GOAL: A Parsimonious Geographic Routing Protocol for Large Scale Sensor Networks

Myounggyu Won, Wei Zhang and Radu Stoleru

Abstract—Geographic routing is well suited for large scale sensor networks, because its per node state is independent of the network size. However, due to the *local minimum* caused by holes/obstacles, the path stretch of geographic routing may degrade up to $O(c^2)$, where c is the path length of the optimal route. Recently, a geographic routing protocol based on the *visibility graph* (VIGOR) showed that a constant path stretch can be achieved. The constant path stretch, however, is achieved at the cost of communication and storage overhead, which makes the practical deployment of VIGOR in large scale sensor networks challenging. To this end, we propose GOAL (Geometric Routing using Abstracted Holes), a routing protocol that *provably achieves a constant path stretch, with lower communication and storage overhead.* To compactly describe holes, we develop a novel distributed convex hull algorithm, which improves the message complexity $O(n \log^2 n)$ of state of art distributed convex hull algorithm to $O(n \log n)$. The concise representation of a hole is used by nodes to make locally optimal routing decisions. Our theoretical analysis proves the correctness of the proposed algorithms and constant-stretch property of GOAL. Through extensive simulations and experiments on a testbed with 42 EPIC motes, we demonstrate the effectiveness of GOAL and its feasibility for resource constrained wireless sensor networks; specifically, we show that GOAL eliminates part of communication overhead of VIGOR and reduces the memory overhead of VIGOR by up to 51%.

Index Terms—wireless sensor networks, geographic routing protocols, path stretch

1 Introduction

Geographic routing protocols have attracted significant attention from the wireless sensor network (WSN) research community, because they are simple and scalable. In geographic routing, a source node obtains the location of a destination node from a location service [1], or through a hash-function in a data centric storage scheme [2]. A packet is then forwarded to the neighbor that is geographically closest to the destination. This greedy approach allows near-optimal path length in uniform and dense networks without obstacles (i.e., network holes). However, Kuhn et. al. [3] proved that, when holes are present, the path length may degrade up to $O(c^2)$, where c is the optimal path length, because of the "local minimum" phenomenon.

In order to bypass the local minimum and ultimately improve the path stretch, some geographic routing protocols use the non-local information on a hole (i.e., the size, location, and shape of a hole) [4][5][6][7]. In these protocols, holes are identified first. The size and shape of the identified holes are then propagated to a subset of nodes, so that these nodes use the information when they forward a packet to prevent the packet from ending up in a local minimum. However, unless the information about a hole is appropriately abstracted, such information can only be made known to a limited subset of nodes. For example, only the nodes located at the boundary of a hole called *boundary nodes*, or some of their neighbors receive such information. Thus, a "reaction" to the local minimum – for example, switching to the face routing mode – is only activated when a packet reaches a node at a local minimum called the *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, a hole is represented as a polygon, which is used to build a *visibility graph*, a structure often used in computational geometry to find the shortest path between a source and a destination, given obstacles of polygonal shapes. However, VIGOR suffers from non-negligible protocol-related communication overhead. First, in order to build the visibility graph, the locations of all Visibility-based Overlay Network (VON) nodes (the vertices of the polygons) must be flooded to all nodes in the network. Second, the VON nodes must iteratively exchange messages among themselves until a complete routing table is constructed. Third, for each source/destination pair, the source node must send a control packet to the destination using a default routing protocol (e.g., GPSR [9]) to find the entry point among VON nodes, before it begins data transmission. Furthermore, each node in a network must store the locations of all VON nodes in a network, incurring the storage overhead.

In this article, we propose a geographic routing protocol that achieves a constant path stretch, while signifi-

• *M. Won, W. Zhang and R. Stoleru are with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77840.*
E-mail: {mgwon, stoleru}@cse.tamu.edu, charleyhuman@gmail.com

cantly reducing the communication and storage overhead of VIGOR. In our protocol, a hole is compactly described as a set of extreme points of the convex hull covering the boundary nodes of the hole. A novel distributed convex hull algorithm is introduced to build the convex hull of a hole. To the best of knowledge, this distributed algorithm improves the message complexity of the state-of-art distributed convex hull algorithm [10][11], from $O(n \log^2 n)$ to $O(n \log n)$. A distributed extreme points reduction algorithm further reduces the size of the set of extreme points. Consequently, the locations of only a few extreme points of a convex hull are locally broadcast to nodes within h-hops from the hole. Based on this information on nearby convex hulls, a source node identifies the interfering holes within h-hops that block the straight path to the destination. A source node then computes a set of intermediate destinations that guide a packet along the locally optimal path. When a packet reaches an intermediate destination, the set of previous intermediate destinations are updated iteratively, improving the routing path. A route to and from the nodes inside a convex hull is also handled efficiently, for guaranteed packet delivery. The contributions of this article are as follows:

- We develop a geographic routing protocol that generates a path with constant stretch in networks with holes. The protocol has lower communication and storage overhead compared with the state-of-the-art protocol.
- We develop a distributed convex hull algorithm to efficiently reduce the size of data that describes a hole. To the best of our knowledge, this distributed convex hull algorithm has smaller communication overhead, when compared with the state-of-art algorithm.
- We present a thorough analysis to prove the correctness and constant path stretch property of our routing protocol.
- We perform extensive simulations and experiments on real motes to confirm the effectiveness of our protocol, and its feasibility for practical deployment in resource constrained WSN.

2 Related Work

Routing protocols for large scale WSNs can be largely categorized into geographic routing protocols and hierarchical routing protocols [12]. Recently, Mao et. al. [12] proposed S4, a novel hierarchical routing protocol, that achieves the worst-case stretch of 3 with small per-node state of $O(\sqrt{N})$, where N is the total number of nodes. Although S4 attains the desirable balance between the path stretch and size of per-node state, S4 fails to eliminate the dependence between the per-node state and network size. In contrast, geographic routing suits particularly well for resource constrained large-scale WSNs, because protocol's state (i.e., the locations of its immediate neighbors) is independent of the network size. However, the path stretch of geographic routing may degrade up to $O(c^2)$, where c is the optimal path length, due to the local minimum caused by topological complexities like holes. Little was known about achieving a constant path stretch for geographic routing protocols.

Several geographic routing protocols have been proposed to improve the path stretch, by identifying the forwarding nodes that may lead to a local minimum. In [5], a node uses the TENT rule to test whether it is a stuck node or not. If a node determines that it is a stuck node, it initiates the BOUNDHOLE algorithm to build a routing path surrounding 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, thereby eliminating the need for implementing face routing. However, the "late reaction problem" (i.e., a detour is made at the boundary node) degrades the path stretch. Arad et. al. [7] use the angle between two adjacent neighbors to identify the nodes that may forward a packet to a local minimum. Nodes compare the angle with a predefined threshold; if the angle is greater than the threshold, the node is "elevated" so that this node is avoided by the greedy forwarding process. However, the decision on whether a node is at a local minimum or not depends on the source and destination locations. 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 worst case path stretch guarantee.

Several researchers proposed to propagate the information on a hole (i.e., the size, shape, and location of a hole) to a subset of nodes in a limited region. In [4], a stuck node and its neighbors form an unsafe area of rectangular shape. The estimated shape of a hole is known to the nodes in this unsafe area. This distributed information model is used by forwarding nodes to avoid the local minimum. Although the propagation of the partial information on a hole helps improve the path stretch, the "late reaction" problem persists, preventing the protocol from providing guaranteed stretch. In [13], a hole is represented as an ellipse, and the abstracted information on a hole is broadcast to nodes within h-hops from the boundary of the ellipse. However, the ellipse often fails to represent all hole shapes. Both protocols [4][13] contribute to the reduction of path stretch, but they both fail to provide a guaranteed constant path stretch.

Recently, Tan et. al. [8] introduced VIGOR, a geographic routing protocol that guarantees 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 holes. One notable aspect of VIGOR, when compared with S4, is that its per node state is $O(N_{von})$, where N_{von} is the total number of VON nodes, thereby eliminating the dependency of the per node state on the network size. However, VIGOR suffers from communication and storage overhead, which makes the practical deployment of the protocol in large scale sensor networks difficult. First, the locations of all VON nodes must be known to all nodes in a network; Second, each VON node iteratively exchanges a message containing its routing table with neighboring VON nodes until the routing table converges; Third, for each source/destination pair, the source node must send a probing packet to the destination by using default routing protocol (e.g., GPSR), before actual data transmission begins.

In [14], we proposed a geographic routing protocol that eliminates the communication overhead for constructing a routing table and for setting up a routing path, while retaining the constant path stretch property. A novel distributed convex hull construction algorithm abstracts the information on a hole into a set of extreme points of the boundary nodes of the hole; the abstracted information is used by forwarding nodes to make a routing decision that leads to a path with guaranteed stretch. This article extends our previous work. First, in a completely new set of simulations we use a more realistic radio model [15][16], to more accurately estimate path stretches for different routing protocols. Second, simulation scenarios are more systematically designed. We propose a new metric describing the "complexity" of holes in the network, and we design simulation scenarios in an increasing order of this new metric. Third, we implemented our proposed routing protocol on mote hardware and performed experiments on a testbed consisting of 42 EPIC motes. In addition, in Section 7.3, we identify a practical issue for geographic routing protocols that achieve a constant stretch. Specifically, the delivery ratio for such protocols may significantly degrade when data rate is relatively high, because such routing protocols rely on a single best path. Thus, this article introduces a new research problem called *constant path stretch multipath routing*, where, depending on the data rate, the routing protocol must offer multiple paths, each ensuring a guaranteed path stretch.

3 PRELIMINARIES

In this section, we define the notations and terms used throughout this article. Figure 1 illustrates these notations and terms. We consider a dense wireless sensor network, consisting of N nodes, denoted by a set $V = \{v_1, v_2, ..., v_N\}$, uniformly distributed in a two dimensional space. We assume that each node knows its location, which is given as two-dimensional coordinates, say (x, y). We further assume that the coordinates (x, y) for each node are unique; that is, we assume that multiple nodes cannot be deployed at the exactly same 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*, denoted by $P^i = \{p_1, p_2, ..., p_n\}$, $p_j \in V$. 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 outer boundary of a network. We define one boundary node in each set P^i, called the *initiator*, as follows:

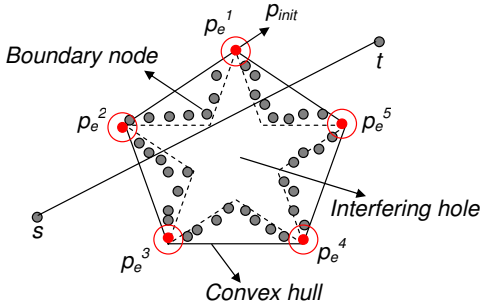


Fig. 1. Illustration of notations and terms.

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

A hole H_i is represented as a set of extreme points of the convex hull covering the set of boundary nodes P^i; this set of extreme points is denoted by $P_e^i = \{p_e^1, p_e^2, ..., p_e^n\}$ (i.e., $P_e^i \subseteq P^i$), where the extreme point p_e^j is defined as follows (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 (i.e., as the number of hops) between s and t is denoted by $|\overrightarrow{st}|$. A set of useful notations are as follows:

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

Given a path $\overrightarrow{st}$, the *interfering holes* are the holes that intersect with line segment $\overline{st}$. The interfering hole can be formally defined as follows:

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, we define that two nodes are *visible* to each others iff there are no interfering holes between them.

4 GOAL: GEOMETRIC ROUTING USING ABSTRACTED HOLES

This section provides the details of the proposed geographic routing protocol. Starting from the overview of the protocol, the following subsections describe the details of each component of the protocol.

4.1 Protocol Overview

GOAL consists of two main components: hole abstraction and packet forwarding. The objective of the hole abstraction process is to concisely represent holes in a network. In this process, a hole is represented as the convex hull of the boundary nodes surrounding the hole. The hole abstraction process is comprised of three phases. In the first phase, boundary nodes surrounding a hole are identified. The second phase constructs the convex hull of the boundary nodes for each hole; specifically, this phase finds a set of extreme points from the boundary nodes by using a probing packet that makes a single traversal along the boundary of a hole. In the last phase, the locations of extreme points are broadcast to nodes within h-hops from each hole.

When the hole abstraction process finishes, each node knows the locations of extreme points for holes within h-hops from it. Nodes use these locations to make a routing decision. To be more specific, a source node identifies interfering holes and runs our forwarding algorithm to find an intermediate destination among all the extreme points that belong to the interfering holes. The source node then sends a packet to the intermediate destination; upon receiving the packet, the node at the intermediate destination runs the same forwarding algorithm to determine the next intermediate destination. This process is repeated until the packet reaches a destination. The details of the two components are presented in the following subsections.

4.2 DCC: Distributed Convex Hull Construction

This section describes the details of the hole abstraction process. We adopt the boundary node detection scheme used in VIGOR [8] for implementing the first phase of this process. This boundary node detection scheme, when it is done, allows each boundary node to have the notion of a left and right neighboring boundary node. Once boundary nodes are found, we select the *initiator* among the boundary nodes for each hole. If boundary nodes form a 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 boundary nodes [17]. This leader election algorithm is based on two assumptions: (1) each node has unique ID; and (2) each node has the notion of a left and right neighbor. The first assumption corresponds to the unique coordinates of each node. The second assumption is satisfied by the boundary node detection scheme. If 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 coordinates to 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}.

Once p_{init} is elected, p_{init} starts 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^1, 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 (Line 2-3). Upon receiving

Algorithm 1 DCC (code for p_{init})

1: **if** hop_count = 0 **then**
2: $P_e \leftarrow P_e \cup \{p_{init}\}$
3: send a probing packet in counter-clockwise.
4: **else**
5: **if** $|N_{p_i}| = 1$ or $p_{init} \in \mathbb{R}(\overrightarrow{p_e^{|P_e|} p_1})$ **then**
6: terminate.
7: **end if**
8: **end if**

Algorithm 2 DCC (code for p_i)

1: **for** each $p_e^l \in P_e$, $1 \leq l \leq |P_e|$ **do**
2: **if** $\forall m$, $l+1 \leq m \leq |P_e|$, $p_e^m \in \mathbb{L}(\overrightarrow{p_e^l p_i})$ **then**
3: $P_e \leftarrow P_e \setminus \{p_e^{l+1}, p_e^{l+2}, ..., p_e^{|P_e|}\}$
4: **end if**
5: **end for**
6: **if** $p_i \in \mathbb{R}(\overrightarrow{p_e^{|P_e|} p_{i+1}})$ **then**
7: $P_e \leftarrow P_e \cup \{p_i\}$
8: // EPRA
9: **if** $|P_e| >$ Threshold **then**
10: find p_{crs}^i with minimum d_{crs}^i
11: $p_e^i \leftarrow p_{crs}^i$
12: $P_e \leftarrow P_e \setminus p_e^{i+1}$
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**

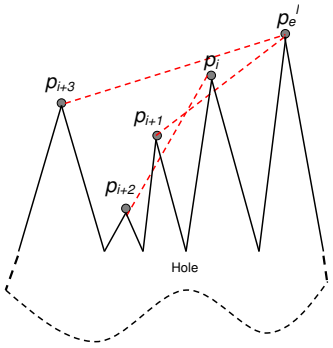


Fig. 2. Illustration of DCC.

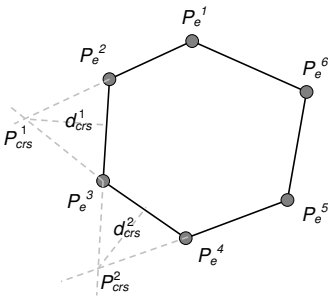


Fig. 3. Illustration of EPRA.

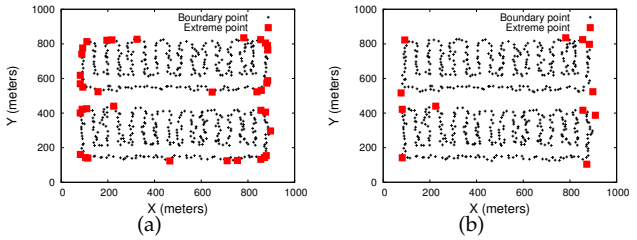


Fig. 4. (a) The "tight" set of extreme points; (b) reduced extreme points.

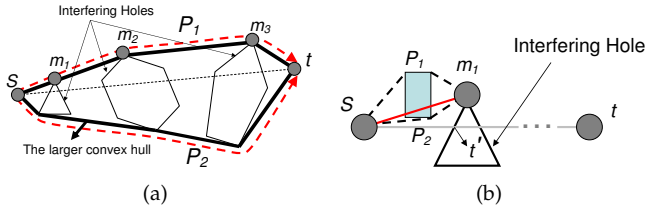


Fig. 5. Illustration of GOAL: (a) Steps 1-4; (b) Step 5.

the packet, a boundary node p_i examines whether it is an extreme point, by executing the code for p_i depicted in Algorithm 2.

Figure 2 illustrates an example describing how extreme points are identified. For each boundary node p_i, if $p_i \in \mathbb{R}(\overrightarrow{p_e^l p_{i+1}})$, where p_e^l is the most recently selected extreme point, then p_i is the farthest (w.r.t. the distance the probing packet traveled) visible boundary node from p_e^l so far, because next boundary node p_{i+1} is not visible from p_e^l; thus, p_i is selected as the next extreme point and added to the set P_e; p_i then forwards the probing packet containing the set P_e to the next boundary node p_{i+1} (Line 6-8). Note that, at this point, P_e is $\{p_e^l, p_i\}$, and the most recent extreme point $p_e^{|P_e|}$ is p_i. Similarly, p_{i+1} is added to the set P_e, because $p_{i+1} \in \mathbb{R}(\overrightarrow{p_i p_{i+2}})$ However, p_{i+2} is not an extreme point, because $p_{i+2} \notin \mathbb{R}(\overrightarrow{p_{i+1} p_{i+3}})$. Thus, so far, $P_e = \{p_e^l, p_i, p_{i+1}\}$. Each time the DCC algorithm checks whether a given boundary node is an extreme point (i.e., the farthest visible node from the most recently selected extreme point), the DCC algorithm also ensures that the set of extreme points P_e is updated when the current boundary node is farthest from any existing extreme point in the set P_e. For example, if a boundary node p_{i+3} that is visible from p_e^l is found, all previous extreme points identified after p_e^l (i.e., p_i and p_{i+1}) are deleted from the set P_e (Line 1-5).

The above process is repeated until the probing packet either returns to the initiator, or reaches the end of the chain (code for p_{init}: Line 5-7). The DCC algorithm requires a single traversal of the probing packet; thus, together with the p_{init} election procedure, the message complexity is $O(n \log n)$, a better bound than the message complexity $O(n \log^2 n)$ of the currently known distributed convex hull algorithm on a ring topology [10][11].

The DCC algorithm generates a "tight" convex hull. This means that for a "smooth" hole, the DCC algorithm might generate many extreme points (e.g., if the hole is a perfect circle, all the boundary nodes will be selected as extreme points). A large number of extreme points will degrade the system performance; thus, in some cases, the number of extreme points must be controlled. To this end, we develop the Extreme Points Reduction Algorithm (EPRA). EPRA limits the total number of extreme points 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 an example that shows how EPRA works. We define p_{crs}^i as the intersection of two lines $\overline{p_e^i p_e^{i+1}}$ and $\overline{p_e^{i+2} p_e^{i+3}}$, where $1 \leq i \leq |P_e|$, and $p_e^{|P_e|+1} = p_e^1$, $p_e^{|P_e|+2} = p_e^2, \ldots$. The Euclidean distance $|\perp (p_{crs}^i, \overline{p_e^{i+1} p_e^{i+2}})|$ denoted by d_{crs}^i, 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 point $p_j \in \mathbb{R}(\overrightarrow{p_e^{|P_e|} p_{j+1}})$, a candidate for an extreme point, and the number of extreme points found so far is greater than the predefined threshold (Line 9), the p_{crs}^i values for previously found extreme points are computed; and p_{crs}^i with the smallest corresponding d_{crs}^i value is assigned as a new extreme point; and existing extreme points p_e^{i+1}, and p_e^{i+2} are removed from P_e (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 the DCC algorithm, and Figure 4(b) depicts the results when DCC algorithm is integrated with EPRA.

When the DCC algorithm completes, the *initiator* for a hole has the locations of all extreme points for the hole. The initiator then broadcasts these locations to nodes within h-hops from the hole.

4.3 Forwarding Algorithm

When the hole abstraction process finishes, the locations of extreme points are made known to nodes within h-

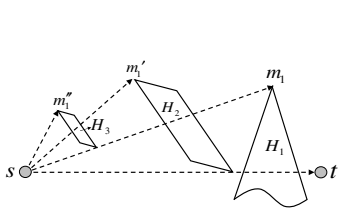


Fig. 6. Illustration of recursive runs of our routing algorithm.

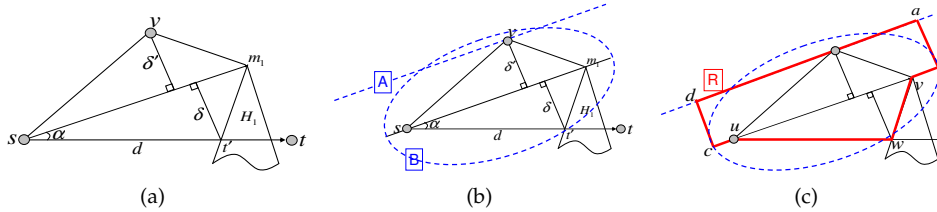


Fig. 7. (a) Symbols for correctness proof; (b) bounding region representing possible locations for intermediate destinations; (c) final bounding region R.

hops from a hole. Using this information, source node s makes a routing decision by following the steps described below:

Step 1: Source s identifies *interfering holes* based on the locations of received extreme points, source s, and destination t. If there is no interfering hole, source s forwards a packet using a geographic forwarding algorithm.

Step 2: If there are interfering holes, source s computes a "larger convex hull" of the set of points $\mathcal{P}$ consisting of the extreme points of the interfering holes, source s, and destination t. Figure 5(a) shows an example of such "larger convex hull". This new convex hull can be easily constructed by applying an existing centralized convex hull algorithm to the set $\mathcal{P}$ [18]. Source s then considers two possible paths: one along the upper part of the hull, denoted by P_1 in Figure 5(a), and the other path along the lower part of the hull, denoted by P_2.

Step 3: Source s selects the shorter path between P_1 and P_2. Source s then uses the extreme points along the selected path as the intermediate destinations. This step is depicted in Figure 5(a) with the intermediate destinations, denoted by m_1, m_2, and m_3.

Step 4: In this step, source s checks whether there exist interfering holes for path $\overrightarrow{sm_1}$. If there is no interfering hole, the packet is forwarded to m_1 using simple geographic forwarding. Otherwise, Step 5 is executed, where new intermediate destinations are found for path $\overrightarrow{sm_1}$. Upon receiving the packet, m_1 becomes a new source s; and the forwarding algorithm reruns from Step 1, to reflect the new vision of m_1.

Step 5: Source s first uses Steps 1 and 2 to find the two possible paths connecting s and m_1. Figure 5(b) shows such paths. If the length of one path is longer than $|\overline{st'}| + |\overline{t'm_1}|$, source s selects the other path. If both paths are shorter than $|\overline{st'}| + |\overline{t'm_1}|$, the path that is closer to line $\overline{sm_1}$ is chosen. Source s then sends a packet to m_1 using Step 4.

Now we show that GOAL has low computational overhead to justify the feasibility for practical deployment of GOAL. For Step 1, each set of extreme points P_e^i is scanned to check whether any hole H_i within h-hops intersects with line segment $\overline{st}$. The computational complexity of Step 1 is thus $O(N_{ext})$, where N_{ext} is the total number of extreme points in the network. Step 2 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 worst case happens when all holes interfere with path $\overrightarrow{st}$. Such an extreme case rarely happens, making the average complexity of GOAL lower than $O(N_{ext} \log N_{ext})$.

However, as shown in Figure 6, Steps 4 and 5 of the forwarding algorithm might be recursively run 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 Section 5.2, we will show that the number of such iterations is bounded by a constant C. Consequently, the computational complexity of GOAL for each node is $O(N_{ext} \log N_{ext})$.

5 GOAL Protocol Analysis

5.1 Correctness of Convex Hull Construction

This section proves the correctness of the DCC algorithm; that is, we shall show that given a hole (i.e., a set of boundary nodes), our algorithm finds all extreme points that belong to the convex hull of the hole's boundary nodes. We first show that the initiator node is an extreme point.

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

Proof: If boundary nodes form a chain, the claim trivially holds. So, we consider only the case where 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 other extreme points. 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 p_{init}, the x coordinates of all extreme points are greater than p_{init}. Thus, p_{init} is not covered by the convex hull, a contradiction. □

As described in Section 4.2, the 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; that is, there exists an extreme point $(p_e^{i+1})'$

that is closer to p_e^i than p_e^{i+1}. If $(p_e^{i+1})' \in \mathbb{L}(\overrightarrow{p_e^i p_e^{i+1}})$, a hole is reshaped as a concave hull. If $(p_e^{i+1})' \in \mathbb{R}(\overrightarrow{p_e^i p_e^{i+1}})$, 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. □

The following lemma shows that the farthest visible node for the last extreme point is the initiator, creating a cycle of extreme points.

Lemma 3: p_{init} is the farthest visible boundary node of p_e^n when $P_e = \{p_e^1, p_e^2, ..., p_e^n\}$.

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

Now we are ready for the correctness proof.

Theorem 1: Given a hole H_i, DCC finds all extreme points, say $P_e^i = \{p_e^1, p_e^2, ..., p_e^n\}$, of the convex hull covering the boundary nodes of H_i.

Proof: By Lemma 1, $p_e^1 = p_{init}$, and subsequent extreme points are determined by Lemma 2. Lastly, by Lemma 3, p_e^1 is the farthest visible node from p_e^n. Thus, connecting all p_e^i, $1 \le i \le n$ (i.e., p_e^1 - ... - p_e^n - p_e^1), we get a convex hull. Now we prove that there are no more extreme points. Assume by 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 $i, 1 \le i \le |P_e|-1$. By Lemma 2, p_e^{i+1} is the farthest visible node of p_i. Consider the case where $p_e' \in \mathbb{L}(\overrightarrow{p_e^i p_e^{i+1}})$. In this case, the resulting polygon becomes concave. If p_e' $\in \mathbb{R}(\overrightarrow{p_e^i p_e^{i+1}})$ 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. □

5.2 Correctness and Constant Path Stretch of GOAL

In this section, we prove the correctness of GOAL, and show that a path generated by GOAL has a constant path stretch. Consider path $\overrightarrow{sm_1}$ in Figure 7(a), where m_1 is the first intermediate destination for path P_1 on the upper hull of the "larger convex hull" (recall Figure 5(a)), and H_1 is an interfering hole for path $\overrightarrow{st}$. Before we present our main proof, we first define some symbols and their geometric properties. Let α be the angle between two line segments $\overline{sm_1}$ and $\overline{st}$. The range of α is $0 \le \alpha < \pi$, because if $\alpha > \pi$, then m_1 would have been in $\mathbb{R}(\overrightarrow{st})$, being a point for path P_2, a path on the lower hull of the "larger convex hull". The term d refers to the Euclidean distance from source s to the first interfering hole (i.e., the length of line segment $\overline{st'}$). d is smaller than $r \cdot h$, where r is the communication radius of a node, and h is the number of hops within which the abstracted information about hole H_1 is broadcast. δ represents the maximum height of an interfering hole for $\overrightarrow{sm_1}$. Note that the height of an interfering hole for $\overrightarrow{sm_1}$, denoted by 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 $0 \le \alpha \le \pi$.

Theorem 2: GOAL is correct.

Proof: In order to prove guaranteed packet delivery, it suffices to show that a packet is successfully routed from source s to its first intermediate destination m_1, because when a packet arrives at the first intermediate destination m_1, m_1 becomes source s; and s applies the same forwarding algorithm from Step 1 to forward a packet to its next intermediate destination. Therefore, if we prove a successful delivery from s to m_1, without loops or arbitrarily long paths, a guaranteed delivery can be proved by induction.

The height of an interfering hole δ' for path $\overrightarrow{sm_1}$ must be smaller than δ, because otherwise such a hole would have been detected as an interfering hole for path $\overrightarrow{st}$ (i.e., $\delta' \le \delta$). Therefore, we obtain the upper bound for the possible positions of a new intermediate destination, which is depicted as a dotted line A in Figure 7(b). Next let point v be the new intermediate destination 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 $s \to t' \to m_1$. Thus, the possible locations of a new intermediate destination (i.e., the location of point v) must be bounded by an ellipse denoted by B having s and m_1 as foci and passing through point t'. Considering the two boundaries we computed and the range of the angle between $\overline{sm_1}$ and $\overline{sv}$ (i.e., 0 to π), the possible locations of a new intermediate destination 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 [19]. 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 source s and destination t, possible locations for an intermediate destination for path $\overrightarrow{uv}$ are bounded by some region R. According to Kuhn et. al. [3], the total number of nodes N_R in region R is bounded by $(k+1)\frac{8}{\pi}(A(R) + p(R) + \pi)$, where $A(R)$ is the area of region R, and $p(R)$ is the perimeter of region R. Thus, the total number of nodes in R is bounded as follows:

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

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

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

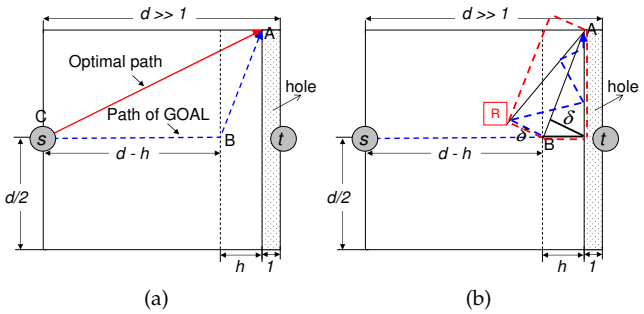


Fig. 8. (a) the single-hole case; (b) the multi-hole case.

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. 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 N_R (i.e., $H_R \leq N_R$).

Now consider all (u, v) pairs between $\overrightarrow{st}$, and assume that the system parameter h is chosen as the maximum hop count of the network so that all nodes in a network know the locations of extreme points. The total number of hops from s to t, H_{st} is then given as follows:

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

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

where $\sum_{(u,v)\in\overrightarrow{st}} |uv|$ is the shortest path in the Visibility Graph [20]. 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: $\sum_{(u,v)\in\overrightarrow{st}} |uv| \leq \frac{1}{1-\sin(\frac{\pi}{\epsilon})}|st|$. Therefore, we get:

$$H_{st} \leq C_1|st| + C_2.$$

$$C_1 = (k+1)\frac{8}{\pi}(\frac{3}{2}r \cdot h + 2)\frac{1}{1-\sin(\frac{\pi}{\epsilon})}.$$

$$C_2 = 2(r \cdot h)^2 + 8(r \cdot h) + \pi.$$

□

5.3 Average Path Stretch of GOAL

We showed 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 can vary. 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 nodes). This analysis for a network region of square shape can be easily extended to a rectangular-shaped network; and arbitrarily shaped network region can be approximated by a rectangular shape. We assume that each node has circular communication range with radius 1, and holes are abstracted as convex hulls.

We first consider the case where there is a single hole in a network. As shown in Figure 8(a), the area of triangle $\triangle ABC$, which represents the degree of deviation from the optimal path, are maximized when: i) source s is located in the middle of one side of the square; ii) destination t is located in the middle of other side of the square that faces the side that has s; and iii) the hole with width 1 is located along the side that has 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 follows:

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

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

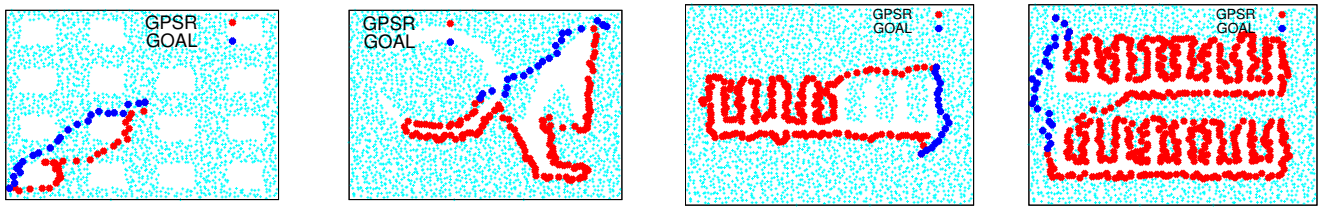
Now we investigate the average stretch for the multi-hole case. A key observation is that the multi-hole case can be considered as a series of single-hole cases for each interfering hole for $\overrightarrow{st}$ for the following reasons: (1) each intermediate destination makes a new routing decision by rerunning the forwarding algorithm, and (2) 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 packet is detoured to the intermediate destination at A. One difference from the single-hole case is that there might be other 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, assume that s visits n intermediate destinations, denoted by $i_1, ..., i_n$. The average path stretch from s to the first intermediate destination i_1 is at most 2.5. Thus, the path length of $\overrightarrow{si_1}$ is



(a) (b) (c) (d)

Fig. 9. Different hole deployment schemes: (a) hole Scenario 1; (b) hole Scenario 2; (c) hole Scenario 3; and (d) hole Scenario 4; These hole schemes are arranged in an increasing order of the concavity of deployed holes.

$2.5d_1$, assuming that optimal path length is d_1. Now we consider the path $\overrightarrow{i_1 t}$ and the possible interfering holes for the path as a single-hole case. We similarly find that the path length of $\overrightarrow{i_1 i_2}$ is at most $2.5d_2$, where d_2 is the optimal path length of $\overrightarrow{i_1 i_2}$. If we repeat this process for all the remaining intermediate destinations, the total average path stretch for $\overrightarrow{st}$ is $\frac{2.5(d_1+\cdots+d_{n-1})+d_n}{d_1+\cdots+d_n} \leq 2.5$. □

5.4 Discussion

It is important to remark that the constant and average path stretch results we have proved for GOAL are when source and destination nodes are outside convex hulls, which is precisely what our motivating application requires. More precisely, we are developing a disaster management application [21] consisting of WSN, adhoc and delay tolerant networks. In our application, and many others, holes typically have "regular" shapes, thus having tight convex hulls. This results in very few nodes falling inside convex hulls. In order to handle the source/destination node inside a convex hull, our GOAL protocol uses the cycle of boundary nodes to guide a packet to either leave a convex hull, or reach the destination inside a convex hull (it is important to remark that the constant and average path stretch bounds do not apply to this special situation). There are three cases GOAL considers:

Case 1: Source s is inside a convex hull. Source s computes the shortest path based on the 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 forwarding. 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. Upon receiving the packet, the first intermediate destination resumes the GOAL routing.

Case 2: Destination t is inside a convex hull. This case is similarly handled as Case 1. A packet is forwarded along a set of intermediate destinations previously determined by our routing protocol. Upon reaching the last intermediate destination, the packet is greedily forwarded to destination t. If the last intermediate destination has a clear path to destination t, geographic forwarding is sufficient for the packet to reach the destination. 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 source s and destination t are inside convex hulls. This case can be simply handled as a combination of Case 1 and Case 2.

An idea for handling all possible source/destination pairs in a cohesive manner (i.e., including source/destination inside convex hulls), which we are currently exploring, is to use visibility-graph-based routing technique, i.e., VIGOR, *only for routing a packet inside a convex hull*; conceptually, the idea is to decompose the global visibility graph into multiple subgraphs, each subgraph representing the inner structure of each convex hull; the smaller size of the visibility graph reduces the communication overhead for building routing tables, and for setting up a routing path. However, this idea faces some challenges: first, it requires higher implementation overhead, because we need to combine two independent routing protocols into a resource constrained node; thus, extracting common features of the two routing protocols, removing conflicting or unnecessary part of the protocol, and efficiently merging them are important to design an efficient routing protocol; second, this idea fails to completely remove the communication overhead of VIGOR. Thus, the development of this idea remains as our future work.

6 Simulation Results

In this section we present performance evaluation results of our protocol executing in large scale sensor networks. These large scale sensor network results are obtained through simulations. In Section 7 we will provide experimental results for our GOAL protocol, obtained through a real hardware implementation and evaluation a real testbed consisting of 42 EPIC motes.

We implemented VIGOR [8], GOAL, GPSR [9], GOAFR$^+$ [22], and the centralized shortest path routing

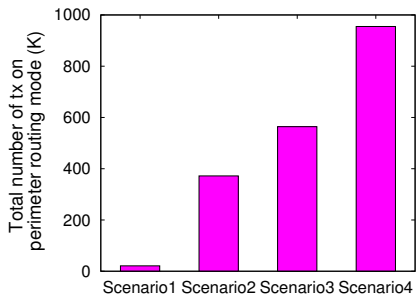


Fig. 10. Total number of packet transmissions in perimeter-routing mode. This measure represents the degree of concavity of a hole.

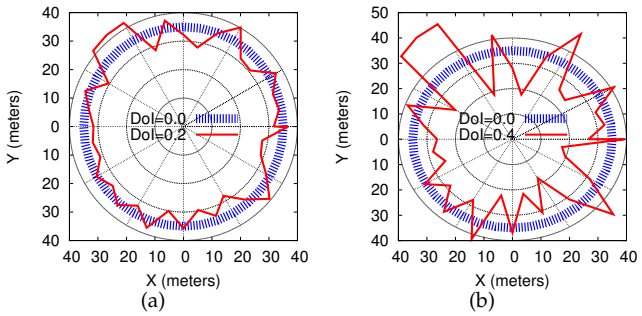


Fig. 11. Illustrations of radio ranges for different DOI values: (a) Radio range with DOI=0.2; and (b) Radio range with DOI=0.4. We adopt a radio model based on the degree of irregularity (DOI).

protocol) in C++, since we are focusing on the topological behavior of routing protocols. For this set of simulations, we randomly deployed 3,000 nodes in a two dimensional network of 1,000×1,000m^2 region. Holes with varying sizes and shapes are strategically designed; specifically, we considered four network configurations, each having different *degrees of concavity* of deployed holes, as shown in Figures 9(a), 9(b), 9(c), and 9(d). The *degree of concavity* is quantified by measuring the total number of packet transmissions in perimeter-routing mode, given the data transmissions between randomly selected 10,000 source/destination pairs. Figure 10 shows the degree of concavity for each network configuration. As shown, the four network scenarios are arranged in the increasing order of the concavity of the deployed holes.

In modeling the physical layer of each node, we adopted the radio model from [15][16]. This model is useful to account for the realistic communication channels. He et. al. [15][16] defines the *degree of irregularity* (DOI) as the maximum radio range variation in the direction of radio propagation. Figure 11(a) and Figure 11(b) show the radio range for DOI=0.2 and DOI=0.4, respectively. The default communication radius was set to 30m with DOI=0.4; the corresponding average node density was approximately 9.

We compared our protocol, GOAL, with VIGOR, GPSR, and GOAFR$^+$; specifically, we measured and compared average path stretch, maximum path stretch, communication overhead, and storage overhead among the four protocols. The focus of these comparisons is to show: (1) the low path stretch of GOAL, (2) the low communication and storage overhead of GOAL; and (3) the impact of important parameters. In particular, the comparisons with GOAL and GOAFR$^+$ serve as a base line; that is, the results of such comparisons are used for representing how much the routing protocols with constant path stretch (i.e., GOAL and VIGOR) can improve the performance compared with widely used geographic routing protocols. For this set of experiments, we varied the following parameters: h, δ, communication radius r, and the percentage of location errors p. The term h is the number of hops from the boundary of a hole, within which nodes receive the information on the hole; δ refers to the width of a bounding box used to find VON nodes [8]; the percentage of location errors p is used to simulate the error in the location of a node; that is, the location of a node, represented as two-dimensional coordinates (x, y), may change to $(x \pm p', y \pm p')$ with random probability, where $p' = \{z \in \mathbb{R} : z > 0 \text{ and } z < r \cdot p\}$.

6.1 Hop Stretch

In this set of experiments, we measured the path stretches for different routing protocols. Given a source and a destination, we define the path stretch as the following:

$$path\_stretch = \frac{measured\_hop\_count}{minimum\_hop\_count},$$

where the *measured_hop_count* is the number of hops along the routing path connecting the source and destination; the *minimum_hop_count* means the hop count for the shortest path connecting the source and destination; the *minimum_hop_count* is measured by using a centralized shortest path routing protocol. We set δ to 30m, and h to a sufficiently large number (e.g., $h = 10$ for all four scenarios) to allow the abstracted information on a hole to reach all nodes in the network. We randomly selected 10,000 source/destination pairs, and the path stretch for each pair was calculated.

Figures 12(a), 12(b), 12(c), and 12(d) depict the CDF of calculated path stretches for Scenario 1, Scenario 2, Scenario 3, and Scenario 4, respectively. We observe that, regardless of the network scenarios with the different level of concavity, the path stretches of both VIGOR and GOAL are close to 1. The difference between the path stretch of VIGOR and that of GOAL is negligible. The main reason for such low path stretches of the constant-stretch routing protocols is that those protocols, by using non-local information (e.g., abstracted information on existing holes), allow forwarding nodes to choose a neighbor such that the packet does not end up in a

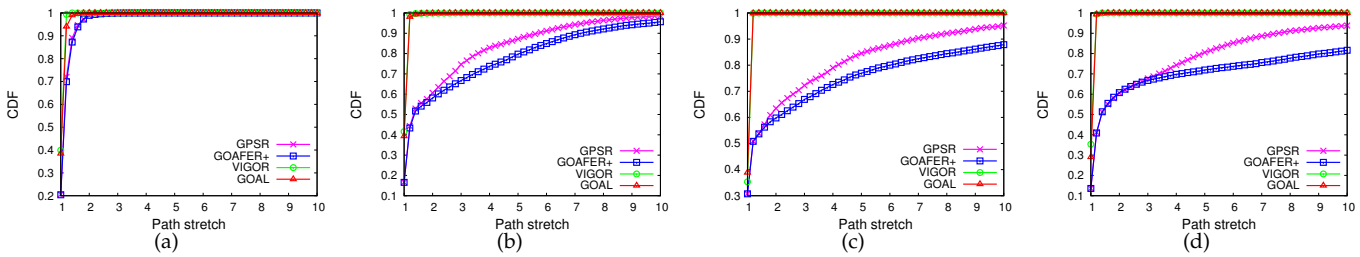


Fig. 12. The CDF graphs of path stretches for each deployment scenarios: (a) Scenario 1; (b) Scenario 2; (c) Scenario 3; and (d) Scenario 4.

local minimum. In contrast to the small path stretches of constant-stretch routing protocols, the classic geographic routing protocols, GPSR and GOAFR$^+$, exhibit large path stretches. We also observe that, as the degree of hole-concavity increases, the path stretches of GPSR and GOAFR$^+$ deteriorate. In particular, a network scenario with small holes with low concavity (i.e., Scenario 1) shows that the path stretches of GPSR and GOAFR$^+$ do not degrade much, because the chances for a packet being stuck in a local minimum is relatively smaller compared with other scenarios having holes with high concavity.

We summarize the statistical data of path stretches by means of the average and maximum path stretch. The average and maximum stretch are often used to measure the average and worst-case performance of a routing protocol, respectively. Figure 13(a) depicts the average stretches for different routing protocols in the four scenarios. As shown, the average path stretches of GOAL and VIGOR are small, regardless of the hole scheme, as both protocols prevent a packet from falling into a local minimum. Compared with the classic geographic routing protocols (e.g., GPSR and GOAFR$^+$), the average path stretch of GOAL is up to 500% smaller in our network configurations. We also observe that the concavity of deployed holes influences the performance of a routing protocol. As shown in Figure 13(a), the average path stretch increases when there are holes with more complex shapes in a network. The difference in the worst-case performance between traditional geographic routing protocols and GOAL is much larger as shown in Figure 13(b); the maximum stretch of GOAL is smaller than GPSR and GOAFR$^+$ by up to 3,800%. Additionally, we observe that, similar with the results for the average path stretch, the maximum path stretch increases, as there are holes with more complex shapes in the network.

6.2 Communication Overhead

Section 6.1 shows that the path stretch of GOAL is as small as that of VIGOR. Besides the constant path stretch property, the strength of GOAL lies in the reduced communication and storage overhead compared with VIGOR. In this section, we shall analyze the communication overhead of VIGOR to verify the energy efficiency of GOAL.

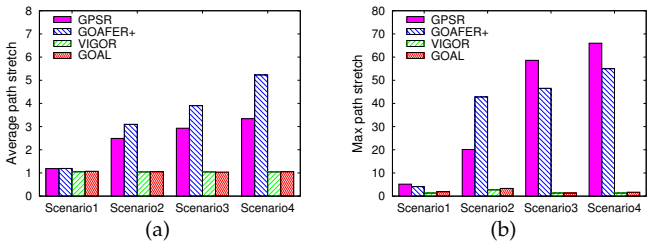


Fig. 13. Statistical summary of path stretches: (a) Average hop stretch; and (b) Maximum hop stretch. GOAL achieves significant improvements in both average and maximum path stretch.

The main source of VIGOR's communication overhead comes from two aspects of the protocol: first, each VON node must iteratively exchange routing tables with other VON nodes until its routing table converges; second, for each source/destination pair, the source node must send a control packet to the destination using the default routing protocol (e.g., GPSR) to find the entry point among the VON nodes; that is, the source node must join the overlay network of the VON nodes, by sending a control packet to the destination, before the packet transmission begins.

We first measured the total number of packet transmissions used for building routing tables by varying δ. The parameter δ defines the width of a bounding box that is used to construct VON polygons. If this value is large, we can reduce the number of total VON nodes, thereby reducing the communication overhead for constructing the routing tables. However, if δ is large, the edges of VON polygons may intersect; furthermore, the path stretch may deteriorate, because the boundary of a hole is not precisely represented. Figure 14 depicts the results. As shown, each network scenario suffers from a large number of data packet transmissions to build routing tables. We also find that this communication overhead decreases when δ increases, at the cost of imprecise representation of the boundary of a hole that leads to increased path stretch.

We then measured the total number of control packet transmissions used for setting up a routing path. Similar with the experimental configurations in Section 6.1,

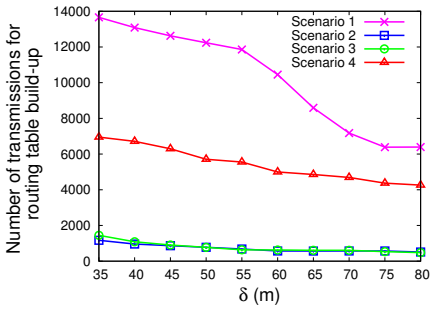


Fig. 14. Communication overhead due to routing table construction. VON nodes iteratively exchange messages until routing tables are built.

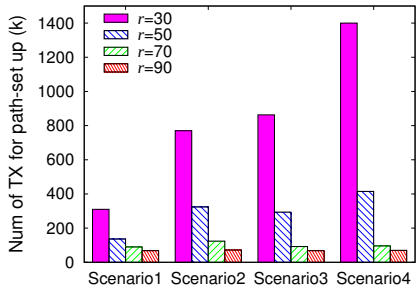


Fig. 15. Communication overhead due to path set-up. VIGOR requires each source/destination pair to exchange a control packet, using the underlying default geographic routing protocol, before actual data transmission occurs.

we randomly chose 10,000 source/destination pairs. Figure 15 shows the results. One observation is that this communication overhead is influenced by the concavity of deployed holes. The main reason is that VIGOR uses the underlying default geographic routing protocol for transmitting this control packet to a destination, and the path stretch of the geographic routing protocol degrades when there are holes of complex shapes. The communication radius r of a node is another factor that affects this type of communication overhead. As shown in Figure 15, larger communication radius allows nodes to use fewer packet transmissions.

Finally, it is important to mention that GOAL eliminates these two types of communication overhead of VIGOR, thereby achieving its energy efficiency, and its feasibility for practical deployment.

6.3 Storage Overhead

In the previous section, we analyzed the communication overhead of VIGOR. In this section, we study the storage overhead of VIGOR. In VIGOR, each VON node maintains a routing table; each entry of the routing table specifies the best neighboring VON node for a given destination VON node. In addition, each non-VON node has to maintain the *visibility set*, a set of visible VON nodes.

TABLE 1
Storage overhead of VIGOR and GOAL

	δ					
	30	40	50	60	70	80
VIGOR	74	67	60	57	55	54
GOAL	36	36	36	36	36	36

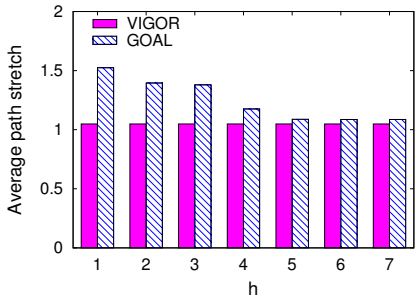


Fig. 16. The impact of system parameter h. The parameter h defines the number of hops from a hole within which the information about the hole is distributed.

In order to compute this visibility set, each non-VON node has to know all the locations of VON nodes. Thus, additional memory requirement for nodes in VIGOR is theoretically $O(V_{von})$, where V_{von} is the number of VON nodes. Compared with VIGOR, the memory overhead of GOAL is theoretically $O(V_{ext})$, where V_{ext} refers to the number of total extreme points in a network, because each node has to maintain the locations of all the extreme points when the h value is sufficiently large (note that the memory overhead of GOAL may be reduced by adjusting the h value at the cost of increased path stretch). We note that V_{ext} is smaller than V_{von}, because the extreme points are the subset of the VON nodes. Therefore, GOAL reduces the storage overhead.

Table I compares the memory overhead of GOAL with that of VIGOR in the network scenario shown in Figure 9(d). Specifically, Table I compares the total number of VON nodes of VIGOR, and the total number of extreme points of GOAL on each row of the table with varying δ values. As expected, the number of VON nodes is larger than the number of extreme points. In addition, the number of VON nodes decreases as we increase the δ value, and obviously, GOAL is not influenced by the δ value. If the δ value is sufficiently large, the number of VON nodes may be reasonably small; however, this is possible at the cost of worse path stretch, possibly with crossing edges, because imprecise representation of the boundary of a hole increases the chances of using the face routing when a packet is routed between two neighboring VON nodes.

6.4 Impact of h

The system parameter h is the number of hops from the boundary of a hole, within which nodes receive hole

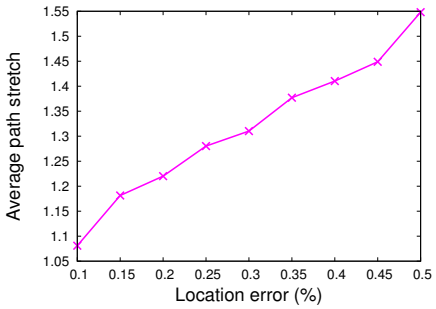


Fig. 17. The impact of location error rates. The location L of each node may randomly change in the range of $L \pm [0, r \cdot p]$ depending on the error rate p.

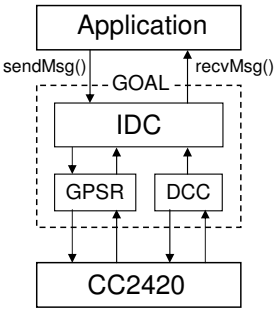


Fig. 18. System components of GOAL. GOAL consists of three main components: Intermediate Destination Locator (IDC), Distributed Convex Hull Construction (DCC), and underlying geographic routing (GPSR)

information (i.e., the set of extreme points for the hole); thus, if we use small h, the communication overhead is reduced. However, small h value increases the path stretch, because nodes that are far from a hole do not know about the existence of a hole. Those nodes send a packet along a suboptimal routing path. To see the impact of h on a path stretch, we measured the average path stretches of both GOAL and VIGOR by varying the parameter h in the network Scenario 4. Figure 16 presents the results. As shown, if we increase h, the path stretch of GOAL decreases. If h becomes larger than the maximum hop count from a hole to the boundary of the network ($h = 6$ in the network Scenario 4), the path stretch no longer decreases. Note that VIGOR is not influenced by this parameter; the path stretch of VIGOR is drawn for the comparison with GOAL. When h is small, the path stretch of GOAL is larger than VIGOR; however, as we increase h, the path stretch of GOAL becomes as small as VIGOR.

6.5 Impact of Location Error

In this section, we investigate how location errors affect our protocol. We vary the location error rate p and investigate how the path stretch changes depending on different error rates. Assume the location of a node is given as two-dimensional coordinates (x, y). This location may randomly change based on the location error p; that is (x, y) may change to $(x \pm p', y \pm p')$ with random probability, where $p' = \{z \in \mathbb{R} : z > 0 \text{ and } z < r \cdot p\}$. We measured the average path stretches for different location error rates, from 10% to 50%. Figure 17 depicts the results. As shown, the path stretch increases as we increase the error rates. A reason is that forwarding nodes might choose a neighbor based on the geographic forwarding algorithm, but the chosen neighbor might not be the neighbor that is geographically closest to the destination, due to location errors, thereby increasing the path stretch.

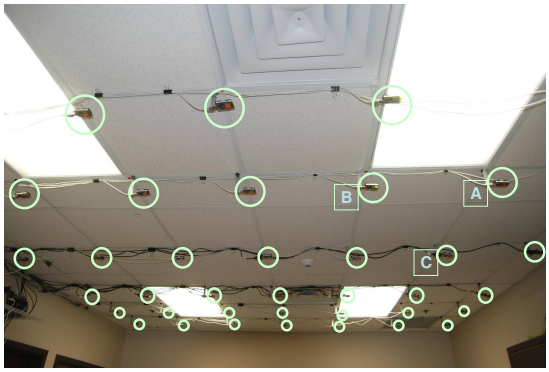


Fig. 19. A snapshot of the testbed. This testbed consists of 42 EPIC motes connected to the main PC through a USB interface. The nodes that are denoted by characters A, B, and C are used to test the link quality.

7 System Implementation and Experiments

We implemented our GOAL protocol on Epic motes running TinyOS 2.1.1. The GOAL protocol consists of three main modules: Intermediate Destination Locator (IDC), Distributed Convex Hull Construction (DCC), and the underlying routing algorithm (GPSR). Figure 18 shows the structure of GOAL protocol. The IDC module provides interfaces, i.e., *sendMsg()* and *recvMsg()*, for an application layer to send and receive a packet, respectively. The main role of the IDC module is to find the next intermediate destination, based on the received extreme points in its database. If the intermediate destination is determined, the IDC module invokes the send command of the GPSR module to send a packet to the intermediate destination. The IDC module also receives a packet from the underlying GPSR module, and computes the new intermediate destination based on the current location and the set of extreme points. The DCC module is used to abstract the information on existing holes in a network. The GPSR module simply routes a packet to the destination that is provided by the IDC module; upon receiving a

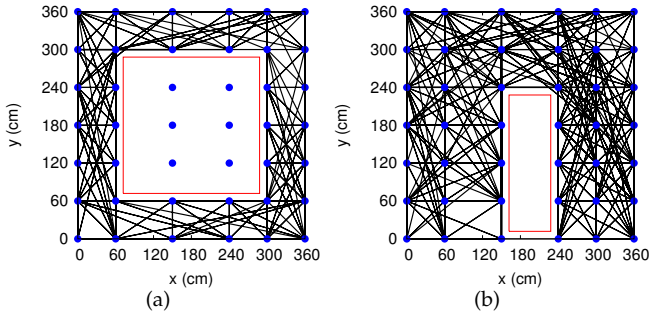


Fig. 20. The topology of testbed: (a) a topology with a large hole; (b) a topology with a small hole.

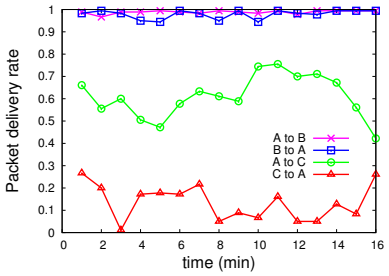


Fig. 21. The packet delivery rate of the link in the testbed. Some links are highly asymmetric.

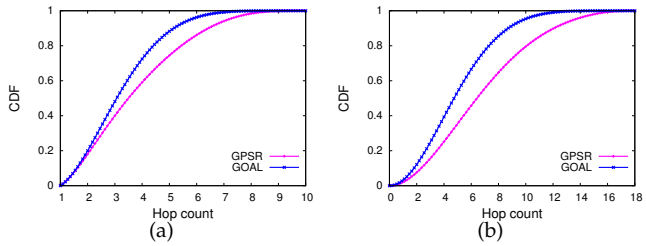


Fig. 22. Illustration of path length as the number of hop counts: (a) in a network with a large hole; and (b) in a network with a small hole.

packet from the underlying CC2420 module, this module forwards the packet to the IDC module.

Experiments were performed in an indoor testbed consisting of 42 EPIC motes. The motes are attached to the ceiling of an office area, which measures about 6m $\times$ 4.5m. The EPIC mote has a MSP430 processor running at 25MHz, 10KB RAM, 48KB program memory, and CC2420 IEEE 802.15.4 Chipcon wireless transceiver [23]. In order to prevent significant signal attenuation, each mote is detached from the ceiling approximately 6cm. The motes are connected to the PC through a USB interface, which enables us to debug, power, and program the deployed motes. Figure 19 shows the map of our testbed. We set the RF power of motes to 2 (i.e., CC2420_DEF_RFPOWER is set to 2) [24]. The resulting topologies of the testbed with a large hole and a small hole are shown in Figures 20(a) and 20(b), respectively. To simulate holes, the links that pass through the holes are not considered for our experiments.

Figure 21 shows the packet delivery rate for the links between node A and node B, and between node A and node C, as depicted in Figure 19. As shown, the link between node A and node B is relatively symmetric and shows stable packet delivery rate of nearly 100%; on the other hand, the link between node A and node C is highly asymmetric, and has unreliable packet delivery rate. To account for such asymmetric links with unreliable wireless channels, we employ link-layer retransmission. The default retransmission limit is set to 10. When a node fails to send a packet more than 10 times, it attempts to send the packet to the next best neighbor to enhance the packet arrival rate, while allowing for larger hop stretch.

In the following sections, we compare GOAL with GPSR, mainly focusing on verifying the feasibility of the practical deployment of GOAL protocol. We measure the path length, total number of transmissions, and packet delivery ratio. We vary the hole size and packet arrival rate.

7.1 Path Length

In this set of experiments, we validate that GOAL achieves a low path stretch in realistic environments. We considered two network scenarios: one with a large hole and the other one with a small hole. We randomly selected 10 source/destination pairs, and measured the hop count for each routing protocol in different network scenarios. The results are drawn as the CDF of the hop count for each routing protocol in each network scenario.

We note that it is hard to expect dramatic improvements in path length in this network configuration due to several reasons: first, the small size of the testbed restricts us from creating a sufficiently large hole with various shapes that will degrade the path stretch of classic geographic routing protocols; second, due to the limited number of motes, the network has limited hop counts. Despite these difficulties, we found that GOAL still achieves a better path stretch than GPSR.

Figure 22(a) depicts the CDF of hop count for the network scenario with a large hole. As shown, 90% of the hop counts are less than about 5 for GOAL, and 6.5 for GPSR. The average hop count is 3.5 and 4.1 for GOAL and GPSR, respectively. Overall, GOAL performs better than GPSR in terms of the path length.

Now we consider the network scenario with a small hole. Our expectation was that the performance gap between GOAL and GPSR would be smaller for this network scenario than for the scenario with a large hole, because smaller holes with the same shape reduce the chances for a packet being stuck in a local minimum.

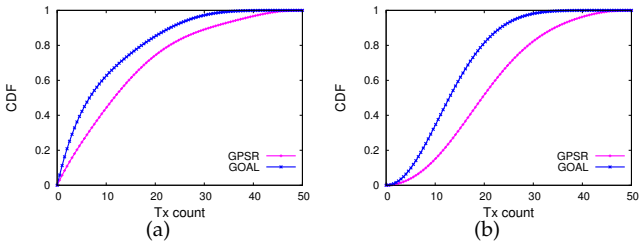


Fig. 23. The total number of packet transmissions including link-layer retransmissions: (a) in a network with a large hole; and (b) in a network with a small hole.

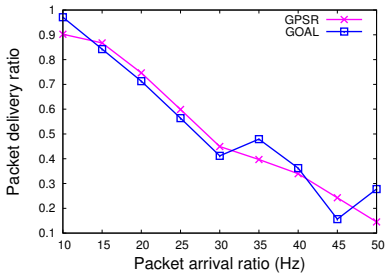


Fig. 24. Packet delivery ratio of GOAL and GPSR.

However, we found that the results are the opposite, because, as shown in Figure 20(b), in this particular network configuration, the small hole does not allow data to flow around its bottom side; thus, when data flow is directed to the bottom side of the hole, the data flow must be redirected to the upper side of the hole, creating the larger hop count. Figure 22(b) shows the results. The average hop count for GOAL is 5, while the average for GPSR is 7.1, showing a larger gap between the two protocols. Figures 22(a) and 22(b) also depict the increase in the gap between the two protocols.

7.2 Communication Overhead

Each node implements the link-layer retransmission. Due to unpredictable wireless channels, the link-layer retransmissions frequently occur. Thus, a single transmission-trial over a single hop often means multiple packet transmissions. This fact indicates that the impact of the larger hop count is profound, particularly for large-scale sensor networks. In this section, we quantify the impact of larger path length as the total number of packet transmissions including the link-layer retransmissions.

Figures 23(a) and 23(b) show the CDF of the total number of transmissions for a network scenario with a large hole, and a small hole, respectively. In both scenarios, GOAL has a smaller number of packet transmissions compared with GPSR, because the path length of GOAL is shorter than the path length of GPSR. We also observe that the scenario with a small hole shows a larger performance gap between the two protocols.

In Section 7.1, we showed that, in the network scenario with a large hole, the path length of GOAL is 15% shorter than that of GPSR; and for the scenario with a small hole, the path length of GOAL is 27% shorter than that of GPSR. Now we consider the performance improvements in terms of the total number of packet transmissions. We found that GOAL has 34% fewer packet transmissions than GPSR for the scenario with a large hole; and GOAL shows 41% fewer packet transmissions than GPSR for the scenario with a small hole. These results indicate that the performance gain obtained from the shorter path length is more significant than the actual difference in the path length when we consider the practical deployment (i.e., when we view the difference from the perspective of the total number of packet transmissions).

7.3 Packet Delivery Ratio

It was empirically shown in many sensor network applications that the packet delivery ratio (PDR) for a path has a close relationship with the number of hops of the path [25]. A simple reason is that each link of a path has its probability of packet loss; thus, if a path consists of multiple links, such probability for each link is multiplied, resulting in a higher packet loss rate. This section is designed to empirically validate that GOAL has smaller PDR, because GOAL has a smaller average path stretch. For this set of experiments, we picked two nodes such that the line segment connecting the two nodes crosses the hole in the network. We then varied the data rate from 10Hz to 50Hz.

Figure 24 presents the results. Interestingly, the PDR of GOAL is not much greater than that of GPSR; the PDR of GOAL is even lower than that of GPSR for some data rates. We investigated the causes for the unexpectedly high PDR of GOAL, and found the reason was that GPSR, when faced with high packet arrivals, chooses alternate paths. More precisely, GPSR chooses a next best neighbor when the retransmission exceeds the link-layer retransmission limit; thus, when data rate is high, and the receiver does not respond with ACK, the sender often chooses a next best neighbor, creating an alternate routing path. Considering this "multipath generation" of GPSR, GOAL provides competitive PDR. These results provide motivation for a new research direction: the development of constant-stretch multipath routing protocol in network with holes and high packet arrival rate. We plan to investigate how multiple paths, each with guaranteed path stretch, can be provided for better reliability and higher PDR.

8 Conclusions

In this article, 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. The data representing holes enable each source node to take an early detour around a hole, achieving a constant stretch. We prove the correctness for the proposed algorithms and show the constant path stretch property of GOAL. Through extensive simulations and real-world experiments on a testbed consisting of 42 EPIC motes, we demonstrate the effectiveness of GOAL and its feasibility for practical deployment in resource constrained large scale sensor networks.

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] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Proc. of ACM MOBICOM*, 2000.

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

[11] S. Rajsbaum and J. Urrutia, "Some problems in distributed computational geometry," *Theor. Comput. Sci.*, vol. 412, pp. 5760–5770, September 2011.

[12] 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.

[13] 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.

[14] M. Won, R. Stoleru, and H. Wu, "Geographic routing with constant stretch in large scale sensor networks with holes," in *Proc. of IEEE WiMob*, 2011.

[15] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *Proc. of ACM MOBICOM*, 2003.

[16] L. Hu and D. Evans, "Localization for mobile sensor networks," in *Proc. of ACM MOBICOM*, 2004.

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

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

[19] 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.

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

[21] S. M. George, W. Zhou, H. Chenji, M. Won, Y. Lee, A. Pazarloglou, R. Stoleru, and P. Barooah, "DistressNet: a wireless AdHoc and sensor network architecture for situation management in disaster response," *IEEE Communications Magazine*, vol. 48, no. 3, Mar. 2010.

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

[23] P. Dutta, J. Taneja, J. Jeong, X. Jiang, and D. Culler, "A building block approach to sensornet systems," in *Proc. of ACM SenSys*, 2008.

[24] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *Proc. of ASPLOS*, 2000.

[25] V. Shnayder, B.-r. Chen, K. Lorincz, T. R. F. F. Jones, and M. Welsh, "Sensor networks for medical care," in *Proc. of ACM SenSys*, 2005.