (V, E)$. The cluster graph $H = (V_H, E_H)$ is defined as follows. Partition the vertex set V into disjoint subsets S_1, S_2, $\cdots$, S_k such that for each S_i ($1 \leq i \leq k$), there is a vertex $u_i \in S_i$ that is adjacent to all other vertices in S_i. V_H consists of vertices v_1, v_2, $\cdots$, v_k, such that (1) v_i has the same position as u_i in the plane; (2) there is an edge between v_i, v_j in H if and only if in G, there is an edge connected two vertices respectively in S_i and S_j. Such a graph H is called the *cluster graph of* G. Experiments show that the faces in H are much smaller than the faces in G for wireless networks. Our routing protocol actually routes messages *conceptually* along the faces in H instead of G.

C. Analysis on the locality property of faces

Our extensive simulations show that wireless networks strongly tend to have faces surrounding holes (of moderate or large sizes). This feature becomes especially nice in cluster graphs, where the faces exhibit the following *locality* property: (1) there are faces *closely* surrounding holes most of the time; (2) in the areas where no hole exists, the faces tend to be small and very localized. That property is experimentally shown to hold for a wide range of network models. Examples of faces in wireless networks and in their cluster graphs are shown in Fig. 4. There, Fig. 4(a) and Fig. 4(c) show networks of two different models: quasi-UDG model and directional antenna model [1]. They are two popular models for wireless networks. We will introduce their details in Section IV. Fig. 4(b) and Fig. 4(d) show their corresponding cluster graphs. The methods for reducing face sizes introduced in the previous subsection are applied to the cluster graphs. The original wireless networks tend to have faces surrounding holes but not very localized. Examples of two very large faces of that type are shown in Fig. 4(a) and Fig. 4(c) with thick lines. For the cluster graphs in Fig. 4(b) and Fig. 4(d), three typical faces are shown: a face *closely* surrounding a hole; a face close to the outside boundary of the network; and a randomly selected face in regions with no holes, which is very small and localized. More statistics on faces are presented in Fig. 5 and 6. We discuss them in more detail in Section IV.

The locality property of the faces in the cluster graph is key to the good performance of the face-tracing based routing protocol. It is very intriguing why such a property exists, since the question is related to the complex relationship between the wireless network's geometry in the Euclidean plane and its embedding in a topological surface. In this subsection, we attempt to shed some light on its understanding by studying the robustness of the faces surrounding holes.

The generation of an ad hoc wireless network can be seen as the random generation of nodes and edges following some rules (e.g., an edge cannot be too long). Assume that we have a graph that contains a face surrounding a hole. We consider the following question on the robustness of the face: *if we add or remove edges from the graph, in which case will there no longer be a face surrounding the hole?*

First, let's define a hole in the following way: *A hole is a continuous region in the plane that does not contain any vertex or part of any edge.*

The definition of "if and how" a face surrounds a hole is more subtle. To present the definition, we use a concept called *Surrounding Index (SI)*.

Let G be a graph in a plane, and let h be a hole. Let P denote a walk in G. Let c be a fixed point in the hole h, and let p be a point on the walk P. Consider the ray starting at c and goes through p. When p moves along the walk P with a small step, the ray sweeps the plane with a small angle. We give the angle a positive (negative) sign if the ray sweeps in the counterclockwise (clockwise) direction. The *surrounding index* of the walk P for the hole h, $SI(P, h)$, is defined to be the total angle that the ray sweeps over when the point p moves through the whole walk P exactly once. We see a face as a close walk (where each edge direction is visited only once); therefore, a face's surrounding index must be $2\pi i$, where i is an integer. Note that a face may circle around a hole multiple times, so i may be an integer whose absolute vale is greater than 1. If a face does not enclose a hole, then its surrounding index is 0. Now we define: *A face f* **surrounds** *a hole h if* $SI(f, h) \neq 0$.

If we partition a face f into a set of smaller walks P_1, P_2, $\cdots$, P_k, then clearly, $SI(f, h) = \sum_{i=1}^{k} SI(P_i, h)$.

Let us consider adding an edge $[u, v]$ to graph G. (In the following, we always assume that the rotation scheme labels the edges incident to a vertex based on their counterclockwise order in the plane. The results below can in fact be extended for general rotation schemes.) Let $[u, w_1]$, $[u, w_2]$ be the two edges that, respectively, *precedes* and *follows* edge $[u, v]$ in u's rotation. Let $[v, a_1]$, $[u, a_2]$ be the two edges that, respectively, *precedes* and *follows* edge $[v, u]$ in v's rotation. Before the edge $[u, v]$ is added, let f_1 (resp., f_2) denote the face that contains the edge directions $\langle w_1, u\rangle$, $\langle u, w_2\rangle$ (resp., $\langle a_1, v\rangle$, $\langle v, a_2\rangle$). Then, based on the definitions of face tracing and surrounding index, it is simple to see that the following proposition holds. We skip its detailed proof due to the space limitation.

Proposition 1: Let h be a hole. (1) If f_1 and f_2 are the same face, then the addition of the new edge $[u, v]$ splits it into two different faces f_3 and f_4, where f_3 is "$\cdots w_1 \rightarrow u \rightarrow v \rightarrow a_2 \rightarrow \cdots \rightarrow w_1 \rightarrow u \rightarrow \cdots$" and f_4 is "$\cdots a_1 \rightarrow v \rightarrow u \rightarrow w_2 \rightarrow \cdots \rightarrow a_1 \rightarrow v \rightarrow \cdots$". $SI(f_1, h) = SI(f_3, h) + SI(f_4, h)$. (2) If f_1 and f_2 are two different faces, then the addition of the new edge $[u, v]$ merges them into one face f_3: "$\cdots w_1 \rightarrow u \rightarrow v \rightarrow a_2 \rightarrow \cdots \rightarrow a_1 \rightarrow v \rightarrow u \rightarrow w_2 \rightarrow \cdots \rightarrow w_1 \rightarrow u \rightarrow \cdots$". $SI(f_1, h) + SI(f_2, h) = SI(f_3, h)$.

Now we consider removing an edge $[u, v]$ from graph G. Before $[u, v]$ is removed, let f_1 be the face containing the edge directions $\langle w_1, u\rangle$, $\langle u, v\rangle$, $\langle v, a_2\rangle$, and let f_2 be the face containing the edge directions $\langle a_1, v\rangle$, $\langle v, u\rangle$, $\langle u, w_2\rangle$. Similarly we have:

Proposition 2: Let h be a hole. (1) If f_1 and f_2 are the same face, then the removal of the edge $[u, v]$ splits it into two different faces f_3 and f_4: f_3 is "$\cdots w_1 \rightarrow u \rightarrow w_2 \rightarrow \cdots \rightarrow w_1 \rightarrow u \rightarrow \cdots$" and f_4 is "$\cdots a_1 \rightarrow v \rightarrow a_2 \rightarrow \cdots \rightarrow a_1 \rightarrow v \rightarrow \cdots$". $SI(f_1, h) = SI(f_3, h) + SI(f_4, h)$. (2) If f_1 and f_2 are two different faces, then the removal of the edge $[u, v]$ merges them into one face f_3: "$\cdots w_1 \rightarrow u \rightarrow w_2 \rightarrow \cdots \rightarrow a_1 \rightarrow v \rightarrow a_2 \rightarrow \cdots \rightarrow w_1 \rightarrow u \rightarrow \cdots$". $SI(f_1, h) + SI(f_2, h) = SI(f_3, h)$.

By the above two propositions, when we split a face f_1 surrounding a hole into two faces f_3 and f_4, one of them must still be surrounding the hole. That is because when $SI(f_1, h) = SI(f_3, h) + SI(f_4, h)$ and $SI(f_1, h) \neq 0$, either $SI(f_3, h) \neq 0$ or $SI(f_4, h) \neq 0$. When we merge two faces f_1 and f_2 into one face f_3, if — say, — f_1 surrounds a hole h (so $SI(f_1, h) \neq 0$), then f_3 also surrounds h unless $SI(f_2, h) = -SI(f_1, h) \neq 0$. So to eliminate a face in a graph that surrounds a hole, the only way is to merge it with another face of the opposite surrounding index, where at least one edge need be added. By the collected statistics on faces shown in Section IV, we see that in the cluster graphs, the faces usually closely surround holes and the outer boundary; and because of the 'right-hand rule' for face tracing, often the only type of face pairs of opposite non-zero surrounding indices are a face closely surrounding a hole and a face enclosing the outside network boundary. These restrictions make it less likely to eliminate faces surrounding holes in a graph by adding or removing a small number of edges, which provides some insight on the robustness of the hole-surrounding property.

III. Face-tracing based routing protocol

The face-tracing based routing consists of two modes: the *greedy forwarding* mode, and the *face tracing* mode. A message is first routed in the original network using greedy forwarding. If it reaches a dead-end node v, the message enters the *face tracing* mode and routes along the faces in the network's cluster graph, until it reaches a node that is geographically closer to the destination than v is. Then, the message returns to the greedy forwarding mode. The message alternates between those two modes until it reaches the destination. The nice locality property of the faces make this process very efficient. In the following, we introduce the components of the routing protocol.

A. Preprocessing

The network is preprocessed before any routing starts. The procedure consists of three elements: *building the cluster graph of the network*, *letting nodes recognize the faces (of the cluster graph) they are in*, and *reducing the sizes of the faces using the methods described in Section II*. The specific process is: **First**, the nodes distributively partition the network into very small clusters, where each cluster consists a 'cluster head node' that is adjacent to all the other nodes in the cluster. Every node remembers the connectivity between nodes in its own cluster; and from now on, the nodes in the same cluster acts as one node in the cluster graph. The nodes then distributively build the cluster graph by remembering the edges from their own clusters to the adjacent clusters; **Second**, for each triangle in the cluster graph, the two endpoints of its longest edge

mark the edge as "removed". Note that removed edges will not be used for face tracing; **Third**, each node in the cluster graph uses the **FaceTrace** algorithm to learn the faces they are in, by sending inquiry messages along each incident edge in the cluster graph. The faces are assigned IDs, and the nodes remember the IDs of the faces they are in. To reduce the number of inquiry messages, we let the nodes initiate such inquiry messages asynchronously. If a node receives an inquiry message from an incoming edge e, it no longer needs to initiate an inquiry message along the outgoing link that *follows* e. To improve routing performance, each node also remembers the positions of t randomly selected nodes in every face that the node is in. (We find through experiments that $t = 5$ is sufficient.) We call t the **sampling rate**. **Fourth**, if there is a link $[u, v]$ in the cluster graph that can be removed by using the first method presented in Section II for reducing faces sizes, we remove the edge $[u, v]$. By Theorem 1, only a face containing u and v is affected (which is split into two faces). So only u and v send out two messages to trace the two new faces, and inform all the nodes in those two faces of that change. The above operations can all be implemented in a very *efficient*, *distributed* and *asynchronous* way.

B. Routing

The routing consists of two modes: *greedy forwarding* and *face tracing*. When a message just enters the face tracing mode at node v, among the faces containing v, we heuristically choose the face that contains a sampled node whose Euclidean distance to the destination is the minimum. (Recall that v remembers the positions of t sampled nodes in the face.) The message is routed in that face using the **FaceTrace** algorithm, which is nearly **stateless**. If that face does not get the message any closer to the destination, then the message routes from face to face. (Two faces are *adjacent* if they share an edge. The message goes from one face to another through a vertex in such a common edge.) Every time it enters a face, it routes along that face to see if can get closer to the destination than the previous dead-end node does. Note that going from face to face in the cluster graph is the same as going from vertex to vertex in the dual graph of the embedded cluster graph in its corresponding topological surface. Therefore, if we traverse all the faces in this way, we can reach the whole graph, including the routing destination. In our implementation, the message remembers the IDs of the faces it has traversed in the current round of face tracing, and uses the depth-first search (DFS) to go from face to face. So the delivery is guaranteed. The overhead for remembering the traversed faces' IDs is very small, due to the protocol's ability to route messages out of dead-end regions quickly. Note that each cluster of the network acts as one node in the cluster graph. Since a cluster is of diameter 2 or less, realizing the face tracing in the true network is very simple.

C. Network dynamics

In a wireless network, links and nodes may come and go. Our protocol adapts to such network dynamics efficiently. By Propositions 1 and 2, when a link is added or removed, at most two faces are affected, so only two messages need be sent by the two endpoints of the link to learn the new faces. Adding or removing a node is the same as adding or removing its incident links. The only additional case to consider is that when nodes/links are added or removed, clusters can change, appear or disappear. As nodes in the same cluster remember connectivity information about the whole cluster, such changes can be efficiently processed.

IV. Simulation

We have implemented the face-tracing based routing protocol, and conducted extensive simulations for various network connectivity models and deployment environments. The protocol has shown very stable performance across the various environments and parameter configurations. In this section, we present simulations for a typical set up of ad hoc wireless networks, and consider two different wireless connectivity models: the *quasi unit-disk graph* (quasi-UDG) model, and the *directional antenna* (DA) model.

The *quasi-UDG* model is a generalization of the UDG model for wireless networks. It has three parameters: R, r and p. ($R \geq r$, $0 \leq p \leq 1$.) An edge exists (does not exist) between two vertices if their Euclidean distance is less than r (more than R); if the Euclidean distance is between r and R, the edge exists with probability p.

The model we adopt for *directional antennas* (DA) is a simplification of the real DAs [1]. It has two parameters: R_{DA} and θ. ($0 \leq \theta \leq 2\pi$.) A vertex u can directly send messages to a vertex v if and only if v falls inside a cone of angel θ rooted at u and is also within Euclidean distance R_{DA} from u. The orientation of that cone is uniformly randomly selected. There is an edge between two vertices if and only if they can both directly send messages to each other.

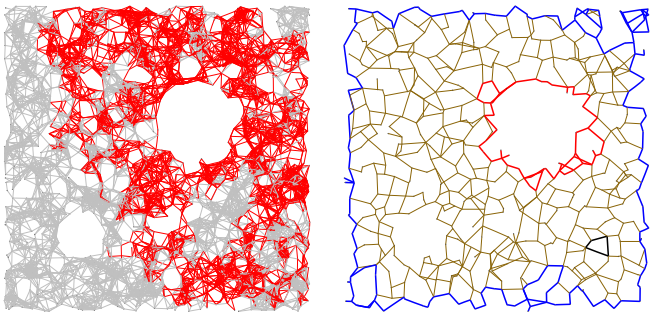
In the experiments, we uniformly randomly deploy n_0 wireless nodes in a 2-D space of size 20×20. To mimic nontrivial deployment environments, we randomly put two holes (areas where nodes cannot be placed) of radius 1.5 and 2.5 in the plane. (The network also has naturally formed voids due to the sparsity of nodes.) Corresponding to each fixed set of parameters, we randomly generate 30 networks. Then in each network, we randomly pick 10,000 source and destination pairs for routing.

The focus of these experiments is to verify the validity of the new geographic routing approach based on face tracing. We concentrate on the topological level of the routing, and study the routing performance, properties of faces, network preprocessing overhead, packet overhead and adaptivity to network dynamics. Many important factors at the MAC layer, such as link quality or packet acknowledgement, have not been addressed and will be studied in our future work. We compare the face-tracing based routing protocol with the current geographic routing approach that uses perimeter routing. In particular, we compare it with the combination of GPSR [9] and CLDP [10]. GPSR is a well known geographic routing protocol, and CLDP is a novel network planarization protocol

that supports GPSR. The performance of combining GPSR with CLDP has been studied in [10]. We also compare with a popular geographic routing approach that combines greedy forwarding with local flooding. When greedy forwarding fails, that approach uses local flooding (expanding ring search with doubling radius) to route out of the dead end region. The experiment results show that the face-tracing based routing approach has a much better performance.

A. Statistics on faces

Fig. 4 shows some typical examples of the quasi-UDG networks, directional antenna (DA) networks, their cluster graphs, and some faces in them. Details of the figures were introduced in Section II, so we skip them here. We comment that in nearly all the cluster graphs generated in the experiments, there are faces surrounding the holes and the outside boundary. In areas with no holes, the faces are very small and localized. The type of faces most helpful for getting a message out of a dead-end region are those that *closely* surround relatively large holes. Let's define a face to be *closely* surrounding a hole if the average Euclidean distance from the vertices in the face to the boundary of the hole is less than Δ. We set $\Delta = 4$ here. The statistics on such *close* faces are shown in Fig. 5.



(a) Quasi-UDG and a large face

(b) Cluster graph of quasi-UDG and 3 faces

(c) DA graph and a large face

(d) Cluster graph of DA graph and 3 faces

Fig. 4. Examples of quasi-UDG and directional antenna (DA) networks, their cluster graphs, and examples of the faces (represented by dark edges) in them. 2000 nodes are deployed in a 20×20 plane. For the quasi-UDG, $R = 1$, $r = 0.1$, $p = 0.5$, average degree is 7.885. For the DA graph, $\theta = 120^o$, $R_{DA} = 2$, average degree is 7.290.

As shown in Fig. 5, the probability of having faces *closely* surrounding the holes is high in most cases. In some cases, that probability becomes relatively lower. That is because in those cases, the faces surrounding the holes become large and contain vertices further away from the holes, and we do not count them as '*close*.' Fig. 6 shows that the average face size is very small. That also indicates the strong locality of the faces. It is also shown in Fig. 6 that on average, a vertex is contained only in a small number of faces.

		Statistics on faces **closely** surrounding holes					
		Fraction of networks containing faces that **closely** surround both holes and the boundary			Average distance between face vertices and the holes they surround		
		hole-1	hole-2	boundary	hole-1	hole-2	boundary
n_0	$\frac{R}{r}$	Network Connectivity Model: Quasi-UDG					
2000	2	100%	100%	100%	0.740	0.751	0.686
2000	10	100%	97%	100%	1.036	1.065	1.070
4000	2	100%	100%	100%	0.853	0.664	0.768
4000	10	87%	60%	20%	3.851	3.514	1.485
n_0	θ	Network Connectivity Model: Directed Antenna					
2000	90^o	57%	39%	12%	1.861	1.323	0.662
2000	150^o	86%	62%	11%	1.682	1.542	0.673
4000	90^o	77%	60%	6.8%	1.752	1.593	0.675
4000	150^o	100%	89%	11%	1.666	1.671	0.746

Fig. 5. Statistics on faces that **closely** surround holes and the outside boundary, in the cluster graphs of quasi-UDG networks and directional antenna (DA) networks. n_0 is the number of vertices in the original networks. $R = 1$, $p = 0.5$, $R_{DA} = 2.5$. 'Hole-1' and 'hole-2' are the two randomly placed holes of radius 1.5 and 2.5, respectively. 'Boundary' is the outside boundary of the deployment region.

		Statistics on all faces			Statistics on Network		
		Avg face size	Std dev. face size	Avg #faces per node	m_0	n	m
n_0	$\frac{R}{r}$	Network Connectivity Model: Directed Antenna					
2000	2	5.219	6.032	4.768	9877.97	449.07	1484.13
2000	10	6.465	11.257	4.821	7921.80	563.83	2022.27
4000	2	7.405	9.442	6.331	39600.30	573.77	2905.27
4000	10	10.705	29.011	7.519	31663.80	754.30	4637.17
n_0	θ	Network Connectivity Model: Directed Antenna					
2000	90^o	10.205	27.047	3.891	5451.57	838.03	2361.37
2000	150^o	12.114	25.731	5.968	12598.70	609.67	2811.73
4000	90^o	17.603	39.947	6.427	20818.70	1394.33	6698.17
4000	150^o	21.053	37.285	8.690	49956.00	937.90	6440.43

Fig. 6. Statistics on faces in the cluster graphs of quasi-UDG networks and directional antenna (DA) networks, and the networks themselves. $R = 1$, $p = 0.5$, $R_{DA} = 2.5$. The six columns of data are, respectively: the average face size, the standard deviation of face size, the average number of faces that a node of the cluster graph is in, the number of edges in the original network (m_0), the number of vertices in the cluster graph (n), and the number of edges in the cluster graph (m).

B. Network Preprocessing Overhead

Both the face-tracing based routing protocol and CDLP require preprocessing when the network is initialized. Our protocol does clustering and requires nodes to recognize faces. CLDP probes the network to remove crossing links. The total number of messages sent is taken as the preprocessing overhead. When the same message is transmitted over k hops, we count it as k messages. The results are shown in Fig. 7. We see that the face-tracing based protocol improves the overhead significantly, by a factor of 10^3 to 10^4.

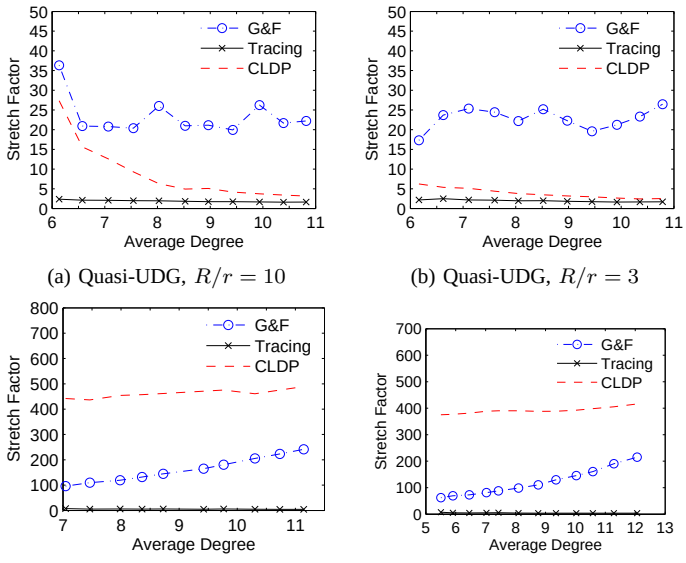
C. Quality of Routing Paths

Given a source and a destination, if greedy forwarding alone succeeds, the face-tracing based routing (short as *Tracing here*), CLDP+GPSR (short as *CLDP* here), and greedy

	Quasi-UDG model			
	$n_0 = 2000$ $R/r = 2$	$n_0 = 2000$ $R/r = 10$	$n_0 = 4000$ $R/r = 2$	$n_0 = 4000$ $R/r = 10$
Tracing	1.40×10^4	1.37×10^4	4.15×10^4	3.93×10^4
CLDP	1.67×10^7	1.63×10^7	1.77×10^8	1.79×10^8
	Directional antenna model			
	$n_0 = 2000$ $\theta = 90^o$	$n_0 = 2000$ $\theta = 150^o$	$n_0 = 4000$ $\theta = 90^o$	$n_0 = 4000$ $\theta = 150^o$
Tracing	1.73×10^4	2.06×10^4	4.24×10^4	6.17×10^4
CLDP	1.94×10^7	5.37×10^7	1.83×10^8	4.67×10^8

Fig. 7. Network preprocessing overhead: number of messages sent. $R = 1$, $p = 0.5$, $R_{DA} = 2.5$. Here 'Tracing' refers to the face-tracing based routing protocol.

forwarding plus local flooding (short as *G&F* here) produce exactly the same routing path. In our experiments, greedy forwarding succeeds in around 30% to 70% cases for quasi-UDGs, while for directional antenna (DA) graphs this percentage is less than 1%. We compare the routing performance only for the cases where greedy forwarding alone does not succeed. Define *stretch factor* as the average ratio of the hop distance in a routing path (generated by one of the three routing protocols) and the minimum hop distance between the source and the destination. For a good understanding, we measure the routing performance while considering the changes in vertex degree, vertex density, network size, and face sampling rate.

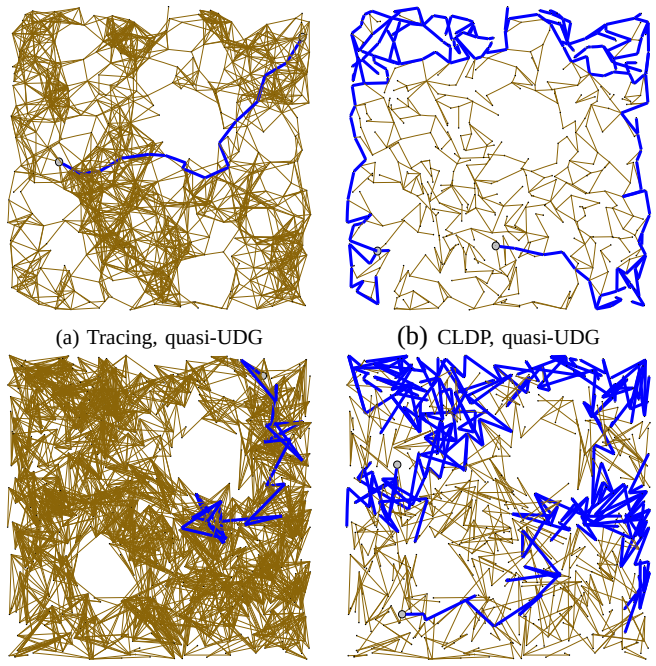


(a) Quasi-UDG, $R/r = 10$ (b) Quasi-UDG, $R/r = 3$

(c) DA, $\theta = 90^o$ (d) DA, $\theta = 120^o$

Fig. 8. Stretch factor vs. average vertex degree in original networks. $R = 1$.

1) Stretch factor vs. node degree: For quasi-UDGs (DA graphs), we adjust the value of connectivity probability p (the radius R_{DA}) to change the average vertex degree. The average routing stretch factors are shown in Fig. 8. The face-tracing based routing protocol performs much better than the other two. Its stretch factor is in the ranges of $[1.59, 2.37]$, $[1.63, 2.47]$, $[4.48, 6.87]$, $[3.57, 7.20]$ for Fig. 8 (a), (b), (c) and (d). It exhibits a particular stable performance, with a stretch factor that is several, tens or even hundreds of times better than the other two routing protocols.



(a) Tracing, quasi-UDG (b) CLDP, quasi-UDG

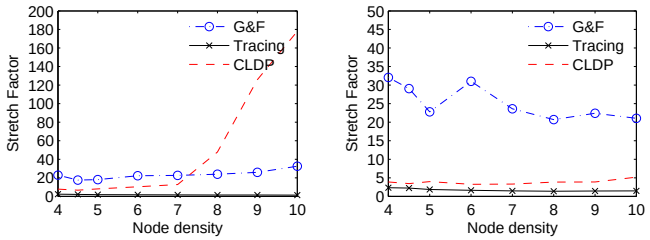
(c) Tracing, DA (d) CLDP, DA

Fig. 9. Examples of routing paths (represented by dark edges). The underlying networks in (a) and (c) are the original networks. The underlying networks in (b) and (d) are showing only the edges not removed by CLDP.

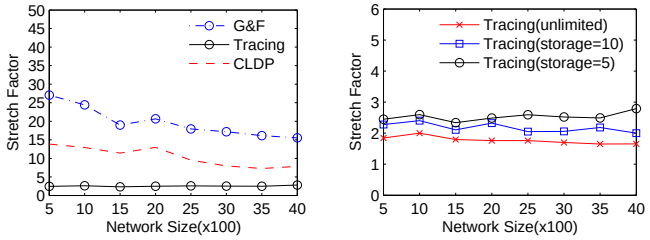
Examples of the routing paths are shown in Fig. 9. The paths are represented by thick lines. The face-tracing based routing protocol outperforms greedy forwarding plus local flooding because the faces guides messages around holes efficiently. (See Fig. 9 (a) for an example, where the left end of the path is the source.) It also outperforms CLDP+GPSR because it removes much fewer edges than CLDP does. (Note that both use all the edges in the greedy forwarding mode.) CLDP usually removes about twice the number of edges than our protocol. Fig. 9 (b) and (d) show how sparse the network can be when links are removed by CLDP for perimeter routing.

2) Stretch factor vs. vertex density, network size, and face sampling rate: We increase the number of vertices, and measure the routing stretch factors. The results are shown in Fig. 10 (a), (b). (We skip the results for directional antenna graphs due to limited space.) The stretch factor for our protocol is in the ranges $[1.38, 2.35]$ and $[1.39, 3.41]$ for Fig. 10 (a) and (b). Again, it stably outperforms the other two protocols.

We increase the network's deployment area and the number of vertices, while keeping the vertex density and average degree constant. The results are shown in Fig. 10(c), which indicate that all three routing protocols are scalable in the network size for stretch factors. We also change the face sampling rate (the number of sample vertices a node remembers about a face) from 5 to 10 to infinity. As shown in Fig. 10 (d), the performance of our protocol does not change much. It means that setting the sampling rate to be 5 is already sufficient. In all our other experiments, we use the sampling rate 5.



(a) Quasi-UDG, $R/r = 10$ (b) Quasi-UDG, $R/r = 3$



(c) Quasi-UDG, $R/r = 10$ (d) Quasi-UDG, $R/r = 10$

Fig. 10. (a) and (b): Stretch factor vs. node density (number of nodes per unit area). (c) Stretch factor vs. network size (the total number of nodes) (d) Stretch factor vs. face sampling rate.

D. Packet Overhead

With the face-tracing based routing, when a message enters the face tracing mode, it needs to remember the faces it has traversed in this round. We call such a storage overhead in the message the *packet overhead*. The average packet overhead *for the face tracing mode* is shown in Fig. 11. The unit there is the the ID of a face, which is roughly $\log_2 n$ bits, where n is the number of vertices in the cluster graph. We see that on average the message traverses only a few faces in a face-tracing phase, which is very efficient.

Quasi-UDG				Directional antenna			
$n_0 = 2000, R/r = 2$	$n_0 = 2000, R/r = 10$	$n_0 = 4000, R/r = 2$	$n_0 = 4000, R/r = 10$	$n_0 = 2000, \theta = 90^o$	$n_0 = 2000, \theta = 150^o$	$n_0 = 4000, \theta = 90^o$	$n_0 = 4000, \theta = 150^o$
1.24	1.69	1.04	2.21	2.82	2.98	4.52	4.20

Fig. 11. Average packet overhead of the face-tracing based algorithm, for the face-tracing mode. $R = 1$, $p = 0.5$, $R_{DA} = 2.5$.

E. Adaptivity to Network Dynamics

When there are dynamics in wireless networks (the deletion/insertion of links or nodes), the face-tracing based routing protocol needs to recognize the new faces, while CLDP needs to re-probe the network to remove crossing links and restore some other links. Our protocol has the advantage that its adaptation to the network dynamics is *on demand*: it can efficiently recognize new faces with localized operation only when links/nodes are deleted/inserted. CLDP chooses the strategy of periodic probing of the whole network. We measure the number of messages that our protocol needs to send to recognize new faces and adapt to the changed network. When the same message is sent over k hops, we count it as k messages. The results are shown in Fig. 12. We see that the overhead is very low.

Quasi-UDG				Directional antenna			
$n_0 = 2000, R/r = 2$	$n_0 = 2000, R/r = 10$	$n_0 = 4000, R/r = 2$	$n_0 = 4000, R/r = 10$	$n_0 = 2000, \theta = 90^o$	$n_0 = 2000, \theta = 150^o$	$n_0 = 4000, \theta = 90^o$	$n_0 = 4000, \theta = 150^o$
Deletion of a link							
17.4	23.7	24.2	21.8	37.5	25.8	49.7	32.2
Insertion of a link							
12.7	9.7	24.0	18.2	1.9	9.1	4.1	19.1
Deletion of a node							
138.5	205.7	140.6	177.9	491.4	269.6	557.4	298.1
Insertion of a node							
222.7	249.2	263.9	269.2	462.3	284.7	470.0	312.7

Fig. 12. Average number of messages sent for the deletion/insertion of a link/node. When a message is sent over k hops, it is counted as k messages.

V. Conclusion

In this paper, we introduce a novel geographic routing approach based on face tracing. It exhibits excellent routing performance. It will be very interesting to further explore the properties of face tracing and its application in routing. We are also interested in the further optimization of the routing protocol and its integration with existing network protocols. Those remain as our future research.

***Acknowledgments** This research is supported in part by the NSF Grant CCF-0430683 and by the Texas Advanced Research Program under Grant No. 003581-0006-2006.

References

[1] C. A. Balanis, *Antenna Theory: Analysis and Design*, 3rd edition, Wiley-Interscience, 2005.

[2] L. Barriere, P. Fraigniaud and L. Narayanan, Robust position-based routing in wireless ad hoc networks with unstable transmission ranges, in *Proc. of DialM*, pp. 19-27, 2001.

[3] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, Routing with guaranteed delivery in ad hoc wireless networks, in *3rd Int. Workshop on Discrete Algorithms and methods for mobile computing and communications (DialM '99)*, pp. 48–55, 1999.

[4] J. Bruck, J. Gao and A. Jiang, MAP: Medial axis based geometric routing in sensor networks, in *Proc. ACM International Conference on Mobile Computing and Networking (MobiCom)*, pp. 88–102, 2005.

[5] J. Bruck, J. Gao and A. Jiang, Localization and routing in sensor networks by local angle information, in *Proc. the Sixth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pp. 181-192, 2005.

[6] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang, GLIDER: Gradient landmark-based distributed routing for sensor networks, in *Proc. INFOCOM*, March 2005.

[7] 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.

[8] J. Gross and T. Tucker, Topological graph theory, *Wiley-Interscience*, New York, 1987.

[9] B. Karp and H. Kung, GPSR: Greedy perimeter stateless routing for wireless networks, in *Proc. MobiCom*, pp. 243-254, 2000.

[10] Y.-J. Kim, R. Govindan, B. Karp and S. Shenker, Geographic routing made practical, in *Proc. NSDI*, 2005.

[11] E. Kranakis, H. Singh and J. Urrutia, Compass routing on geometric networks, in *Proc. 11th Canadian Conference on Computational Geometry*, 1999.

[12] F. Kuhn, R. Wattenhofer, Y. Zhang and A. Zollinger, Geometric Ad-Hoc Routing: Of Theory and Practice, *Proc. PODC*, pp. 63-72, 2003.

[13] J. Newsome and D. Song, GEM: Graph EMbedding for routing and data-centric storage in sensor networks without geographic information, in *Proc. of SenSys*, 2003.