Geographic Routing with Constant Stretch in Large Scale Sensor Networks with Holes

Myounggyu Won, Radu Stoleru, Haijie Wu
Department of Computer Science and Engineering, Texas A&M University
{mgwon, stoleru, bylike}@cse.tamu.edu

***Abstract*—Geographic routing is well suited for large scale sensor networks deployments, because the per node state it maintains is independent of the network size. However, due to the "local minimum" caused by holes/obstacles, the path stretch of geographic routing can grow as $O(c^2)$, where c is the length of the optimal path. Recently, VIGOR, a geographic routing protocol based on the *visibility graph*, shows that a constant path stretch can be achieved. This, however, is possible with increased overhead. To address this issue, we propose GOAL (Geometric Routing using Abstracted Holes), a routing protocol that *provably achieves a constant path stretch, with lower message, space and computational overhead.* We develop a novel distributed convex hull construction (DCC) algorithm that compactly describes holes. This compact representation of a hole is leveraged by nodes to make locally optimal routing decisions. Our theoretical analysis proves the constant stretch property and average stretch of GOAL. Through extensive simulations and a hardware implementation, we demonstrate the effectiveness of GOAL and its feasibility for large-scale sensor networks. In our network settings, GOAL reduces the energy consumption by up to 32%, routing table size by an order of magnitude, when compared with VIGOR.**

I. Introduction

Geographic routing has attracted much attention from the WSN research community, because it is simple and scalable. In geographic routing, a source node obtains the location of a destination node by using a location service [1], or hash-functions in a data centric storage scheme [2]. A packet is forwarded to the neighbor who is closest to the destination. This greedy approach allows near-optimal path length in uniform and dense networks without holes/obstacles. However, as Kuhn et al [3] proved, the path length can grow as much as $O(c^2)$, where c is the optimal path length, when the holes/obstacles are present, because of the "local minimum" phenomenon.

In order to overcome the local minimum and ultimately improve the path quality, some geographic routing protocols using the non-local information have been proposed [4][5][6][7]. In these protocols, the holes are identified beforehand, and the size and shape of the identified holes are propagated to some nodes so that forwarding nodes can use the information to avoid the holes. However, for a resource-constrained WSN, the information about the hole (the size and shape of the hole) can only be made known to a limited subset of nodes, e.g., the nodes located at the boundary of the hole (referred as boundary nodes), or some of their neighbors, due to the large volumes of data to describe the holes. Thus, a "reaction" to the local minimum is only activated when a packet reaches the node at the local minimum (i.e., a *stuck node*), or some neighboring nodes of the stuck node. This late reaction problem results in a suboptimal routing path.

Recently, Tan et. al. [8] proposed VIGOR, a geographic protocol that achieves a constant path stretch. In VIGOR, the hole is represented as a polygon, which is used to build the visibility graph, a structure often used in computational geometry to find the shortest path, given obstacles. However, VIGOR suffers from non-negligible protocol-related message overhead. First, in order to build the visibility graph, the locations of all the VON nodes (the vertices of the polygons) need to be flooded to all the nodes in the network. Second, the VON nodes must iteratively exchange messages among themselves until a complete routing table is constructed. The performance deteriorates further when the hole has a complex shape, resulting in many VON nodes.

In this paper, we propose a geographic routing protocol that achieves a constant path stretch while incurring less overhead. In this protocol, the holes are compactly described as a set of extreme points of the convex hull covering the boundary nodes of the hole. A route to and from the nodes inside the convex hull is efficiently handled for guaranteed packet delivery. A novel distributed convex hull algorithm is introduced to build the convex hull of the hole with low message complexity. A distributed extreme points reduction algorithm further reduces the size of the hole information. Consequently, the locations of only a few extreme points of the convex hull are locally broadcast to nodes within h hops from the hole. Based on the small database about nearby convex hulls, a source node identifies the interfering holes within h hops that block the straight path to the destination with small computational overhead. A source node then computes a set of intermediate destinations that guide a packet along the locally optimal path. When a packet reaches a node with better hole information, the intermediate destinations are updated, incrementally improving the routing path. The contributions of this paper are as follows:

- We develop a geographic routing protocol that generates a path with constant stretch in networks with holes. The protocol has low message and computational complexity.
- We develop a distributed convex hull algorithm to efficiently reduce the size of data that describes a hole.
- We present a thorough analysis to prove the correctness and constant path stretch of our protocol.
- We perform extensive simulations and a hardware implementation to confirm the effectiveness of our protocol, and its feasibility for WSN.

978-1-4577-2014-7/11/$26.00

II. Related Work

Routing protocols for large scale WSN can be largely divided into geographic routing and hierarchical routing [9]. Recently, S4 [9], a hierarchical routing protocol, was shown to achieve the worst case stretch of 3 with small per node state $O(\sqrt{N})$, where N is the total number of nodes. Although S4 obtains a desirable balance between the path stretch and state size, unlike geographic routing protocols, the per node state depends on the network size N. In contrast, geographic routing suits particularly well to resource constrained large-scale WSNs, since it is stateless and fully distributed. However, the path stretch of geographic routing protocol can be as bad as $O(c^2)$, where c is the optimal path length, due to the local minimum caused by topological complexes like holes. Little was known about achieving a constant stretch for geographic routing protocols.

Several geographic protocols have been proposed to improve the path quality by using the non-local information. In [5], a node uses the TENT rule to test if it is a stuck node. If a node finds itself to be the stuck node, BOUNDHOLE algorithm is run to build a route around the hole by discovering a set of the boundary nodes to guide a packet out of the local minimum. The boundary nodes are marked and used as landmarks for a future packet to bypass the hole. However, the "late reaction problem" (a detour is made at the boundary node) degrades the average path length. In [7], finding an angle between two adjacent neighbors to be greater than predefined threshold, the node is "elevated" so that this node is avoided by the greedy forwarding process. However, a decision on whether a node is at a local minimum or not depends on the locations of source and destination. Thus, such heuristic approach results in the frequent failure of the algorithm. To remedy this problem, in [6], a network is divided into k regions. Each node maintains a vector of size k, where each element indicates whether this node is a local minimum for the i-th region or not. To determine the value of each element, the local minimum angle b is defined, and if the region is covered more than a certain percentage by this angle, the region is considered to be the local minimum region. However, none of these protocols provide the worse case path length guarantees.

Several researchers proposed to propagate the information about the holes to some nodes in a limited region. In [4], a stuck node and some of its neighbors form an unsafe area shaped as a rectangle. The estimated shape of the hole is known to the nodes in this unsafe area. This distributed information model is used to avoid the local minimum. Although the path stretch is improved due to the propagation of the hole information to some nodes, only a few neighboring nodes of the stuck node are aware of the information, so the "late reaction" problem persists. In [10], the hole is regularized with an ellipse, and the abstracted information about a hole is broadcast to the nodes within h hops from the boundary of the ellipse. However, the ellipse often fails to represent various kinds of hole shapes. More importantly, the worst case path stretch is not guaranteed.

Recently, Tan et. al. [8] introduced VIGOR, a geographic routing protocol that achieves a constant path stretch. VIGOR finds a near optimal routing path by exploiting the *visibility graph*. A hole is represented as a set of *VON nodes*, and a virtual overlay network consisting of these *VON nodes* guides a packet along the close-to-shortest path that bypasses the holes. One notable aspect of VIGOR compared with S4 is that its per node state is $O(N_{von})$, where $N_{von}(<< N)$ is the total number of VON nodes, thereby eliminating the dependency of the per node state on the network size. However, there are several technical issues, other than the protocol-related message overhead discussed in Section I, to be solved. First, before packet transmission starts, a source node has to set up a routing path, by exchanging the visibility set with a destination node. This routing set up process incurs additional overhead and delay for each source and destination pair. Second, the probing message, used to build the VON polygon, might contain a large amount of data, since the message is piggybacked with all the locations of the boundary nodes that the message has visited until a new VON node is found.

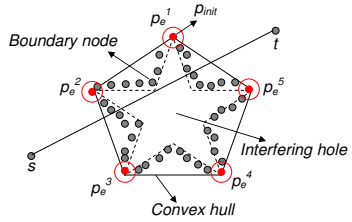


Fig. 1. An illustration of notations and terms.

III. Preliminaries

In this section, we define the notations and terms used throughout this paper. These notations are illustrated in Figure 1. We consider a dense wireless sensor network, consisting of N nodes $V = \{v_1, v_2, ..., v_N\}$, uniformly distributed in a two dimensional space. We assume that each node knows about its location. There are m holes in the network, denoted by $H_1, H_2, ..., H_m$. Each hole H_i is surrounded by a set of *boundary nodes*, $P^i = \{p_1, p_2, ..., p_n\}$, $p_j \in V$. A hole is either a closed cycle ($p_1 = p_n$), or a chain ($p_1 \neq p_n$) that forms a cycle with the edge(s) of a network. We define one boundary node in each set P^i as the *initiator* as the following:

Definition 1: An *initiator*, denoted by p_{init}, is a boundary node in P^i with the highest y coordinate. If there are more than one boundary nodes with the same highest y coordinates, one with the lowest x coordinate among them becomes the initiator p_{init}. □

A hole H_i is represented as a set of extreme points, denoted by $P_e^i = \{p_e^1, p_e^2, ..., p_e^n\}$, of the convex hull covering the set of boundary nodes P^i, where the extreme point p_e^j is defined as the following (Note that we will use the terms "node" and "point" interchangeably):

Definition 2: An *extreme point* is the corner point of a convex hull.

A path between source s and destination t is denoted by $\overrightarrow{st}$, and the length of the path, as the number of hops, between s

and t is denoted by $|\overrightarrow{st}|$. Given a path $\overrightarrow{st}$, the *interfering holes* are formally defined as the following:

Definition 3: Given any two points p and q, $\mathbb{L}(\overline{pq})$ is a set of points on the left hand side of a vector $\overrightarrow{pq}$, and $\mathbb{R}(\overline{pq})$ represents a set of points on the right hand side of $\overrightarrow{pq}$. □

Definition 4: Given a path $\overrightarrow{st}$, a hole represented by a set of extreme points, P_e, is called the *interfering hole* iff there exists some $p_e^k \in P_e, 1 \leq k \leq |P_e| - 1$ such that $p_e^{k-1} \in \mathbb{L}(\overline{st})$ and $p_e^{k+1} \in \mathbb{R}(\overline{st})$, or $p_e^{k-1} \in \mathbb{R}(\overline{st})$ and $p_e^{k+1} \in \mathbb{L}(\overline{st})$. □

In particular, two nodes are *visible* to each others iff there is no interfering holes between them.

IV. GOAL: GEOMETRIC ROUTING USING ABSTRACTED HOLES

A. Protocol Overview

Our routing protocol consists of mainly two components: the hole abstraction and packet forwarding. The hole abstraction process is designed to produce the compact representation of the hole. In this process, the hole is represented as a set of extreme points of a convex hull covering the boundary nodes of the hole. The hole abstraction process consists of three phases. First, the boundary nodes surrounding the hole are identified using either statistical [11] or topological methods [12]. The second phase is the construction of the convex hull: a set of extreme points for each hole is found by the single traverse of a probing packet along the boundary nodes in a fully distributed manner. In the last phase, the abstracted information about each hole is broadcast to the nodes within h hops from the hole.

When the hole abstraction process is finished, each node knows about the locations of the extreme points of the holes within h hops. This information is used to make a routing decision. Specifically, source s identifies the interfering holes blocking the straight path to destination t, and runs our forwarding algorithm to compute a subset of extreme points among all the extreme points that belong to the interfering holes. This subset of extreme points are the intermediate destinations used to guide a packet along the path $\overrightarrow{st}$ that minimizes $|\overrightarrow{st}|$. A packet carries the locations of intermediate destinations, and is forwarded to each intermediate destination by simple geographic forwarding. When a packet reaches an intermediate destination, the routing algorithm is rerun and the previous intermediate destinations are updated if necessary. The details of the protocol are presented in the following sections.

B. DCC: Distributed Convex Hull Construction

We adopt [12] to find the boundary nodes. If the boundary nodes form a closed cycle, p_{init} is elected using an existing leader election algorithm on a ring topology, which has message complexity $O(n \log n)$, where n is the number of the boundary nodes [13]. If the boundary nodes form a chain, one of the two boundary nodes at each end of the chain is elected as p_{init}. Specifically, if node p finds that it is the boundary node at the end of the chain by checking the number of neighboring boundary nodes, node p sends a message containing its x and y coordinates to the boundary node q at the other end of the chain. If node q's y coordinate is greater than node p's y coordinate, node q becomes p_{init}. If node q's y coordinate is the same as node p's y coordinate, x coordinates are compared, and if node q's x coordinate is smaller, then node q elects itself as p_{init}.

Algorithm 1 DCC (code for p_{init})

```
1: if hop_count = 0 then
2:     P_e ← P_e ∪ {p_init}
3:     send a probing packet in counter-clockwise.
4: else
5:     if |N_{p_i}| = 1 or p_init ∈ ℝ(p_e^{|P_e|} p_1) then
6:         terminate.
7:     end if
8: end if
```

Algorithm 2 DCC (code for p_i)

```
1: for each p_e^l ∈ P_e, 0 ≤ l ≤ |P_e| do
2:     if ∀m, l+1 ≤ m ≤ |P_e|, p_e^m ∈ 𝕃(p_e^l p_i) then
3:         P_e ← P_e \ {p_e^{l+1}, p_e^{l+2}, ..., p_e^{|P_e|}}
4:     end if
5: end for
6: if p_i ∈ ℝ(p_e^{|P_e|} p_{i+1}) then
7:     P_e ← P_e ∪ {p_i}
8:     // EPRA
9:     if |P_e| > Threshold then
10:        find p_crs^i with minimum d_crs^i
11:        p_e^i ← p_crs^i
12:        P_e ← P_e \ p_e^{i+1}
13:    end if
14:    // End of EPRA
15:    forward a probing packet to p_{i+1}.
16: else
17:    forward a probing packet to p_{i+1}.
18: end if
```

Once p_{init} is elected, p_{init} initiates the DCC algorithm. Algorithm 1 describes the pseudo code for p_{init}. p_{init} adds its location to the set P_e as the first extreme point p_e^0, and piggybacks the set P_e on a probing packet. This packet is sent to p_{init}'s left neighboring boundary node, starting to traverse the boundary nodes of the hole in a counter-clockwise direction. Upon receiving the packet, the boundary node examines if it is the extreme point of the convex hull by executing Algorithm 2.

Figure 2 illustrates the DCC algorithm for p_i. For each boundary node p_i, if $p_i \in \mathbb{R}(\overline{p_e^l p_{i+1}})$, for some $l, 0 \leq l \leq |P_e|$, then p_i is the farthest (w.r.t. the distance the probing packet traveled) visible boundary node from p_e^l so far, since the next node p_{i+1} is not visible from p_e^l. So p_i is added to the set P_e, and the probing packet containing the set P_e is forwarded to the next boundary node (Line 6-8). However, if another boundary node $p_j, j > i+1$ that is visible from p_e^l is found, p_i is deleted from the set P_e, since p_i is no longer the farthest visible node from p_e^l (Line 1-5).

The above process is repeated until the probing packet either returns to the initiator, or reaches the end of the chain if

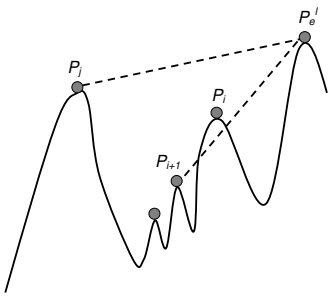


Fig. 2. An illustration of DCC.

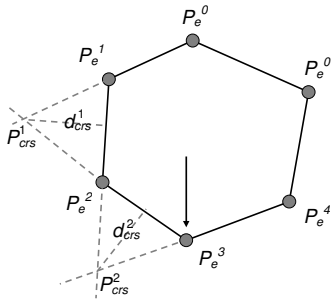


Fig. 3. An illustration of EPRA.

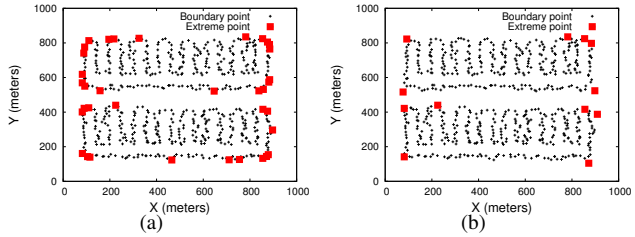


Fig. 4. (a) The "tight" set of extreme points; (b) reduced extreme points.

the cycle is not closed (code for p_{init}: Line 5-7). Because the DCC algorithm requires a single traversal of the probing packet, combined with the p_{init} election procedure, the message complexity is $O(n \log n)$, a better bound than the currently known one for a distributed convex hull algorithm on a ring topology [14].

The DCC algorithm generates a "tight" convex hull. This means that for a "smooth" hole, the DCC algorithm might generate many extreme points (i.e., if the hole is a perfect circle, all the boundary nodes will be selected as the extreme points). A large number of extreme points will degrade the system performance; thus, the number of extreme points must be controlled in real time, during the execution of the DCC algorithm. Thus, we develop the Extreme Points Reduction Algorithm (EPRA). EPRA limits the probing packet size by a user defined threshold. It does not incur additional communication overhead, because it operates as part of the DCC algorithm.

EPRA is embedded in the DCC algorithm as shown in Algorithm 2 (Line 8-14). Figure 3 illustrates the algorithm. Specifically, we define p_{crs}^i as the point at an intersection between the two lines $\overline{p_e^{i-1}p_e^i}$ and $\overline{p_e^{i+1}p_e^{i+2}}$, where $p_e^{-1} = p_e^{|P_e|}$. The Euclidean distance $d_{crs}^i = | \perp (p_{crs}^i, \overline{p_e^i p_e^{i+1}})|$, where $\perp (p, \overline{uv})$ is a line segment connecting p and p's projection on line $\overline{uv}$, is computed for each p_{crs}^i. When the probing packet reaches the point $p_i \in \mathbb{R}(\overrightarrow{p_e^{|P_e|}p_{i+1}})$, and the number of the extreme points found so far is greater than the predefined threshold (Line 9), corresponding p_{crs} values for the extreme points found so far are computed, and p_{crs}^i with the smallest d_{crs}^i is assigned as a new p_e^i, and p_e^{i+1} is removed (Line 10-12). This process is repeated as the probing packet traverses the boundary nodes. Figure 4(a) shows an example of the extreme points generated by DCC algorithm, and Figure 4(b) depicts

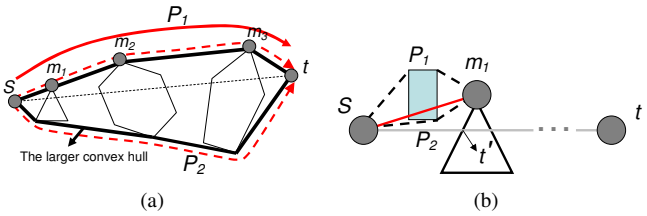


Fig. 5. An illustration of GOAL: (a) Step1-4; (b) Step5.

the results when DCC algorithm is integrated with EPRA.

C. Forwarding Algorithm

In this section, we present the details of our forwarding algorithm. When the hole abstraction process finishes, the nodes within h hops from a hole know about the locations of the extreme points of the convex hull covering the hole. Using this information, source s makes a routing decision following the steps described below:

Step1: Source s first identifies the interfering holes within h hops from it. If there is no interfering holes, source s forwards the packet using geographic forwarding.

Step2: A larger convex hull is constructed from a set of points, say $\mathcal{P}$, consisting of the extreme points of the interfering holes, source s, and destination t, as shown in Figure 5(a). The larger convex hull can be constructed by applying an existing centralized convex hull algorithm to the set $\mathcal{P}$ [15], because source s has all the locations of the extreme points for the interfering holes. The larger convex hull yields two possible paths, one along the upper part of the hull, P_1, and the other one along the lower part of the hull, P_2.

Step3: Source s selects a shorter path between P_1 and P_2. Source s then sets the extreme points along the selected path as the intermediate destinations. This step is depicted in Figure 5(a) with the intermediate destinations, m_1, m_2, and m_3.

Step4: Source s then checks if there exist any interfering holes for $\overrightarrow{sm_1}$. If there is no interfering hole, the packet is forwarded to m_1 using simple geographic forwarding. Otherwise, Step5 is executed, where new intermediate destinations are set up for $\overrightarrow{sm_1}$. Upon receiving the packet, m_1 becomes a new source s; and the algorithm reruns to reflect the new vision of m_1.

Step5: Step1 and Step2 are used to find the two possible paths connecting s and m_1. Figure 5(b) shows the possible two paths. And then, if the length of one path is longer than $|\overline{st'}| + |\overline{t'm_1}|$, the other one is selected. If both are shorter than $|\overline{st'}| + |\overline{t'm_1}|$, the path that is closer to line $\overline{sm_1}$ is chosen.

Here we explain that GOAL has low computational overhead. For Step1, each set of the extreme points P_e^i is scanned to check if any hole H_i within h hops intersects with the line segment $\overline{st}$. The computational complexity of Step 1 is thus $O(N_{ext})$, where N_{ext} is the total number of extreme nodes in the network. Step2 can be easily implemented using an existing centralized convex hull algorithm. We note that the best performance of currently known centralized convex hull algorithms is $O(N_{ext} \log N_{ext})$. The worse case happens when

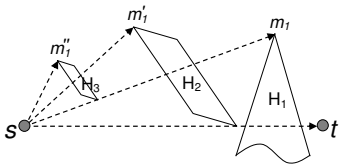
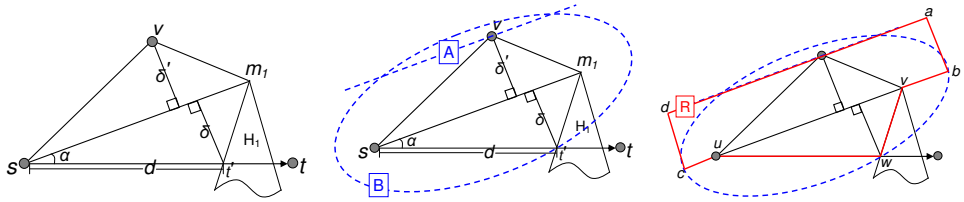


Fig. 6. An illustration of recursive runs of our routing algorithm.



(a) (b) (c)

Fig. 7. (a) Symbols for correctness proof; (b) bounding region representing possible locations for intermediate destinations; (c) final bounding region R;

all the holes interfere with the path $\overrightarrow{st}$, but such an extreme case rarely happens, making the average complexity of GOAL much lower than $O(N_{ext} \log N_{ext})$.

However, as shown in Figure 6, in Step4 and Step5, the algorithm might be iteratively repeated if there is an interfering hole, H_2, for path $\overrightarrow{sm_1}$, then another interfering hole, H_3, for path $\overrightarrow{sm'_1}$, and so on. In the following section, however, we will show that the number of iterations is bounded by a constant C. Consequently, the total computation complexity of GOAL for each node is $O(N_{ext} \log N_{ext})$.

D. Special Cases

In designing GOAL we need to consider special situations, where source s or destination t (or both) is located in a convex hull. To handle these scenarios, the cycle of boundary nodes is used to guide a packet to either leave the convex hull, or reach the destination in the convex hull, without relying on any graph planarization techniques. There are three cases to be considered:

Case 1, Source s is in a convex hull: source s computes the shortest path based on our GOAL routing protocol, and it sends the packet to the first intermediate destination. If source s has a clear path to the first intermediate node, the packet is routed to the first intermediate destination by using greedy routing. However, if the packet is blocked by a hole, then the packet would reach one of the boundary nodes. The packet then starts a counter clockwise traversal along the boundary nodes until greedy routing to the first intermediate node can be resumed.

Case 2, Destination t is in a convex hull: This case is similarly handled as the Case 1. The packet follows a set of the intermediate destinations previously determined by our routing protocol. After passing the last intermediate destination, the packet is greedily forwarded to destination t. If the last intermediate destination has a clear path to destination t, then routing can proceed. Otherwise, the packet would reach one of the boundary nodes. Then, the packet starts traversing the set of boundary nodes in a counter clockwise direction until greedy forwarding to destination t can be resumed.

Case 3, Both n_{src} and n_{dst} are in convex hulls: This case can be simply handled as a combination of the Case 1 and Case 2.

V. GOAL Protocol Analysis

A. Correctness of Convex Hull Construction

We first prove the correctness of the DCC algorithm. Specifically, we will show that given a hole, our algorithm finds all the extreme points of the tight convex hull covering the hole.

Lemma 1: p_{init} is an extreme point.

Proof: If the boundary nodes form a chain, the claim trivially holds. So, we consider only the case where the boundary nodes form a cycle. Assume, by contradiction, that p_{init} is not an extreme point. By definition, the y-coordinate of p_{init} is larger than any extreme point. Thus, p_{init} is not covered by the convex hull, which is a contradiction. Note that if there is an extreme point with the same y-coordinate as that of p_{init}, the x coordinates of all the extreme points are greater than p_{init}. Thus, p_{init} is not again covered by the convex hull, a contradiction. □

As described earlier, DCC algorithm searches for the farthest visible node from the last discovered extreme point. The following lemma shows that such farthest visible node is the next extreme point.

Lemma 2: Given an extreme point p_e^i, the farthest visible boundary node from p_e^i, say p_e^{i+1}, is the next extreme point.

Proof: Assume by contradiction that p_e^{i+1} is not the next extreme point, i.e., there exists an extreme point $(p_e^{i+1})'$ that is closer to p_e^i than p_e^{i+1}. If $(p_e^{i+1})' \in \mathbb{L}(\overline{p_e^i p_e^{i+1}})$, a hole is reshaped as a concave hull. If $(p_e^{i+1})' \in \mathbb{R}(\overline{p_e^i p_e^{i+1}})$, or if $(p_e^{i+1})'$ is on the line $\overline{p_e^i p_e^{i+1}}$, then p_e^{i+1} is not visible from p_e^i. □

Lemma 3: p_{init} is the farthest visible boundary node of p_e^n when $P_e = \{p_e^0, p_e^1, ..., p_e^n\}$.

Proof: Since the y-coordinate of p_{init} is the highest among all boundary nodes, and the x-coordinate of p_{init} is the lowest among all boundary nodes, any boundary nodes on p_{init}'s left hand side are not visible from p_e^n. Thus, p_{init} is a visible node that is the farthest from p_e^n. □

Theorem 1: Given a hole H_i, DCC finds all extreme points, say $P_e^i = \{p_e^0, p_e^1, ..., p_e^n\}$, of the tight convex hull covering the boundary nodes of H_i.

Proof: By Lemma 1, $p_e^0 = p_{init}$, and subsequent extreme points are determined by Lemma 2. Lastly, by Lemma 3, p_e^0 is the farthest visible node from p_e^n. Thus, connecting all p_e^i, $0 \leq i \leq n$, as p_e^0 - ... - p_e^n - p_e^0, we get a convex hull. Now we prove that there is no more extreme points. Assume in contradiction that there is one more extreme point in P_e. Without loss of generality, assume that a point p'_e is between two extreme points, p_e^i and p_e^{i+1} for some $i, 0 \leq i \leq |P_e|-1$. By Lemma 2, p_e^{i+1} is the farthest visible node of p_i. Consider the case where $p'_e \in \mathbb{L}(\overline{p_e^i p_e^{i+1}})$. In this case, the resulting polygon becomes concave. If $p'_e \in \mathbb{R}(\overline{p_e^i p_e^{i+1}})$ or p'_e is on the line $\overline{p_e^i p_e^{i+1}}$,

then, p_e^{i+1} is no longer visible from p_e^i. □

B. Correctness and Constant Stretch of GOAL

In this section we prove the correctness of GOAL, and show that a path generated by GOAL has a constant stretch. Consider path $\overrightarrow{sm_1}$, where m_1 is the intermediate destination on the upper hull P_1, and H_1 is the interfering hole for $\overrightarrow{st}$ as depicted in Figure 7(a). We first explain some symbols and their geometric properties, and introduce our main proof. Let α be the angle between the two line segments $\overline{sm_1}$ and $\overline{st}$. The range of α is $0 < \alpha < \pi$, because if $\alpha > \pi$, then m_1 would have been in $\mathbb{R}(\overline{st})$, being a point for a path P_2. d is the Euclidean distance to a hole along a line segment $\overline{st}$. d is smaller than $r_{max} \cdot h$, where r_{max} is the maximum radio range, and h is the number of hops to which the location of H_1 is broadcast. If $d > r_{max} \cdot h$, a node s would not have discovered the hole H_1. δ represents the maximum height of an interfering hole for $\overrightarrow{sm_1}$. Note that the height of interfering hole for $\overrightarrow{sm_1}$, δ', is smaller than δ, because if not, s will choose a path $s-t'-m_1$. δ can be expressed as $d\sin\alpha$, where $0 \le \alpha \le \pi$. A line segment $\overline{m_1t'}$ is the shortest among all possible connections between the two points t' and m_1 that change depending on the shape of the hole. $|\overline{m_1t'}|$ can be as long as $|\overline{t's}| + |\overline{sm_1}|$.

Theorem 2: GOAL is correct.

Proof: In order to prove guaranteed packet delivery, it suffices to show that a packet is successfully routed from s to the next intermediate destination, say m_1, since each time a packet arrives at the next intermediate destination m_1, m_1 becomes s, and runs the same algorithm from the Step 1. Therefore, once we prove a successful delivery for $\overrightarrow{sm_1}$ without any loops or arbitrarily long paths, a guaranteed delivery can be proved by induction.

The maximum height of an interfering hole for $\overrightarrow{sm_1}$ must be smaller than δ, because otherwise such a hole would have been detected as an interfering hole for $\overrightarrow{st}$ (Note $\delta' \le \delta$). Therefore, we obtain the upper bound for the possible positions of new intermediate destinations, which is depicted as a dotted line A in Figure 7(b). Next assume that a point v is the new intermediate point that belongs to the interfering hole for $\overrightarrow{sm_1}$. One observation is that $|\overline{sv}| + |\overline{vm_1}|$ must be smaller than $d + |\overline{t'm_1}|$, because if $|\overline{sv}| + |\overline{vm_1}| > d + |\overline{t'm_1}|$, then s would have selected a path $s \to t' \to m_1$. Therefore, such a point v must be bounded by an ellipse B having s and m_1 as foci and passing through a point t'. Considering the boundaries we computed and the range of an angle between $\overline{sm_1}$ and $\overline{sv}$ (0 to π), the possible locations for any new intermediate points are bounded by region R as shown in Figure 7(c). This region cannot be arbitrarily large, since δ is at most d which depends on the constant parameter h. □

Theorem 3: GOAL has constant stretch.

Proof: Without loss of generality, we represent our network as a Unit Disk Graph (UDG). More precisely, we adopt the k bounded degree unit disk graph where the degree of each node is bounded by k [16]. However, k bounded degree unit disk graph can be constructed from a general unit disk graph [3].

As shown in Theorem 2, for any pair of intermediate points u and v, including s and t, possible locations for new intermediate destinations due to an interfering hole for a path $\overrightarrow{uv}$ are bounded by some region R. By Kuhn [3], the total number of nodes N_R in region R is given by: $N_R \le (k+1)\frac{8}{\pi}(A(R) + p(R) + \pi)$, where $A(R)$ is the area of R, and $p(R)$ is the perimeter of R. $A(R)$ consists of a rectangle $\square abce$ in upper part of R, and a triangle $\triangle uvd$ in lower part of R. The area of the rectangle is $(d + |t'm_1|)d\sin\alpha \le (2d + |uv|)d\sin\alpha$, and the area of the triangle is $\frac{1}{2}|uv|d\sin\alpha$. The perimeter of R is $6d + 2d\sin\alpha + 2|uv|$. Thus, the total number of nodes in R is given by:

$$N_R \le (k+1)\frac{8}{\pi}\{(\frac{3}{2}d\sin\alpha+2)|uv|+2d\sin\alpha\cdot(d+1)+6d+\pi\}$$

$$\le (k+1)\frac{8}{\pi}\{(\frac{3}{2}d+2)|uv| + 2d^2 + 8d + \pi\}\ (0 < \alpha < \pi).$$

By the assumption of dense and uniform distribution of nodes and the property of greedy forwarding, a packet is forwarded outward from a point s at each step of the algorithm unless it hits the bounding region R. This implies that each node in region R is visited at most once. Thus, the total number of hops H_R in R is bounded by: $H_R \le N_R$.

Now consider all (u, v) pairs between $\overrightarrow{st}$, and assume that h is the maximum hop count of the network. The total number of hops from s to t, H_{st} is given by:

$$H_{st} \le \sum_{(u,v)\in\overrightarrow{st}} [(k+1)\frac{8}{\pi}\{(\frac{3}{2}d+2)|uv| + 2d^2 + 8d + \pi\}]$$

$$\le ((k+1)\frac{8}{\pi}\{(\frac{3}{2}d+2)\sum_{(u,v)\in\overrightarrow{st}} |uv| + 2d^2 + 8d + \pi\}.$$

, where $\sum_{(u,v)\in\overrightarrow{st}} |uv|$ is the shortest path in the Visibility Graph [17]. By [8], the shortest path between s and t in the Visibility Graph is bounded by some constant factor of Euclidean distance between s and t as the following: $\sum_{(u,v)\in\overrightarrow{st}} |uv| \le \frac{1}{1-\sin(\frac{\pi}{\epsilon})}|st|$. Therefore, we get:

$$H_{st} \le ((k+1)\frac{8}{\pi}\{(\frac{3}{2}d+2)\frac{1}{1-\sin(\frac{\pi}{\epsilon})}|st| + 2d^2 + 8d + \pi\}$$

, where $d \le r_{max}h$. □

C. Average Stretch of GOAL

We have shown that the worst case path stretch of GOAL is constant. Now we theoretically analyze the average path stretch of GOAL when the system parameter h is given as an input. In this analysis, we consider a $d \times d$ square region in which nodes are uniformly and densely deployed (i.e., a path between two nodes can be thought of as a line segment connecting the two). We assume that each node has circular communication range with radius 1, and the holes are abstracted as convex hulls.

We first consider the case with a single hole in the network. As shown in Figure 8(a), both the range of input h and the area of triangle $\triangle ABC$, which represents the deviation from

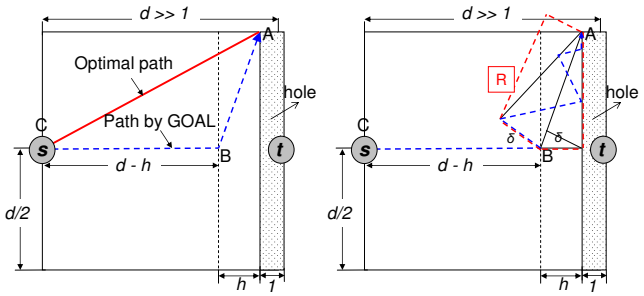


Fig. 8. (a) the single-hole case; (b) the multi-hole case.

the optimal path, are maximized when: i) source s is located in the middle of one side of the square; ii) destination t faces s from the midpoint of the opposite side of the square; and iii) the hole with width 1 is located along the side having t. The following lemma proves the average path stretch of GOAL for the single hole case.

Lemma 4: The average stretch λ of GOAL for the single hole case is $1 + \frac{1}{1+\sqrt{5}}$.

Proof: The path length of optimal path is $\sqrt{\frac{d^2}{4} + d^2} + \frac{d}{2}$, and the path length of GOAL is $(d-h) + \sqrt{\frac{d^2}{4} + h^2} + \frac{d}{2}$. Thus, the path stretch is $f(h) = \frac{(d-h)+\sqrt{\frac{d^2}{4}+h^2}+\frac{d}{2}}{\sqrt{\frac{d^2}{4}+d^2}+\frac{d}{2}}$, and the average path stretch λ for input h is given as the following:

$$\lambda = \frac{\sum_{h=1}^{d} f(h)}{d} \leq 1 + \frac{1}{1+\sqrt{5}}$$

, since $\frac{1}{(\sqrt{5}+1)d} << 1$ and $\sum_{h=1}^{d} \frac{\sqrt{\frac{d^2}{4}+h^2}+\frac{d}{2}}{\sqrt{\frac{d^2}{4}+d^2}+\frac{d}{2}} \leq d$. □

Now we investigate the average stretch for the multi-hole case. A key observation is that the multi-hole case can be considered as a series of the single-hole case for each interfering hole for $\overrightarrow{st}$ due to the following reason. A new routing decision is made when the packet reaches each intermediate destination of the hole, and if there are more than two interfering holes within h hops, they are considered as a single convex hull covering all the holes. Consider Figure 8(b). When a packet reaches a node at B, the node B knows about the hole and the packet is detoured to the intermediate destination at A. One difference from the single-hole case is that there might be another holes that interfere with the path from B to A. However, as proven in Theorem 2, the deviation of the path $\overrightarrow{BA}$ is bounded by the region R. Thus, the average stretch of path $\overrightarrow{BA}$, λ' becomes:

$$\lambda' = \frac{\sum_{h=1}^{d} \frac{(d-h)+\sqrt{\frac{d^2}{4}+h^2}+\sqrt{\frac{d^2}{4}+h^2}}{\sqrt{\frac{d^2}{4}+d^2}}}{d} \leq 2 + \frac{1}{\sqrt{5}} \approx 2.5.$$

Using this property, we obtain the following result.

Theorem 4: The average path stretch of GOAL is $2 + \frac{1}{\sqrt{5}}$.

Proof: Given source s and destination t, a set of interfering holes, $H_1, ..., H_n$, are identified. The path stretch from s to the intermediate destination of H_1, say H_1' is at most 2.5. Thus, the path length of $\overrightarrow{sH_1'}$ is $2.5d_1$, assuming that optimal path length is d_1. Now we consider the path $\overrightarrow{H_1't}$ and a hole H_2 as a single-hole case. Similarly, we obtain the path length of $\overrightarrow{H_1'H_2'}$ is $2.5d_2$, where d_2 is the optimal path length of $\overrightarrow{H_1'H_2'}$. If we repeat this process for all interfering holes, the total average path stretch for $\overrightarrow{st}$ is $\frac{2.5(d_1+\cdots+d_{n-1})+d_n}{d_1+\cdots+d_n} \leq 2.5$. □

VI. Simulation Results

We extended the original implementation of VIGOR [8] and conducted simulations to evaluate the performance of our protocol. We randomly deployed 3,000 nodes in a 1,000×1,000m^2 region. The locations and shapes of the holes in a network were predefined, and the nodes inside the holes are not considered. The communication range of the nodes was 30m, and the average node density was 9 per radio range. We compared our protocol GOAL with VIGOR, more specifically VIGOR-R. GOAL is also compared with the classic and widely used geographic routing protocol, GPSR, and the centralized shortest path routing. Specifically, 1,000 source and destination pairs were randomly selected among the nodes that are not in convex hulls. The following metrics were used: average hop stretch (path stretch is defined as the total hop count divided by the hop count of the shortest path routing), maximum hop stretch, protocol-related message overhead, and protocol related memory overhead. We varied the following parameters: h, the number of hops within which nodes receive the hole information, and b, the interval of broadcasting.

A. Hop Stretch

Two network scenarios were considered for this experiment: one with the two holes with high concavity shown in Figure 9(a), and the other one with many holes with low concavity as depicted in Figure 9(b). For this experiment, we set h to a sufficiently large number. (We will show how h affects the performance in the next subsection.) For both scenarios, GOAL outperforms GPSR. Especially for the holes with high concavity, GOAL shows dramatic improvements in hop stretch.

An observation is that VIGOR might generate a path with arbitrarily high path stretch. This happens because the *Direction Rule 3* in [8] might result in a "bad path". Figure 9(c) shows an example of such "bad path". A wrong routing decision is made at node p located at the arrow-marked position. Figure 10 depicts the schematic diagram of this example. We borrow some notations from [8]: u is a source; v is a destination; P^{CCW} and P^{CW} are the neighboring VON nodes in a counter clockwise and clockwise directions respectively. Because $\overrightarrow{pv}$ intersects with $\overline{P^{CCW}P^{CW}}$, *Direction Rule 3* is applied, and a packet is routed in a clockwise direction, because p is on the right side of $\overrightarrow{uv}$. As a result, the packet is routed back along the long boundary of the lower zig-zag hole until it reaches node q such that $\overline{qq_{next}}$ intersects with $\overline{pv}$.

Figure 11 shows the snapshot of the hop stretches for 300 s/t pairs. While GOAL achieves a stably low path stretch, VIGOR

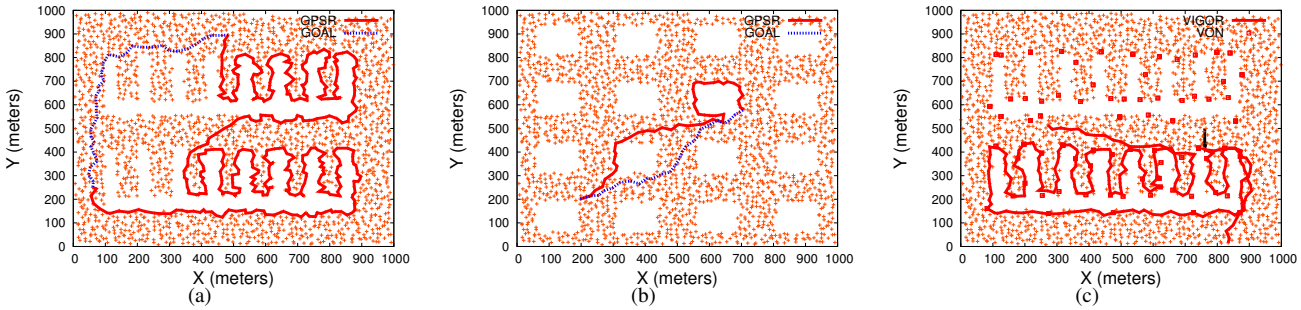


Fig. 9. (a) Scenario 1: two holes with high concavity;(b) Scenario 2: many holes with low concavity;(c) an example of bad path generated by VIGOR;

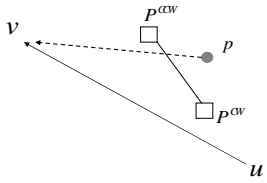


Fig. 10. Example of wrong routing decision in VIGOR.

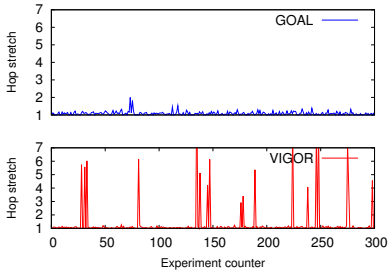


Fig. 11. Snapshot of hop stretches for initial 300 s-t pairs.

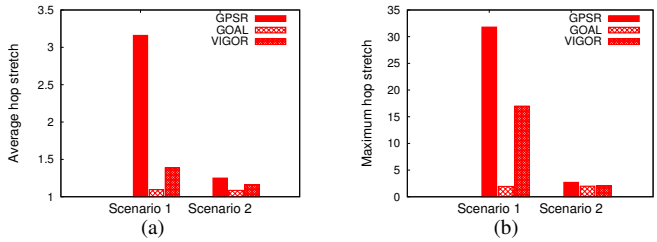


Fig. 12. (a) Average hop stretch. (b) Maximum hop stretch.

suffers from intermittently arising bad paths shown as the spikes in the lower graph despite its significantly lower path stretch. These bad paths affect the average performance of VIGOR as shown in Figure 12(a). In this figure, the GOAL shows the average path stretch close to 1 in both scenarios, as we set the h to a sufficiently large number, outperforming GPSR and VIGOR. However, we note that increasing the h value would lead to an increase in the routing table size. Figure 12(b) shows the maximum stretches of different protocols for different scenarios. The performance gain of GOAL is much higher in the maximum hop stretch.

B. Impact of h

The system parameter h determines how far the information on the hole is broadcast. Smaller h would decrease the message overhead. However, the average hop stretch would become higher for smaller h value. This is because, with smaller h, more nodes are unaware of the locations of holes. Thus, the decision to make a detour around a hole is more likely to be delayed, resulting in higher average hop stretch. Figure 13(a) depicts the result for Scenario 1, which confirms our expectations. As h increases, average hop stretch decreases as expected. For very small h values like 1 or 2, average stretch of GOAL is worse than VIGOR, but from $h = 4$, GOAL starts to outperform VIGOR.

C. Message Overhead

Similar to the experimental setting of [8], we adopted the update interval b, i.e., the extreme points are broadcast every b seconds. We assume that the beacon interval is 1 second. We measured the total number of messages sent per second, denoted by M, for VIGOR, GOAL with $h = 1$, and GOAL with $h = MAX$ (MAX is > 6 for Scenario 1). The results are depicted in Figure 13(b). First, it is obvious that larger update interval b reduces M. The graph also shows that more messages are sent for VIGOR than that for GOAL regardless of b or h. This is because VIGOR requires additional messages other than flooding the locations of VON nodes to run the distance vector algorithm; VON nodes exchange messages with each other along multiple hops to maintain routing tables. As noted in Section VI-B, smaller h permits smaller message overhead.

D. Memory Overhead

In VIGOR, each VON node maintains a routing table with the routing entries for all other VON nodes. Thus, additional memory requirement for VON node is $O(N_{von})$, where N_{von} is the number of VON nodes. Furthermore, each non-VON node has to maintain the *visibility set*. In order to compute the visibility set, each non-VON node has to know all the locations of the VON nodes; thus the space complexity for non-VON node is $O(V_{von})$. Compared with VIGOR, the memory overhead for GOAL is theoretically $O(V_{ext})$, because each node has to maintain the locations of all the extreme points when $h = MAX$. However, V_{ext} is typically smaller than V_{ext} due to more compact representation of a hole. Additionally, memory overhead for GOAL is adjustable by changing h.

Table I compares the memory overhead of GOAL and VIGOR, for the two scenarios. The size of required memory for GOAL was smaller than VIGOR in both scenarios. In particular, GOAL performed better in scenario 1 where there are less holes, but with high concavity. We also observe that smaller h reduces the memory overhead. One reason is that for smaller h, nodes maintain the states from smaller number of holes.

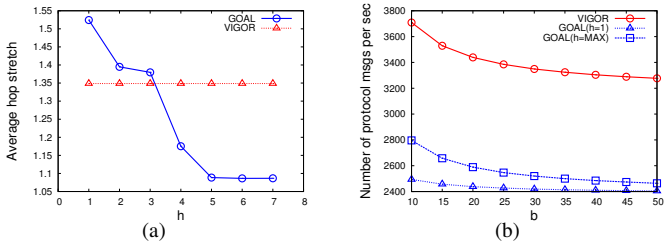


Fig. 13. (a) Impact of h. (b) Message overhead.

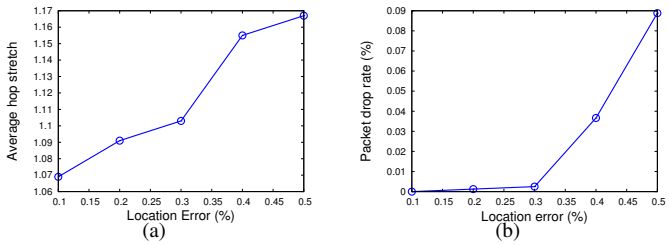


Fig. 14. Effect of location error: (a) average path stretch; (b) packet drop rate.

TABLE I
THE NUMBER OF ROUTING TABLE ENTRIES

	GOAL ($h = 1$)	GOAL ($h = MAX$)	VIGOR
Scenario 1	6	12	74
Scenario 2	6	64	80

E. Impact of Location Error

In this section, we investigate how the location error affects our protocol. We assume that the x and y coordinates of any node can vary at most $\pm$30m, which is the communication range. We measured the average path stretch and packet drop rate for different location error rates, specifically from 10% to 50 % of the maximum deviation. Figure 14(a) depicts our first results. As shown, the average path stretch increases as the location error increases. Next, Figure 14(b) shows the impact of location error on the packet drop rate. We observe that the packet drop rate is nearly 0 until location error is about 30%, then starts to rapidly increase.

VII. System Implementation

We implemented the DCC and EPRA algorithms on Epic motes running TinyOS 2.1.1. Experiments were performed in an indoor testbed consisting of 21 motes. Each mote is equipped with a CC2420 IEEE 802.15.4 wireless transceiver and MSP430 processor. The entire nesC implementation was $\sim$1,100 lines. Node locations are pre-programmed, and a hole is artificially created by turning off some of the nodes. Figure 15 depicts the results, where (x, y) represents the location of each node. As shown, the boundary nodes are identified and depicted as the chain of dotted lines. Our DCC algorithm discovers the extreme points that are represented as squares.

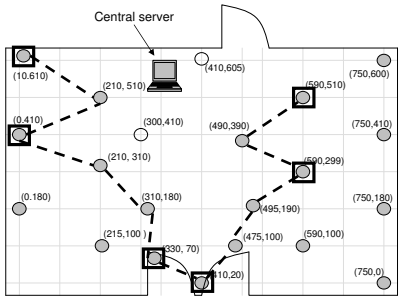


Fig. 15. System implementation and test-bed evaluation.

VIII. Conclusions and Future Work

In this paper, we present GOAL, a geographic routing protocol that achieves constant stretch with low overhead. A hole is compactly represented by leveraging our distributed convex hull construction algorithm. Our extreme points reduction algorithm further reduces the data size. This data representing holes enable each source node take an early detour around a hole, achieving a constant stretch. As future work, we will design a more efficient algorithm to route a packet to a node inside a convex hull. We also plan to develop a concept of different levels of convex hulls, called "isolines," to achieve an even distribution of energy consumption. Selecting the proper threshold for EPRA algorithm also remains as our future work.

References

[1] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris, "A scalable location service for geographic ad hoc routing," in *Proc. of ACM MOBICOM*, 2000.

[2] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "Ght: a geographic hash table for data-centric storage," in *Proc. of ACM WSNA*, 2002.

[3] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric ad-hoc routing: of theory and practice," in *Proc. of ACM PODC*, 2003.

[4] Z. Jiang, J. Ma, and W. Lou, "An information model for geographic greedy forwarding in wireless ad-hoc sensor networks," in *Proc. of IEEE INFOCOM*, 2008.

[5] Q. Fang, J. Gao, and L. J. Guibas, "Locating and bypassing holes in sensor networks," in *Proc. of IEEE INFOCOM*, 2004.

[6] C. Liu and J. Wu, "Destination-region-based local minimum aware geometric routing," in *Proc. of IEEE MASS*, 2007.

[7] N. Arad and Y. Shavitt, "Minimizing recovery state in geographic ad hoc routing," *IEEE Transactions on Mobile Computing*, vol. 8, 2009.

[8] G. Tan, M. Bertier, and A.-M. Kermarrec, "Visibility-graph-based shortest-path geographic routing in sensor networks," in *Proc. of IEEE INFOCOM*, 2009.

[9] Y. Mao, F. Wang, L. Qiu, S. Lam, and J. Smith, "S4: Small state and small stretch compact routing protocol for large static wireless networks," *Networking, IEEE/ACM Transactions on*, pp. 761 –774, june 2010.

[10] P. Li, G. Wang, J. Wu, and H.-C. Yang, "Hole reshaping routing in large-scale mobile ad-hoc networks," in *Proc. of IEEE GLOBECOM*, 2009.

[11] S. P. Fekete, A. Kröller, D. Pfisterer, S. Fischer, and C. Buschmann, "Neighborhood-based topology recognition in sensor networks," *Proc. of ALGOSENSORS*, vol. 3121, pp. 123–136, 2004.

[12] Y. Wang and J. Gao, "Boundary recognition in sensor networks by topological methods," in *Proc. of ACM MOBICOM*, 2006.

[13] H. Attiya and J. Welch, *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons, 2004.

[14] S. Rajsbaum and J. Urrutia, "Some problems distributed computational geometry," in *International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, 1999.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press, 2001.

[16] Y. Wang and X.-Y. Li, "Localized construction of bounded degree and planar spanner for wireless ad hoc networks," in *Proc. of DIALM-POMC*, 2003.

[17] K. Clarkson, "Approximation algorithms for shortest path motion planning," in *Proc. of ACM STOC*, 1987.